Video Prediction with Invertible Linear Embeddings

Robert Thomas Pottorff
Brigham Young University

Video Prediction with Invertible Linear Embeddings

Robert Thomas Pottorff

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

David Wingate, Chair
Sean Warnick
Bryan Morse

Department of Computer Science

Brigham Young University

ABSTRACT

Video Prediction with Invertible Linear Embeddings

Robert Thomas Pottorff
Department of Computer Science, BYU
Master of Science

Using the recently popularized invertible neural network we predict future video frames from complex dynamic scenes. Our invertible linear embedding (ILE) demonstrates successful learning, prediction and latent state inference. In contrast to other approaches, ILE does not use any explicit reconstruction loss or simplistic pixel-space assumptions. Instead, it leverages invertibility to optimize the likelihood of image sequences exactly, albeit indirectly.

Experiments and comparisons against state of the art methods over synthetic and natural image sequences demonstrate the robustness of our approach, and a discussion of future work explores the opportunities our method might provide to other fields in which the accurate analysis and forecasting of non-linear dynamic systems is essential.

Keywords: system identification, invertible neural networks, Hammerstein-Wiener, video extrapolation

ACKNOWLEDGMENTS

Dedicated to my wife and children. Thanks to Jared Nielsen who was an invaluable help. Additional thanks to Ryan Prenger, Bryan Catanzaro, and the other members of the Applied Deep Learning Research Group whose discussions made the details of this thesis successful.

Table of Contents

Chapter 1

Problem Statement and Contributions

This thesis aims to tackle high dimensional video frame extrapolation by modeling the problem as high dimensional non-linear system identification and using modern neural network training techniques to solve.

Formally, we consider a video sequence as an ordered tuple of T real valued frames, each denoted as $o_t \in \mathcal{R}^N$. We can frame the abstract problem of video extrapolation as learning the conditional distribution over future frames, given past frames:

$$p(o_t \mid o_{t-1},\ \ldots,\ o_0)$$

This distribution, in general, is complicated with no closed form we can use to tractably sample, score, or approximate with directly. In lieu of direct approximation, we consider transformed frames $g_\theta(o_t) = z_t$, where g is a neural network parameterized by θ, which we refer to as *embeddings* or *encodings*:

$$p(g_\theta(o_t) \mid o_{t-1}, \ldots,\ o_0) = p_\theta(z_t \mid z_{t-1}, \ldots,\ z_0)$$

Provided that g_θ is sufficiently expressive (i.e that it is capable of closely approximating a given transformation) and invertible, we can define an equivalence between a tractable distribution over observations p_θ parameterized by θ and the true distribution over frames p

using a change of variables:

$$p(o_t \mid o_{t-1},\ \ldots,\ o_0) = p_\theta(z_t \mid o_{t-1}, \ldots,\ o_0) \mid \det \frac{\partial z_t}{\partial o_t} \mid$$

This allows us to fit to the true distribution using a maximum likelihood objective on the transformed variable:

$$\max_\theta\ p_\theta(z_t \mid o_{t-1}, \ldots,\ o_0) \mid \det \frac{\partial z_t}{\partial o_t} \mid \tag{1.1}$$

In this work, we use an invertible neural network as the model class for $g_\theta(o_t) = z_t$ and a linear time-invariant dynamic system (LTI) to define the tractable likelihood p_θ. To our knowledge, this is the first work to demonstrate successful learning in reversible flow networks using an LTI prior and one of the few works in video frame extrapolation to avoid making any assumptions about the data distribution.

Our main contribution, an invertible linear embedding (ILE), combines invertible neural networks and a latent linear dynamical system to explicitly model the true distribution. By leveraging an invertible function approximator and the change of variables formula, frame prediction likelihood can be precisely equated with the likelihood of an observation from a linear dynamical system.

Chapter 2

Video Prediction

Video frame extrapolation (sometimes called video prediction) is the estimation of future frames conditioned on past ones. Thanks primarily to the ubiquity of image and video sensors, this prediction plays a central role in diverse fields such as self-driving vehicles and reinforcement learning. In robotic applications like these, cheap passive sensors like video cameras that provide a rich detailed summary of the world (truly unparalleled by any modern sensor) are invaluable and learning to understand the evolution of this sensor data over time is a critical component of perception, planning and control.

2.0.1 Representation Learning

Frame prediction also offers a well-posed unsupervised objective for representation learning. Any successful algorithm must have extracted salient features useful for describing both the content and dynamics of a scene. To some degree, video prediction and representation learning are essentially the same task. With the right representation, prediction is easy, stable, and efficient; with the wrong one, it may be difficult or impossible [1].

The grand vision of representation learning is to understand how useful encodings can be learned. Although there is no consensus as how this should be done, a video prediction objective offers a well-posed unsupervised task that must describe a rich understanding of the world and. As a result, video prediction may provide insight into how these productive representations may be learned [18]. Thus, in addition to practical applications successful video prediction also offers much in the way of furthering our theoretical understanding of how artificial agents need to perceive the world.

2.0.2 Video Prediction as Probabilistic Modeling

Video prediction as a task can be modeled as accurate approximation of the conditional probability distribution over future frames. This stochasticity presents a challenge to prediction tasks. Consider a video of a falling leaf - it may float in one direction or another with little indication of future direction but predicting only the average direction makes little sense. A truly accurate model would attempt to model the entire distribution over frames.

To fit this distribution of future frames, some approaches use invalid assumptions. For example, some minimize pixel-wise mean squared error (MSE) reconstruction loss which as we will discuss assumes, among other things, that pixels are independent. Others optimize a lower bound, typical of variational auto-encoders (VAEs). Still others use generative adversarial (GAN) discriminators to approximate the likelihood in the data domain. However, none of these approaches model the true distribution. We consider these approaches in slightly more detail in the subsequent section.

2.1 Related Work

Frame prediction is an well studied problem with decades of research. We summarize important relevant bodies of work here.

2.2 Flow Based Methods

In flow based methods, the model extrapolates optical flow (pixel deviations) that can be used for backward sampling to "warp" the frame at $t-1$ to the frame at t. Pixel-based optical flow methods unilaterally assume that object materials are Lambertian (perfectly matte), without reflections, dis-occlusion (when new pixels are revealed as the result of an object uncovering another), or transparency. Despite the error in these assumptions, flow based methods represent the state-of-the-art in short-term frame prediction applications where sharp edges are highly valued such as video compression and feature estimation in robotics [5, 15, 19, 22, 24, 31]. Thanks to the compounding nature of repeated bi-linear sampling and an inability to handle dis-occlusion, flow based methods are unable to provide the stable predictions needed for prediction horizons beyond a few frames, leaving longer-term prediction to latent-variable based generative models.

2.2.1 Connection to Linear Dynamic Systems

Optical flow methods are essentially a sparse, structured, linear dynamical system operating in pixel space directly $x_{t+1} = A_t x_t$ with the state x as the flattened frame, and the matrix A has only four non-zero entries corresponding to the four the bi-linear interpolation coefficients, the location of which are determined by the optical flow. Although efficient, the required structure and sparsity in A results in a poor approximation to the true non-linear dynamics, a limitation I plan to overcome by learning dense dynamics in a latent space coupled with a decoder.

2.3 Generative Models

In contrast to optical flow methods, generative models hallucinate the entire frame from a latent representation. This enables the model to imagine new pixels that appear as the result of dis-occlusion. The most common assumption made by these models is induced via reconstruction loss. Reconstruction loss is almost unilaterally used as a way to describe the likelihood of prediction error in pixel-space but erroneously assumes independence between pixels. We discuss this in great detail in the subsequent section. Generative methods can loosely be divided between Generative Adversarial Network (GAN), Variational Auto-Encoder (VAE), and factorization methods.

In contrast to VAE approaches, GANs employ a learned generator to produce an imagined sample, and a learned discriminator to determine the likelihood that the imagined sample came from the true distribution [6, 9, 17, 32]. Note that the discriminator is not estimating the likelihood of the produced image in the true distribution, but only the likelihood that it is a true sample or not. VAEs on the other hand use an auto-encoder architecture to approximate the true distribution over frames conditioned on a latent code $p(x|z)$. Because the true distribution is intractable, it is approximated by maximizing a lower bound [11]. Both of these methods, while extremely powerful, suffer from the limitation that the true likelihood is never computed. Common failure modes, which differ between the methods, include extreme difficulty in training, mode collapse (when the generator produces the same memorized that perfectly fools the generator) or blurry reconstruction from the independent pixel assumption. Hybrid approaches that use generic reconstruction loss augmented with GAN discriminator losses or VAEs to improve generalization or fill in estimated dis-occlusion are common [14, 18, 27].

2.4 Reconstruction Error and Implicit Assumptions

To distinguish between a large class of prior work and our main contribution, we highlight a distinction between a common candidate objective function and the true distribution described in Equation 1.1. A common framework for video prediction involves learning an encoding function e_θ, a separate decoding function d_θ, and a transition function f_θ. Learning the decoder has the practical purpose that the system avoids perfectly predictable but not particularly useful minima such as $e_\theta = 0$. For example, optical flow models use bilinear sampling as f_θ, VAEs have a literal encoder and decoder, and GAN approaches have a generator as a decoder with a simultaneously trained encoder.

A typical loss in this framework is usually defined:

$$\min_\theta \; \alpha \| f_\theta(e_\theta(o_t)) - e_\theta(o_{t+1}) \| + \beta \| d_\theta(e_\theta(o_t)) - o_t \|$$

When using L^2 as the norm, minimizing this candidate objective function is equivalent to maximum log likelihood learning under three assumptions.

1. First, that the conditional distribution of the error *of the embedding* is an isotropic Gaussian.
$$p_\theta(e_\theta(o_{t+1})|e_\theta(o_t)) = \mathcal{N}(e_\theta(o_{t+1}), \alpha^{-1})$$
2. Second, that the input images are isotropic Gaussian with a mean defined by the decoder
$$p(o_t|e_\theta(o_t)) = \mathcal{N}(o_t, \beta^{-1})$$
3. Third, that the determinant of the encoder's Jacobian is 1.

If our observations are image pixel intensities, these may not hold. While the first assumption is relatively mild (it is valid given any sufficiently expressive encoder), the second and third are not. The term $\|d_\theta(e_\theta(o_t)) - o_t\|$, which we call *reconstruction loss*, compares an observation o_t with its reconstruction $d_\theta(e_\theta(o_t))$ using pixel-wise mean-squared error, known to perform poorly in the case of translations and brightness variations. More generally, it completely ignores valuable higher-order information in images: pixel intensities are neither independent nor do they share the same variance. The third assumption is likewise almost certainly not true for traditional auto-encoders. Put simply, this loss implies false assumptions and results in a different objective than the one we would truly like to minimize.

2.5 Hammerstein-Weiner System Identification

In our work, in addition to the rich history of frame prediction research, we also consider connections to control theory. Conceptually, modeling a nonlinear dynamic system as an tuple of an encoding function and a linear (possibly time-invariant) dynamic system is known in control literature as a Hammerstein-Wiener block model[1] [10]. This literature has historically focused on low-dimensional systems using methods which do not scale well to high dimensional

[1]Technically the model presented here is a Wiener model, but we consider the connection to the generalized model in the literature to be important.

systems like video. We extend this body of research here with neural network techniques for high dimensional systems such as image sequences.

Chapter 3

Invertible Neural Networks & Linear Dynamic Systems

In this chapter, we look at how an invertible neural network can be constructed and used as our invertible function g_θ. Recall that the goal of video prediction is to estimate:

$$p(o_T|o_{t-1}, \ldots, o_0)$$

Which we do by learning an invertible function $g_\theta^{-1}(o_t) = z_t$ and computing the likelihood in the space of the tractable changed variable:

$$p(o_t \mid o_{t-1},\ \ldots,\ o_0) = p_\theta(z_t \mid o_{t-1}, \ldots,\ o_0) \mid \det \frac{\partial z_t}{\partial o_t} \mid$$

This particular implementation is common to a body of work on reversible flows, a relatively new approach to deep generative modeling [3, 4, 12] which leverage a special functional form for an invertible neural network to learn generative models of complex distributions.

We consider a generative model using a known parameterized distribution $p_\theta(z)$ and a deterministic function $g_\theta(z)$:

$$z \sim p_\theta(z)$$

$$o = g_\theta(z)$$

where $g_\theta(z)$ has the compositional form

$$g_\theta^{(N)}(g_\theta^{(N-1)}(\cdots g_\theta^{(0)}(z)\cdots))$$

in which each successive layer operates on the output of the layer before (abbreviated notationally as $g_\theta^i(h_{i-1})$). The change of variables formula enables us to relate:

$$\log p(o) = \log p_\theta(g^{-1}(z)) + \sum_{i=0}^{N} \log \mid \det \frac{\partial h_i}{\partial h_{i-1}} \mid,$$

with $h_0 = o$ and $h_N = z$. Because p_θ is tractable, we need only for the determinant of each layer's Jacobian to be tractable to efficiently compute the density $\log p(o)$.

Borrowing on the early work in this field, we use the following technique for $g_\theta^{(i+1)}(h_i)$ called an *affine coupling* which makes this determinant easy to compute:

$$\begin{aligned} h_i^{\text{left}}, h_i^{\text{right}} &= \text{split}(h_i) \\ s_i &= f_i(h_i^{\text{left}}) \\ h_{i+1}^{\text{right}} &= s_i \odot h_i^{\text{right}} + b_i(h_i^{\text{left}}) \\ h_{i+1} &= P_i \begin{bmatrix} h_i^{\text{left}} \\ h_{i+1}^{\text{right}} \end{bmatrix} \end{aligned}$$

where $h_i \in \mathcal{R}^D$ is the layer input, $\odot$ is the element-wise product, f_i and b_i are arbitrary neural networks (not necessarily invertible), and P_i is a unimodular matrix which mixes elements between the two halves of h_i. Although this may seem intimidating, the computation is straightforward: using half of the layer's input we learn to produce an affine transformation to apply to the other half. This operation is invertible, and the log-determinant of this layer is simply:

$$\log |\det \frac{\partial h_i}{\partial h_{i-1}}| = \log \sum_{j=0}^{D/2} |s_{ij}|.$$

The log-determinant of the entire encoding function is the sum of these terms for all layers i:

$$\log |\det \frac{\partial g_\theta^{-1}(o_t)}{\partial o_t}| = \log \sum_{i=0}^{N} \sum_{j=0}^{D/2} |s_{ij}|.$$

Taken together, this functional form enables us to define our decoding function $g_\theta(z)$, its exact inverse, and an efficient computation for the log-determinant of its Jacobian.

One particular downside to this method is a near-prohibitive memory and parameter requirement. Because the input and output dimensions are equal, and the individual affine transformations are so simple these reversible networks require a large number of layers with a large number of parameters. This complexity can make gradient propagation slow and optimization difficult corresponding to long training times. As the body of literature on reversible flows expands and new techniques are developed we expect to see improvements that would also apply to our model.

3.1 Linear Dynamic Systems

Recall that in our primary objective function, in addition to being able to learn $g_\theta^{-1}(o_t) = z_t$ with a tractable way to compute the determinant of the Jacobian as described the previous section, we must also be able to define a tractable computation for the distribution $p_\theta(z_t|z_{t-1}, \ldots z_0)$. In this section we introduce linear time-invariant systems as the structure for this density function.

A natural model for the evolution of a vector-valued observation is that of a linear dynamic system:

$$x_t = Ax_{t-1} \qquad z_t = Cx_{t-1} + \gamma_{t-1} \qquad \gamma_{t-1} \sim \mathcal{N}(0, I)$$

Where x_t represents the hidden state, and z_t the observation at that hidden state. In this work, we assume that this system is *time-invariant*; however, we note that it is possible to extend this model to not only include time-varying dynamics, but also inputs, process noise over the hidden state, or noise distributions with different distributions, but omit them in this work for simplicity.

Linear Time Invariant (LTI) systems define a conditional distribution with tractable density over observations:

$$p_\theta(z_t|z_{t-1}, \ldots, z_0) = \mathcal{N}(CA^t {x^*}_0, I)$$

where x_0^* is the result of optimal latent state inference. For LTI systems, this is the result of a Kalman filter when conditioned on only past observations, and the Kalman smoother when conditioned on both past and future observations [30]. Although many algorithms [26] exist to compute the optimal smoothing estimate x^* they can all be shown to produce the same least squares estimate:

$$\begin{aligned} x_0^* &= \arg\max_{x_0} \left\| \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_{T-1} \end{bmatrix} - \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{T-1} \end{bmatrix} x_0 \right\|_2^2 \\ &= \arg\max_{x_0} \|\mathcal{Z} - \mathcal{O}x_0\|_2^2 \\ &= \mathcal{O}^+\mathcal{Z} \end{aligned}$$

The tractable density function is:

$$p_\theta(z_t|o_{t-1}, \ldots, o_0) = \mathcal{N}(CA^t\mathcal{O}^+\mathcal{Z}, I)$$

where we use M^+ as the pseudoinverse of M. The efficiency of optimal hidden state inference is one of the motivating factors behind our choice of a linear model for the latent evolution of embeddings.

Using a linear dynamic system as the transition model for our system introduces two major assumptions. The weaker assumption is the Markov property, that future observations are independent of past observations when conditioned on the hidden state. The stronger assumption is that the future hidden states are a linear mapping from past states and that this mapping remains constant through time. Although the linear dynamics prior may seem quite restrictive, LTI systems are surprisingly expressive and have been shown to model the latent dynamics of many high dimensional models [2, 16]. A key theoretical insight in control literature proves the existence of an infinite dimensional linear operator, the Koopman operator, for some nonlinear projection of all nonlinear dynamic systems [13]. When choosing a large latent hidden dimension, we are approximating this infinite-dimensional operator. So while the modeling assumption made by a linear dynamical prior is almost certainly not true, a large enough state space is a good approximation and will demonstrate the viability of ILE for difficult non-linear systems. Future work could explore options for more expressive yet tractable time-variant dynamic system models.

3.2 Parameterization

Given the numerical instability induced from computing a least-squares solution in our training loop, the parameterization of the linear dynamic system is of critical concern. In particular, we must parameterize the learning method to maintain stable state transition matrices A. A stable discrete-time linear dynamic system is one in which the singular values are less than one so A^t does not explode for large t. A common necessary condition is that the largest singular value, or the spectral radius of A denoted $\sigma(A)$, is less than one.

One feature of LTI systems that we can exploit to ensure training stability is that, for a given state-space parameterization A, C, there exist an infinite number of equivalent parameterizations that correspond to the same input-output relationship and thus produce the same observation sequences some of which will be more efficient to regularize during training. This property can be explained intuitively: one can rotate the hidden state space by some transformation T, evolve the state in this transformed space before de-transforming observations with T^{-1}. In practice, this means we can consider any parameterization for A which has the eigenvalues of the true system provided we also learn an unconstrained C matrix.

Because the stability of a linear dynamic system is characterized by the magnitude of the eigenvalues of A, we can choose T so it is easy to compute and restrict these values. If the true system $A^* = Q\Lambda Q^{-1}$ with a complex diagonal matrix of eigenvalues Λ and matrix of eigenvectors Q, then we imagine $T = Q$, implying that we instead learn $A = \Lambda$. This technique both decreases the number of learnable parameters in A while also making enforcing stability relatively trivial.

Jordan Normal Form

The primary issue with learning $A = \Lambda$ is that Λ as the eigenvalues of a real matrix will come in complex conjugate pairs[1]. However, Real Jordan Normal Form (JNF) offers a simple solution. By splitting the real and imaginary parts, we can construct an all-real matrix for which matrix multiplication simulates complex multiplication with this constraint. The

[1]If we assumed that A^* was symmetric, the eigenvalues would have no imaginary components and we could instead simply learn a diagonal real matrix Λ.

following represents a 4×4 example:

$$A = \begin{bmatrix} \alpha_0 & \beta_0 & 0 & 0 \\ -\beta_0 & \alpha_0 & 0 & 0 \\ 0 & 0 & \alpha_1 & \beta_1 \\ 0 & 0 & -\beta_1 & \alpha_1 \end{bmatrix}$$

This form does have its drawbacks. In scenarios where any imaginary components are actually zero, then there should be an additional unique real component. Although somewhat inelegant, the negative impact of this scenario can be mitigated by simply increasing the dimensionality of A. Additionally if $\alpha_0 = \alpha_1$ and $\beta_0 = \beta_1$ true Jordan blocks should additionally have a one in the off-diagonal corresponding to eigenvalues with multiplicity greater than one. In practice this is not an issue as it is difficult to produce *exactly* identical eigenvalues via gradient descent.

Recall that the goal is to learn stable A matrices. Although there are many ways to ensure that the magnitude of each eigenvalue in the JNF does not exceed 1, we found the following re-parameterization to be effective, using θ_α and θ_β as vectors of unconstrained real-valued parameters to produce the vectors of properly constrained real and imaginary components α and β:

$$\alpha = max((1-\epsilon) - |\theta_\alpha|, 0)$$
$$\beta = max(1 - |\theta_\beta|, 0) * \sqrt{1-\alpha^2}$$

where $\epsilon = 10^{-14}$. This particular transformation ensures that every unique real parameter pair $\theta_\alpha, \theta_\beta$ corresponds to a unique complex eigenvalue. The small epsilon subtraction ensures that we never compute $\sqrt{0}$. In our implementation, ϵ is chosen such that when we compute α and β with double precision, and then cast to single precision floating point we avoid $\sqrt{0}$ and allows $\alpha = 1$.

3.3 Shur's Decomposition

Similar to Jordan Normal Form, Shur's decomposition expresses a matrix as $A' = ZUZ^{-1}$ only with U as an upper triangular complex matrix instead of a diagonal one. The diagonal of U are the eigenvalues of A' and thus assuming $T = Z$ we can learn $A = U$ Using the same block-diagonal form and constraints from Jordan Normal Form for the diagonal. This decomposition is an over-paramterized way to express A relative to Jordan Normal Form. Although over-parameterization has has been shown to be useful to the success of gradient descent algorithms and system identification[8], in our experience we did not see any major difference in performance between Shur's Decomposition parameterization and Jordan Normal Form.

3.4 Singular Value Decomposition

In addition to the parameterizations above, it is also possible to learn the Singular Value Decomposition of A, and place limits on the singular values directly. The SVD defines a unique[2] factorization of a matrix A using a unitary matrices U and V (matrices whose inverse is equal to their conjugate transpose) and a diagonal matrix of positive values Σ generally referred to as singular values.

$$A = U\Sigma V^*$$

The singular values Σ of a matrix A are equivalent to the magnitude eigenvalues of A^TA, and as a result are ideal for enforcing stability. In our work we use a real valued unconstrained parameter vector θ_σ and define Σ such that $\Sigma = 1 - clamp(abs(\theta_\sigma), 0, 1)$. Learning unitary matrices U and V with unconstrained parameters is decidedly more difficult.

[2]To a permutation, although generally the singular values are ordered to ensure the decomposition is unique

3.5 Parameterizing Unitary Matricies

Although important for our parameterization here, unitary matrices have also been shown to help improve gradient propagation and initialization in neural networks [21], and long horizon recurrent models[28]. We borrowed insights from this literature to develop fast computations which construct these matrices. We consider three practical methods for constructing a unitary matrix from unconstrained parameters and summarize the methods here:

1. **Product of Householder matrices** The details of this algorithm are better understood in the context of a large body of literature, but the core idea is that all orthogonal matricies can be represented as the product of N householder reflection matrices.

2. **Cayley's Transform** Author Cayley showed how all orthogonal matricies Q have a one-to-one correspondance with skew-symmetric matricies A such that $Q = (I-A)(I+A)^{-1}$. Constructing a skew-symetric matrix S from unconstrained matrix M is relatively easy: $S = M - M^T$, thus the parameterization becomes $Q = (I-(M-M^T))(I+(M-M^T))^{-1}$. In practice, the inverse operation was a bottleneck and the householder method was faster.

3. **LQ decomposition** The simplest method is to simply take the LQ decomposition of an unconstrained matrix, and use only the Q product. The utility of this method will depend greatly on the differentiability and efficiency of the LQ algorithm in a particular implementation.

Chapter 4

Invertible Linear Embeddings

4.1 Learning Problem Refined

Recall the learning problem described earlier:

$$\max_{\phi}\ p_{\phi}(y_{T+1} \mid y_0,\ y_1,\ \ldots,\ y_{T-1}) \mid \det \frac{\partial y_t}{\partial o_t} \mid$$

We now present our primary contribution: the invertible linear embedding.

Using an invertible neural network as our encoding and decoding function $g_\theta(o)$ and $g_\theta^{-1}(z)$, and an LTI dynamic system as described for the conditional distribution, we can derive our final loss function:

$$\begin{aligned}
\mathcal{L} &= -\log p_\theta(o_t \mid o_{t-1}, \ldots, o_0) \\
&= -\log[\ p_\theta(g_\theta^{-1}(o_t) \mid o_{t-1}, \ldots, o_0)|\det \frac{\partial g_\theta^{-1}(o_t)}{\partial o_t}|] \\
&= -\log\ p_\theta(z_t \mid o_{t-1}, \ldots, o_0) - \log\ |\det \frac{\partial z_t}{\partial o_t}| \\
&= -\log\ \mathcal{N}(CA^t\mathcal{O}^+\mathcal{Z}, \Sigma) - \log\ |\det \frac{\partial z_t}{\partial o_t}|
\end{aligned}$$

Which results in:

$$\mathcal{L} = \frac{1}{2}\|\mathcal{Z} - \mathcal{O}\mathcal{O}^+\mathcal{Z}\|_2^2 - \log\ \sum_{i=0}^{N}\sum_{j=0}^{\frac{D}{2}} |s_{ij}| \tag{4.1}$$

When minimized using sufficiently expressive $g_\theta(z)$, A, and C parameters, this loss function corresponds to the exact maximum likelihood model of a video sequence which is assumed to have latent linear dynamics.

We can describe the function of these two terms intuitively. The first term (the *predictive error*) is the result of encoding each frame independently, solving for the best possible LTI dynamic system trajectory, and applying gradient descent to minimize any error. The more the embeddings behave as a linear system, the lower the predictive error. The second term (the *log-determinant*) encourages the embeddings to be large, preventing the first term from collapsing to easy-to-predict but useless trajectories such as $z_t = 0$. Although it may seem like a strange regularization to "maximize the embedding values", the application of change of variables and strict invertibility ensures that this is the correct way to learn a mapping between our assumed latent model, and the true observations in image space.

4.2 Addressing the Scale Ambiguity

When learning both the encoding function and the dynamic system parameters simultaneously, there is an ambiguity between the scale of the embedding and the scale of the dynamic system when the covariance is learned. As an illustrative example, consider the following system:

$$y_t = \gamma_t f_\theta(o_t) \qquad \hat{y}_t = \gamma_t C x_t$$

A scaling ambiguity occurs when we try to learn the covariance of the error in addition to the other parameters of our network, i.e when the predictive loss becomes:

$$\begin{aligned}
\log p(y_t|y_{t-1}) &\propto (y_t - \hat{y}_t)^T \Sigma (y_t - \hat{y}_t) \\
&= (\gamma_t f_\theta(o_t) - \gamma_t C x_t)^T \Sigma (\gamma_t f_\theta(o_t) - \gamma C x_t) \\
&= \gamma_t^2 (f_\theta(o_t) - C x_t)^T \Sigma (f_\theta(o_t) - C x_t)
\end{aligned}$$

The γ_t^2 term, which induces downward pressure on the embedding magnitudes when Σ is constant, can be absorbed as a learned Σ adjusts during training. This effectively removes its impact, but leaves behind the upward pressure on magnitudes from the $\log \gamma_t$ term, which will result in the system maximizing γ_t, rather than prediction error. In practice, this results in a runaway scale of the embeddings and numerical issues.

To address this we model the γ^{-1} as another layer in our invertible network which we simply adds another term to our loss function:

$$\mathcal{L} = \log p(\gamma_t^{-1} y_t | \cdot) + \log |\det \frac{\partial y_t}{\partial o_t}| - \log \gamma_t$$

If we assume the covariance of the error to be the identity (i.e unlearned), the log predictive loss (when the model for error is Gaussian) is proportional to $(y_t - \hat{y}_t)^2 = \gamma^2 (f_\theta(o_t) - Cx_t)^2$ and thus scales with γ. As γ decreases, so too does the predictive loss. This induces downward pressure on γ, which in turn induces downward pressure on the scale of the embeddings y. This downward pressure is somewhat mitigated by the upward pressure on the magnitude of the embeddings from the negative log determinant term, but is asymmetric: the downward pressure scales with γ^2 while the upward pressure scales with $\log \gamma$. Nevertheless this implies that there exists an equilibrium between these two forces and that training should be stable regardless of the scale of γ.

In practice, we found that adjusting for γ in the loss improved training stability and wall clock time even when the covariance is held constant during training. Although training is stable without the adjustment, the asymmetry results in a bias toward smaller magnitude embeddings than if γ was fixed. This biases the norm of the decoder's Jacobian to be large[1]. This unfortunately has the unintended consequence of translating small error in the embedding space to large error in the image space. Although asymptotically this issue should equalize, by holding γ constant, this effect is lessened earlier in training.

[1]If the magnitude of the embeddings is small, and the outputs are large, then the norm of the Jacobian of the decoder must be large.

Although γ_t could be learned, we used $\gamma_t = \frac{1}{N}\|y_t\|_1$. We also found that the L^1 norm performs better than the L^2 norm[2]. But we note that "better" in this case refers to wall-clock time needed for quality predictions and not asymptotic correctness.

4.3 Algorithm

Combining all of these, we can now formulate our method, Invertible Linear Embeddings, as an algorithm for video prediction

Algorithm 1 Invertible Linear Embedding

```
 1: Returns the following:
 2:     $g_\theta$: a learned invertible neural network
 3:     $A$: a learned state transition matrix
 4:     $C$: a learned observation matrix
 5: while $\mathcal{L}$ is not minimized do
 6:     Sample $o_0, \ldots, o_{T-1}$ frames
 7:     for $t = 0, \cdots, T-1$ do
 8:         $z_t = g_\theta^{-1}(o_t) \in \mathbb{R}^D$
 9:         $s_t = |\det \frac{\partial z_t}{\partial o_t}|$
10:     end for
11:     $\mathcal{Z} = \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{T-1} \end{bmatrix} \quad \mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{T-1} \end{bmatrix}$
12:     $x_0^* = \mathcal{O}^+\mathcal{Z}$
13:     $\hat{\mathcal{Z}} = \mathcal{O}x_0^*$
14:     $\gamma = \frac{1}{D}\|\mathcal{Z}\|_1$
15:     $\mathcal{L} = \frac{1}{2}\|\gamma^{-1}(\mathcal{Z} - \hat{\mathcal{Z}})\|_2^2 + \sum_t^T[\log s_t] - \log\gamma$
16:     Take gradient step in $A$, $C$, $\theta$ to minimize $\mathcal{L}$
17: end while
18: $o_T = CA^T x_0^*$
```

[2]Presumably because it better propagates small gradients in each dimension of y_t.

Chapter 5

Experimental Results

5.1 Datasets

We show results for both a synthetic and realistic dataset. The synthetic data, entitled Bouncing-MNIST, is generated using the Moving Symbols algorithm, a published benchmark designed to support the objective study of video prediction networks [25, 29]. Each video sequence samples an MNIST digit, assigns it an initial trajectory, and simulates elastic collisions with the image boundary.

The realistic sequences are sampled from UCF Sports Action [20, 23]. This dataset contains video sequences of various sports such as diving, running, horseback riding, and golfing.

5.2 Network Topology

Our network is most similar to that used by [12], but without 1×1 convolutions, or the act-norm operation. We used 4 blocks of 10 affine-coupling layers each, where each block has an early connection out to the final embedding. Our non-invertible networks used at each step of flow were simple 3-layer networks of 3×3 convolution with two output channels for the affine transformation parameters and 512 channels in the center. For comparison, we implement the adversarial training algorithm of [18], which is known for its sharp image quality.

5.3 Results

Method	First Frame		Fifth Frame	
	PSNR	SSIM	PSNR	SSIM
Invertible Linear Embedding	**23.5**	0.92	**17.4**	0.69
Adversarial Training	20.6	**0.95**	12.1	**0.83**
Last Input	17.3	0.76	14.5	0.67

Figure 5.1: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) scores, taking the mean over 100 held-out test sequences. We generate future frames $o_1, o_2, ..., o_5$ and calculate scores on o_1 and o_5 to measure both immediate and longer-horizon prediction quality. We again note that our approach does not explicitly minimize the mean squared error between predicted frames and ground truth.

We evaluate our algorithm by comparing against adversarial training in three ways: qualitatively through examples, with peak signal-to-noise ratio (PSNR), and with the structural similarity (SSIM) index [29]. Statistical results are reported on the synthetic dataset.

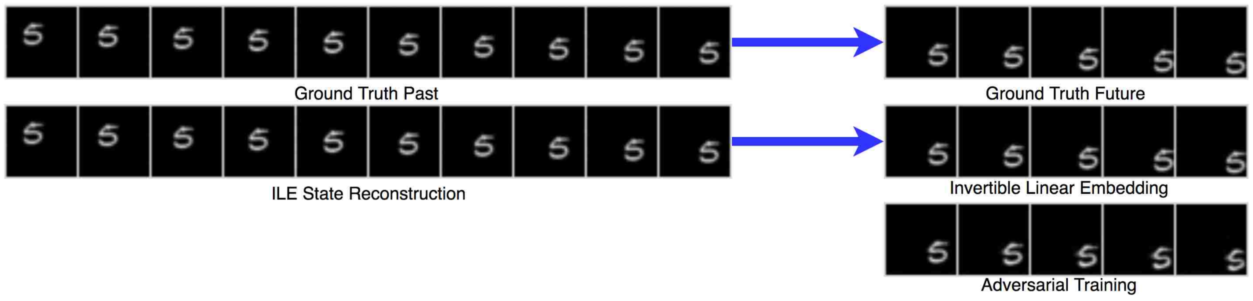

Figure 5.2: The Bouncing-MNIST dataset, modeling elastic collisions which preserve object shape.

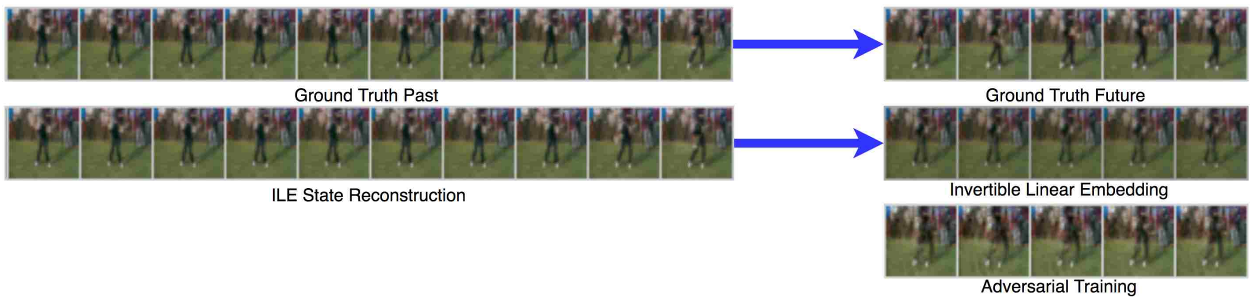

Figure 5.3: The UCF Sports Action dataset, modeling the progression of a golf swing.

Although adversarial training has a slight advantage in SSIM, the ILE algorithm outperforms it in PSNR. The difference is especially pronounced over a longer time horizon. Adversarial training maintains crisp shapes, yet lacks accurate motion projections over even moderate time horizons. After five frames it performs significantly worse than the naive baseline. ILE maintains a reasonable representation of the digit shape, and excels at motion projection over a long time horizon, even accurately predicting bounces off image boundaries. This suggests that the nonlinear dynamic system is being fit quite well.[1]

While adversarial training performs well on sequences where the motion is strictly linear, such as those pictured, it performs poorly in motion that is nonlinear in pixel space. For example when the digit bounces off a wall or when a golf club accelerates in the frame. In contrast, ILE models all motion sequences well, suggesting better generalization ability.

[1]The adversarial training PSNR scores are lower than those reported in [18] because the synthetic dataset has much more motion than the UCF-101 dataset, which the original paper used as a benchmark. In our tests, digit velocity is up to 3 pixels/frame in each direction. However, the high velocity is intentional; a quality benchmark for video prediction should use sequences where motion is noticeable.

Chapter 6

Directions for Future Work

Our work moves toward exact maximum likelihood optimization to improve performance in video prediction. We present here what we consider to be natural next steps, and the implications they might have.

6.1 Action-conditional Latent Structure

By extending the model of the hidden dynamical system to include an action u and linear mapping B it becomes possible to use ILE for model based reinforcement learning and optimal control.

6.2 Time Variant Models

Additionally a simple extension to our model which may prove promising is to learn a time or state-conditional state-transition matrix A_t in lieu of the constant A presented. This particular extension could be done using any standard autoencoding neural network architecture as a JNF state transition matrix is diagonal and invertibility is not a requirement. Although time-varying linear dynamic systems are more difficult to analyze, they are models of much greater capacity and therefore could be better suited for difficult problems that would require infinite, or near-infinite dimensional state dimensions.

6.3 Scaling to larger frame dimensions

Invertible networks, perhaps as a direct result of the difficult task of modeling the entire unknown distribution over video frames, are large and difficult to train. In particular, memory usage in our model even for these relatively small frame sequences was a computational constraint. Architectural improvements such as those recently proposed by Grathwohl et al. could extend our results into images approaching modern video resolutions.

Chapter 7

Conclusions

We have presented the invertible linear embedding, which provides exact maximum likelihood learning of video sequences. Our key contribution is to combine invertible networks with linear dynamical systems. While images sequences may lie on a complex probability manifold in high-dimensional space, an invertible network coupled with a change of variables learns how to properly map that manifold of probability to the well-behaved conditional Gaussian created by a linear dynamic system. By formulating this with a single learning objective, we arrive at an elegant joint optimization problem. The primary advantage of this approach is that we avoid making any assumptions about the distribution of the input domain.

In future work we believe even better qualitative performance can be had as more becomes known about optimization and training of invertible networks.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[2] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[3] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[4] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[5] Chelsea Finn, Ian J. Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *CoRR*, abs/1605.07157, 2016. URL `http://arxiv.org/abs/1605.07157`.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

[7] Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[8] Moritz Hardt, Tengyu Ma, and Benjamin Recht. Gradient descent learns linear dynamical systems. *CoRR*, abs/1609.05191, 2016. URL `http://arxiv.org/abs/1609.05191`.

[9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[10] Andrzej Janczak. *Identification of nonlinear systems using neural networks and polynomial models: a block-oriented approach*, volume 310. Springer Science & Business Media, 2004.

[11] D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.

[12] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018.

[13] Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[14] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P. Xing. Dual motion GAN for future-flow embedded video prediction. *CoRR*, abs/1708.00284, 2017. URL `http://arxiv.org/abs/1708.00284`.

[15] Ziwei Liu, Raymond A. Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. *CoRR*, abs/1702.02463, 2017. URL `http://arxiv.org/abs/1702.02463`.

[16] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950, 2018.

[17] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2813–2821. IEEE, 2017.

[18] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[19] Viorica Patraucean, Ankur Handa, and Roberto Cipolla. Spatio-temporal video autoencoder with differentiable memory. *CoRR*, abs/1511.06309, 2015. URL `http://arxiv.org/abs/1511.06309`.

[20] Mikel D Rodriguez, Javed Ahmed, and Mubarak Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. 2008.

[21] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[22] Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, 1993.

[23] Khurram Soomro and Amir R Zamir. Action recognition in realistic sports videos. In *Computer vision in sports*, pages 181–208. Springer, 2014.

[24] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, abs/1502.04681, 2015. URL `http://arxiv.org/abs/1502.04681`.

[25] Ryan Szeto, Simon Stent, German Ros, and Jason J. Corso. A dataset to evaluate the representations learned by video prediction models. In *International Conference on Learning Representations (Workshop Track)*, Apr 2018.

[26] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[27] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *CoRR*, abs/1609.02612, 2016. URL `http://arxiv.org/abs/1609.02612`.

[28] Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3570–3578. JMLR. org, 2017.

[29] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[30] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[31] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Trans. Cir. and Sys. for Video Technol.*, 13(7):560–576, July 2003. ISSN 1051-8215. doi: 10.1109/TCSVT.2003.815165. URL `https://doi.org/10.1109/TCSVT.2003.815165`.

[32] Tianfan Xue, Jiajun Wu, Katherine L. Bouman, and William T. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. *CoRR*, abs/1607.02586, 2016. URL `http://arxiv.org/abs/1607.02586`.