TOWARDS NON-PARAMETRIC MODELS FOR CONFIDENCE AWARE VIDEO PREDICTION ON SMOOTH DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

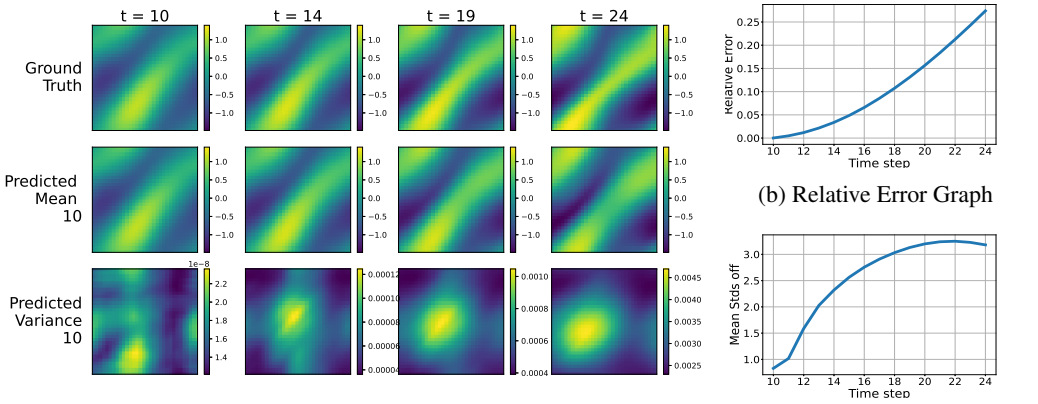
The ability to envision future states is crucial to informed decision making while interacting with dynamic environments. With vision providing an information rich sensing modality, the field of video prediction has garnered a lot of attention in pursuit of this ability. Current state of the art methods rely on neural network based models for prediction. Though often accurate, these methods require large amounts of training data that are often unavailable when encountering unknown environments. The predictive accuracy of such methods also breaks down without warning, when tested on data far outside their training distribution. This problem is exacerbated by the fact that these networks can be prohibitively expensive to update with recent data acquired online. To overcome these drawbacks we use non-parametric models to take a probabilistic approach to video prediction for problems with little training data. We generate probability distributions over sequentially predicted images and propagate our uncertainty through time to generate a confidence metric for our predictions. We use non-parametric Gaussian Process models for their data efficiency and ability to readily incorporate new training data online. To showcase our method we successfully predict future frames of a smooth fluid simulation environment.

1 INTRODUCTION

The ability to understand scenes and predict their future states is key to enabling smart decision making. Predictions on large scale phenomenon, like weather, enable life saving preventative measures such as evacuation warnings. As humans we utilize small scale predictions to inform our daily actions like navigating through a crowds, and catching falling objects. Perfectly predicting such diverse phenomena requires unique and complex models that rely on underlying state information. However, humans are often able to make decisions using imperfect models that are informed solely from visual input. We are able to readily interact with constantly changing, new environments, after only a small number of observations. In this paper we focus on the task of video prediction given a limited number of initial frames for both context and training.

Current state of the art approaches to visual prediction rely on neural networks Oprea et al. (2020). These methods use a vast amount of resources and data to train. Neural Networks are high order parametric models composed of up to millions of parameters, which are incrementally refined through evaluation against a loss metric on training data Goodfellow et al. (2016). The high parameter count increases model complexity, giving these networks the representational power to accurately capture a wide array of visual dynamics. Despite the predictive accuracy, highly parameterized models are challenged when faced with data outside their training distribution. Though this behavior is expected, the large amount of data and computational resources required to re-compute and refine millions of parameters, prevents these models from being easily updated when encountering new data outside the current training distribution. Often times these deep models often fail to recognize that their inputs are out of distribution. We overcome these limitations with our probabilistic approach to video prediction, using non-parametric Gaussian Process models.

Gaussian Processes (GPs) provide a non-parametric data driven approach to function approximation. They use a Gaussian assumption on the data points to extrapolate predictions from the training data.



(a) Forward Prediction: Image Results

(b) Relative Error Graph

(c) Mean Standard Deviations off

Figure 1: Forward Prediction Experiment: Our model, trained using frames $[z_0, \ldots, z_9]$, is used to predict frames $[z_{10}, \ldots, z_{24}]$ of a 2D Navier Stokes simulation. Fig. (1a) shows the ground truth and predicted mean and variance images. Fig. (1b) and Fig. (1c) show graphs of the relative error and mean standard deviations off between the predicted mean and ground truth images over the predicted time steps. These results showcase our model's ability to learn complex dynamics and predict compelling distributions on future frames that capture the ground truth.

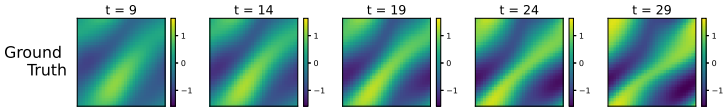
This enables high quality predictions near the training distribution, which can be readily expanded to include newly observed data points. The predictions from these models are probability distributions, which provides an interpretable metric for the model's confidence on its prediction. We use GPs as the core predictive component of our method, where we propagate our predicted distributions on images through time to generate a confidence metric along side our predictions. Visual prediction is a good candidate for a low data non-parametric approach as videos provide a very information rich sensing modality. Short sequences of video contain a large amount of semantic and dynamic information that can be used to understand unseen environments. In many videos, such as satellite weather patterns, fluid motion, and driving cars, the motion can be repetitive, both temporally across different frames, and spatially within the same frame. We take advantage of this repetitiveness to train a high quality model using only a few frames of data.

In this paper we contribute towards the use of non-parametric, data driven, methods for confidence aware short term video prediction. We restrict the scope of our video predictions to the problem of predicting 2D video sequences of smooth navier stokes dynamics and discuss the challenges in extending this method to general non-parametric video prediction. We also make key assumptions that the video frames capture a majority of the state information pertinent to predicting future states.

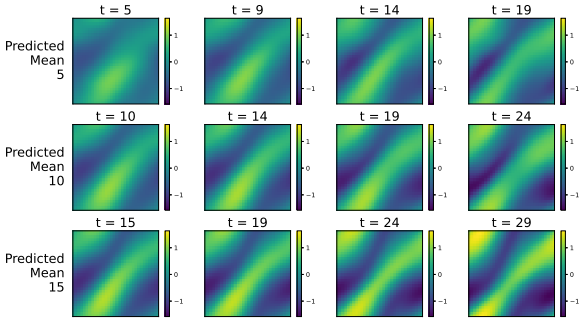
2 Related Works

The problem of video prediction is presented in several different settings, with methods being trained and evaluated on a variety of different datasets. Real world video sequences capture the motion of discrete objects. Toy problems, such as in the bouncing ball dataset (Sutskever et al., 2008) and moving MNIST dataset (Srivastava et al., 2016) have been used to evaluate a method's ability to track such distinct passively moving objects in video. The Robot Pushing Dataset (Finn et al., 2016) provides a dataset for problems involving robot interaction in video prediction and robot control. There is also a large body of work on predicting motion of autonomous agents. This is prominent in research on self driving cars where cars and pedestrians act as autonomous agents. Commonly used datasets for this problem include the Kitti dataset (Geiger et al., 2013), CamVid Dataset (Brostow et al., 2008) and Caltech Pedestrian Dataset (Dollar et al., 2009).

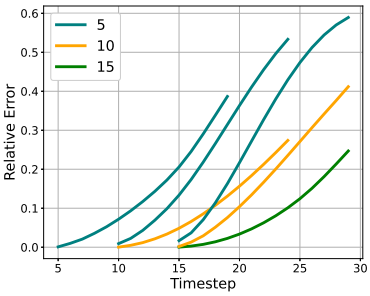
To solve most of the problem settings above, researchers create predictive models using neural networks. The success of neural networks on image based applications has lead to their widespread adoption in fields such as video prediction. Most video prediction works build off of a few baseline neural network architectures: convolutional, recurrent and generative models. Convolutional Neu-



(a) Ground Truth Image Frames



(b) Sequence Prediction Mean Images



(c) Sequence Prediction Errors

Figure 2: Sequential Prediction Experiment: In this experiment our model, trained using frames $[z_0, \ldots, z_{t_0-1}]$, is used to predict frames $[z_{t_0}, \ldots, z_{t_0+15}]$ of a 2D Navier Stokes simulation. We show the results of such predictions for $t = 5, 10, 15$ along the rows of 2b, respectively. Fig. (2a) displays ground truth images. Fig. (2b) shows all the predicted mean images. Fig. (2c) displays a graph of the relative error between the predicted mean and the ground truth images. In 2c we show additional error results in which we start predictive rollouts with the models trained with $5, 10$ images from later time steps. This provides a fair comparison to analyze the effects of adding data. The decrease in error with each new model shows the value of incorporating recent data into our model.

ral Networks (CNNs) that rely on 2D convolutional kernels enabled a breakthrough on challenging problems in the image domain (O'Shea & Nash, 2015). Works including Wang et al. (2019), Aigner & Korner (2018), Vondrick et al. (2016), add a third dimension to their convolutional kernels to incorporate information across time. Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986) and LSTMS (Hochreiter & Schmidhuber, 1997) provide a more principled way of incorporating temporal dependence into network architectures. Explicitly designed to handle sequential data, these methods have been widely embraced across several video prediction works (Chen et al., 2017), (Wichers et al., 2018), and (Walker et al., 2017). The above approaches model the output images conditioned on the given input frames. Generative models, like Variational Auto Encoders (VAEs) (Kingma & Welling, 2014) and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), seek to model the joint distribution between the input and output frames. VAEs utilize embeddings into an underlying latent space. Several methods attempt to utilize this latent embedding for video prediction, and to capture the inherent uncertainty present in the task of future prediction (Bhattacharyya et al., 2019), (Babaeizadeh et al., 2018), (Denton & Fergus, 2018), and (Fragkiadaki et al., 2017). This uncertainty often creates blurry images (Oprea et al., 2020). Several works including Mathieu et al. (2016), Wichers et al. (2018), and Villegas et al. (2018) use GANs and take advantage of their adversarial training methodology to improve the sharpness and realism of their predicted frames.

These baseline techniques are used in a variety of approaches to video prediction. Srivastava et al. (2016), Yu et al. (2020) and Byeon et al. (2018) directly synthesize pixels in the future frames. This methodology often relies on capturing pixel patterns over understanding underlying states and dynamics. Reda et al. (2021), Michalski et al. (2014) and Klein et al. (2015) synthesize output pixels as explicit functional transforms of the input pixels. We take inspiration from this approach of direct pixel synthesis and convolutional models in designing our method. Another approach to prediction involves disentangling and separately predicting components of the video, such as high level motion and lower level details. Denton & Birodkar (2017), Gao et al. (2019), and Hsieh et al. (2018) adopt this methodology. Some works choose to focus on prediction in a high level feature space, rather than render every pixel. Walker et al. (2017), and Tang et al. (2019) predict in the domain of low dimensional human poses, while Minderer et al. (2020) predicts in a general keypoint based

representation. These high level representations and other features, such as control values, have been incorporated into inputs for better predictions (Oh et al., 2015), (Finn et al., 2016).

There is also a large body of work on predicting and simulating fluids. The motion of incompressible fluids is governed by the Navier Stokes equations, a set of partial differential equations (PDEs). Traditionally, complex PDEs are solved using Finite Element Methods (FEMs) and Finite Difference Methods (FDMs). These methods work by approximating solutions on a discretized grid. Modern techniques such as Raissi et al. (2019), Jiang et al. (2020), Greenfeld et al. (2019) and Li et al. (2021) use neural networks to learn to solve complex PDEs directly from data. These works build off of the baseline architectures discussed above, with modifications to make them better suited for the task at hand. These works can give valuable insights into certain video prediction tasks like predicting satellite weather patterns. Using neural networks to predict weather from satellite imagery with other relevant inputs, has been explored by Sønderby et al. (2020) and Klein et al. (2015).

Neural Networks require a lot of data and computational resources to train. Li et al. (2021) utilizes several thousand training images to train a predictive model for its easiest simulation parameterization, requiring far more data to capture more complex dynamics. Certain architectures such as GANs can be quite unstable and difficult to train (Oprea et al., 2020). Network architectures must be hand tailored to fit the specific problem at hand, and often fail without warning on data far from the training distribution. These drawbacks prevent neural networks from being used in problems where data is scarce. We turn to non-parametric methods, in particular Gaussian Processes (GPs), for video prediction with little data. GPs have been used to estimate highly nonlinear functions with probabalistic confidence using little training data. They have been widely used in Robotics to learn inverse kinematic mappings (Wilcox & Yip, 2020), model unknown safety functions (Turchetta et al., 2016), (Turchetta et al., 2019) and to model and propagate uncertainty through the dynamics of a robotic system (Deisenroth & Rasmussen, 2011). In this work we will show how this non-parametric method can be used for short term video prediction in smooth dynamic fluid environments.

3 Background: Gaussian Processes

The core predictive component of our method uses a Single Output Gaussian Process Regression Model. A GP is used to model a function f, using training data $(X, f(X))$. $X = [x_0, x_1, \ldots, x_{n-1}] \in \mathbb{R}^{n \times D}$ are all the training inputs and $f(x) = [f(x_0), \ldots, f(x_{n-1})] \in \mathbb{R}^{n \times 1}$ are the training outputs. Given test inputs $X^{'} \in \mathbb{R}^{m \times D}$ we want to find outputs $f(X^{'})$. Let $X_A \in \mathbb{R}^{(m+n) \times D}$ refer to all the train and test inputs and $f(X_A)$ be the corresponding outputs. A GP relies on the assumption that all the outputs are characterized by a multivariate gaussian distribution $f(X_A) \sim \mathcal{N}(\mu(X_A), \Sigma_{X_A X_A})$. We assume the mean $\mu(X_A) = 0$, and the covariance matrix is characterized by a kernel function $k(x, y)$ such that $\Sigma_{X_A, X_A}[u, v] = k(X_A[u], X_A[v])$. To solve for the distribution of the test outputs $p(f(X^{'})) \sim \mathcal{N}(\mu(X^{'}), \Sigma_{X^{'}X^{'}})$ we use the marginal likelihood of a multivariate gaussian $p(f(X^{'}|X, f(X), X^{'}))$ to get:

$$\mu(X^{'}) = k(X^{'}, X)[k(X, X) + \sigma_n^2 I]^{-1} f(X) \tag{1}$$

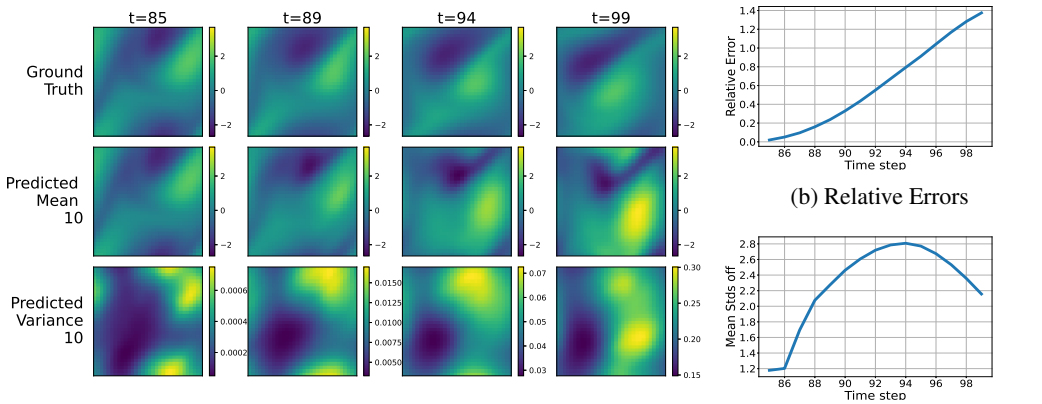
$$\Sigma_{X^{'}X^{'}} = k(X^{'}, X^{'}) - k(X^{'}, X)[k(X, X) + \sigma_n^2 I]^{-1} k(X, X^{'}) \tag{2}$$

$k(X, X)[u, v] = k(X[u], X[v])$, $k(X, X) \in \mathbb{R}^{n \times n}$, $k(X^{'}, X)[u, v] = k(X^{'}[u], X[v])$, $k(X^{'}, X) = k(X, X^{'})^T \in \mathbb{R}^{m \times n}$, σ_n^2 is the noise variance, $I \in \mathbb{R}^{n \times n}$ is the identity matrix. To train the GP Regression model we optimize the noise variance, σ_n, and kernel parameters to maximize the log likelihood of the training data.

In this paper we utilize the Radial Basis Function (RBF) kernel for all our methods:

$$k(x, y) = \alpha^2 \exp(-\frac{(x - y)^T \Lambda^{-1} (x - y)}{2}) \tag{3}$$

The kernel parameters, α and Λ, are optimized during training. To predict outputs of dimension $O > 1$, we train a separate GP model for each output dimension $a \in [0, O - 1]$. Each model has its own kernel $k_a(.,.)$ and is trained by optimizing its noise and kernel parameters. $\sigma_{n,a}, \alpha_a$ and Λ_a.



(a) Out distribution: Images

(b) Relative Errors

(c) Mean Standard Deviations off

Figure 3: Out of distribution variance experiment: Our model is trained with frames $[z_0, \ldots, z_9]$ of a 2d Navier Stokes simulation. This model is used to predict 15 frames starting from the input images $[z_{82}, z_{83}, z_{84}]$, which generate inputs far from the model's training distribution. The increase in the magnitude of the predicted variance, compared to the 'In distribution' predictions shown in fig. 1a, as well as low mean standard deviations off over the predicted time steps demonstrates our model's understanding of its failure modes. The model adjusts its variance, recognizes the break down in the predictive accuracy of its mean images to capture the ground truth within its predicted distributions.

4 METHOD

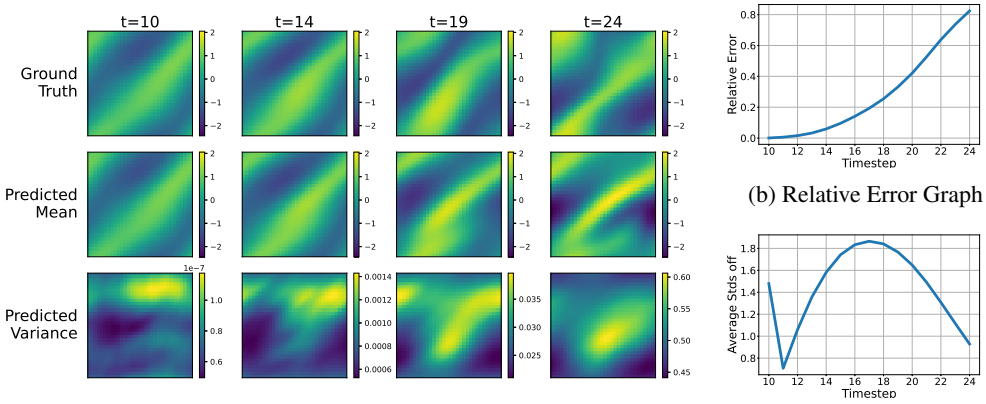
4.1 PROBLEM STATEMENT AND PREDICTION FRAMEWORK

A video is defined as a sequence of images or frames $[z_0, z_1 \ldots z_{t_0-1}, z_{t_0}, z_{t_0+1} \ldots z_{t-1}, z_t \ldots]$. Here $z_i \in \mathbb{R}^{Q \times Q}$ denotes the i-th frame in the sequence. Given initial training frames $[z_0, \ldots z_{t_0-1}]$, we wish to predict frames $[z_{t_0}, \ldots z_t, \ldots]$. We restrict the problem to the regime of new, unseen environments by limiting frames $[z_0 \ldots z_{t_0-1}]$, to be the only training data. As additional frames $[z_{t_0}, \ldots z_{t'}]$ become available, they may be incorporated into the model's training data to improve the accuracy of future predictions.

In our method, we train a model to use re-occurring motion patterns to understand scene dynamics for video prediction. We have our model learn and predict on square image patches of dimension (p, p). This smaller patch scale enables us to better utilize our limited training data and extract smaller repeating patterns that are more likely to re-occur across space and time. Our method predicts one frame at a time given the 3 most recent, seen and predicted, frames. We use 3 frame inputs to capture second order dynamics. We first convert the inputted image frames into patches and then suitable inputs for our GP Regression models. The models output distributions on pixels in future images, that are combined to form a random variable image. This predicted image is incorporated into the next set of inputs and the process is repeated. As such our models must handle a combination of random and known inputs, while propagating probability distributions through time.

4.2 TRAINING

To construct our model, we begin by creating a training data set from frames $[z_0, \ldots, z_{t_0-1}]$. We divide the images into sets of 4 sequential images $[z_i, z_{i+1}, z_{i+2}, z_{i+3}], i \in [0, t_0 - 4]$. To create a datapoint we take p dimensional patches corresponding to the same pixel locations from each image. $z_i[k : k + p, l + l + p] \in \mathbb{R}^{p \times p}$ denotes a p by p patch in image z_i starting at pixel (k, l). A training input, $x_j \in \mathbb{R}^{3p^2}$, is created by flattening and concatenating the patches from the first 3 images. The corresponding training output, $f(x_j) \in \mathbb{R}^{(p-b)^2}$, is created by flattening the corresponding patch from the 4th image z_{i+3}, cropped with a patch boundary term b: $z_{i+3}[k + b : k + p - b, l + b : l + p - b] \in \mathbb{R}^{(p-b)\times(p-b)}$. When $b > 0$, our model does not predict the outer edges of the patch, where predictions may suffer due to contributions from the scene outside our input. Within each set of 4



(a) High Error Experiment: Image Results

(b) Relative Error Graph

(c) Mean Standard Deviations off

Figure 4: High Error Experiment: This figure shows an instance where our predictions have a higher relative error. Our model is trained using 10 frames $[z_0, \ldots, z_9]$ is used to forward predict the next 15 frames $[z_{10}, \ldots, z_{24}]$ of a 2d Navier Stokes simulation. Fig. 4a shows the ground truth and predicted mean and variance images Fig. (4b) and Fig. (4c) show graphs of the relative error and mean standard deviations off between the predicted mean and ground truth images, respectively. These results demonstrate how our model increases its variance to decrease its predictive confidence when it incurs high relative error, allowing it to keep the mean standard deviations off low.

sequential images, we sample data points with a stride of s pixels in the x and y dimensions. In this paper, we utilize a 'wrapping' approach to handle patches that extend beyond the edge of an image. This approach assumes the frame, z_i with g rows and f columns, captures a periodic environment such that $z[g+i, f+j] = z[i-1, j-1]$ and $z[-i, -j] = z[g-(i+1), f-(j+1)]$. Approaches like zero padding frames to indicate boundaries or skipping incomplete patches are possible but not further explored in this paper. We repeat this sampling procedure for every set of sequential images to create the training data set with n data points: $(X, f(X)) = (x_j, f(x_j)), j \in [0, n-1]$. $X \in \mathbb{R}^{n \times 3p^2}$ and $f(X) \in \mathbb{R}^{n \times (p-b)^2}$. This process is shown graphically in Figure 7 in A.6

We create a GP Regression model for every output dimension $O = (p-b)^2$. Each model is trained by optimizing its noise, $\sigma_{n,a}$, and kernel parameters, α_a and Λ_a in $k_a(.,.)$, to maximize the log likelihood of the training data for output dimension $a \in [0, (p-b)^2 - 1]$. To predict future images, each GP model outputs a mean and variance corresponding to a single pixel in the output patch. The predicted image z_i, is represented with a mean and variance image pair (M_i, V_i). Each pixel in M_i and V_i corresponds to the mean and variance of the predicted random variable for that pixel location, respectively. We ensure that every future image pixel is predicted exactly once.

4.3 PREDICTION

Once trained, we can use our model to rollout predictions for any T time steps into the future, starting from 3 known, consecutive input images $[z_i, z_{i+1}, z_{i+3}]$. We use a recursive method to predict multiple frames into the future. Our model takes the 3 most recent seen or predicted frames and uses them to predict one frame into the future, which is represented as a random variable. We incorporate this predicted random variable as the latest image in the 3 frame inputs to predict the next time step. This process is repeated to predict the desired T time steps into the future. We start discussing our predictions in the context of predicting the fourth time step and onwards. Starting at the fourth prediction all the input images utilized are random variables previously outputted by our model. The first three predictions incorporate known, observed input images in addition to the predicted random input images. These initializing predictions will be discussed as a special case of the more general prediction from all random variable input images.

We discuss the general method of predicting z_{i+3} from input images $[z_i, z_{i+1}, z_{i+2}]$, that are all random variables outputted by our model. To predict z_{i+3}, we first create a set of m test inputs $x_j^{'}$, $j \in [0, m-1]$. Each test input is a multivariate gaussian random variable composed of $3p^2$

independent gaussian random variables sampled from the input images. Since the input images are predicted random variables, our test inputs are created by sampling their corresponding mean and variance images: $[(M_i, V_i), (M_{i+1}, V_{i+1}), (M_{i+2}, V_{i+2})]$. We make simplifying assumptions that the predictions of each GP model as well as the outputs of the same model on different inputs are independent. Without assuming independence, we would have the computationally intractable task of tracking the covariance values across all pixels. As a result, the predicted images and their subpatches can be flattened, concatenated and represented as one multivariate gaussian random variable. We use the patch based sampling method described in Section 4.2, separately, on the sets of consecutive mean and variance images to generate the mean and variance test input vectors, $x^{'}_{j,\mu} \in \mathbb{R}^{3p^2}$ and $x^{'}_{j,\sigma} \in \mathbb{R}^{3p^2}$ respectively. These vectors specify the multivariate gaussian distribution of the input $x^{'}_j \sim \mathcal{N}(x^{'}_{j,\mu}, \Sigma_{x^{'}_{j,\sigma}})$. To construct $\Sigma_{x^{'}_{j,\sigma}}$, we use our simplifying independence assumptions such that the input covariance is a diagonal matrix with the vector of variances, $x^{'}_{j,\sigma}$, along the diagonal. We adjust our sampling stride to generate an input to predict every pixel in the future image.

We discuss our method in the context of predicting a single output dimension $a \in [0, (p-b)^2 - 1]$ from a single input $x^{'}_j$. Our model output is the predicted random variable $f(x^{'}_j)[a]$. As in standard Gaussian Process Regression, we are solving for the distribution of $p(f(x^{'}_j)[a])$. We solve for $p(f(x^{'}_j)[a])$ by marginalizing $p(f(x^{'}_j)[a]|X, f(X), x^{'}_j)$ over the input images.

$$p(f(x^{'}_j)[a]) = \int_{-\infty}^{\infty} p(f(x^{'}_j)[a]|x^{'}_j, X, f(X)[:, a])p(x^{'}_j)dx^{'}_j \tag{4}$$

Solving this integral is analytically intractable. We approximate the posterior distribution from 4 to be Gaussian. Having the outputs form a multivariate gaussian, like the inputs, enables the predictions to be recursive. To solve for $p(f(x^{'}_j)[a])$ we take advantage of this assumption and use moment matching in a method akin to Deisenroth & Rasmussen (2011). However, our method is distinguished from Deisenroth & Rasmussen (2011) in its use of multiple past states as inputs, prediction on images, and incorporation of known states in its inputs. Moment matching enables us to directly solve for the mean $\mu(x^{'}_j)[a]$ and variance $\Sigma(x^{'}_j)[a, a]$ of the outputted Gaussian distribution. This gives us the following formula to predict the mean of an output pixel from all random inputs:

$$\begin{gathered} \mu(x^{'}_j)[a] = d_a^T \beta_a \\ d_a[i] = \alpha_a^2 \cdot (|\Sigma_{x^{'}_{j,\sigma}} \Lambda_a^{-1} + I|)^{-\frac{1}{2}} \cdot e^{-\frac{1}{2} v_i^T (\Sigma_{x^{'}_{j,\sigma}} + \Lambda_a)^{-1} v_i} \\ \beta_a = [k_a(X, X) + \sigma_n^2 I]^{-1} f(X)[:, a] \end{gathered} \tag{5}$$

Here $d_a \in \mathbb{R}^n$, and $v_i = x_i - x^{'}_{i,\mu}$. To predict the variance from all random inputs we use:

$$\Sigma(x^{'}_j)[a, a] = \alpha_a^2 - trace\left((k_a(X, X) + \sigma_{n,a}^2 I)^{-1} Q_{aa}\right) + \beta_a^T Q_{aa} \beta_a - \mu(x^{'}_j)[a] \tag{6}$$

$$Q_{aa}[i, k] = k_a(x_i, x^{'}_j) k_a(x_k, x^{'}_j) \cdot |R|^{-\frac{1}{2}} \cdot e^{\frac{1}{2} z_{ik}^T R^{-1} \Sigma_{x^{'}_{j,\sigma}} z_{ik}} \tag{7}$$

Here $Q_{aa} \in \mathbb{R}^{n \times n}$, $R = \Sigma_{x^{'}_{j,\sigma}} 2\Lambda_a^{-1} + I \in \mathbb{R}^{n \times n}$ and $z_{ik} = \Lambda_a^{-1} v_i + \Lambda_a^{-1} v_k$. A walkthrough derivation of these equations, using moment matching, is discussed in the appendix A.2. These equations are used on all the test inputs to predict the mean and variance for every pixel in z_{i+3}. The final predicted image z_{i+3} is stored as a mean, variance image tuple: (M_{i+3}, V_{i+3}). To continue the predictive rollout we incorporate this latest prediction into a new set of input images $[z_{i+1}, z_{i+2}, z_{i+3}]$ to predict z_{i+4}. The predicted mean images $[M_i, \ldots, M_{i+3}, \ldots]$ act as our predicted estimates of the ground truth, while the predicted variance images $[V_i, \ldots, V_{i+3}, \ldots]$ act as a confidence measure on our prediction. This is explored more in Section 5

In the first three predictions some or all of the input images are known, observed images. These predictions are initializing steps to begin the predictive rollout. The initializing predictions are special cases of the general prediction formulation with all random variable inputs. In this case we still treat all components of our input as random variables. We no longer treat the whole input as a single multivariate gaussian. We disentangle the predictive method into parts that solely interact with the input dimensions contributed from the observed images, and those contributed from the predicted random variable images. We treat the random variable component of the input as a multivariate

gaussian and use a delta function at the observed values as the joint probability distribution for the known component.We use these representations to re-derive the moment matching formulas. We walk through this method in A.1 of the Appendix. Figure 8 is an overview of the whole method.

5 EXPERIMENTS AND RESULTS

In this work we test our methods by predicting the vorticity of an incompressible fluid on a unit torus environment. Our data is computed using two dimensional Navier Stokes Equations. We generate our data with traditional PDE solvers using the code and approach detailed in Li et al. (2021). These fluid simulation lack discrete objects and generate frames whose pixels change smoothly across both space and time. This smooth environment is well suited to our method's choice of the RBF Kernel. The dynamics of the toroidal environment wrap around the edges of the video frame. As a result, we utilize the "Wrapping" approach when handling edge patch creation. Each pixel is represented by a single float whose value is centered around 0. As a result, we can directly predict future pixels using a gaussian process model that assumes its outputs have zero mean. In the generated video sequence, frame $z_t \in \mathbb{R}^{Q\times Q}$ represents the vorticity of the fluid at time-step t. We fix the frame resolution at $Q = 32$, and simulate with a viscosity of $1e-3$ for all our simulated data. For all our experiments we train our model to take square input patches of dimension $p = 15$ and output $(1, 1)$ patches using an output patch border $b = 7$. These parameters result in our model using a single GP Regression model to predict a single output dimension. To generate the data, for all experiments, we use a training stride of $s = 2$ pixels in both the x and y dimensions. A test stride of $s = 1$ is used to generate test inputs to predict a distribution for every pixel in the future image. We predict 15 future frames for every experiment. The training images and the set of input images used to start the predictive rollouts are specified for each experiment.

We introduce two metrics: Relative error (Li et al., 2021) ($RE(z, \tilde{z})$) and Mean Standard Deviations off ($StdE(z, \tilde{z_\mu}, \tilde{z_\sigma})$) to analyze the performance of our model.

$$RE(z, \tilde{z}) = ||z - \tilde{z}||_2 \cdot ||z||_2^{-1} \tag{8}$$

In (8) $x \in \mathbb{R}^{Q\times Q}$ denotes the ground truth image, $\tilde{x} \in \mathbb{R}^{Q\times Q}$ denotes the predicted image and $||.||_2$ is the 2 norm of a matrix. This normalizes the error with the magnitude of the original image

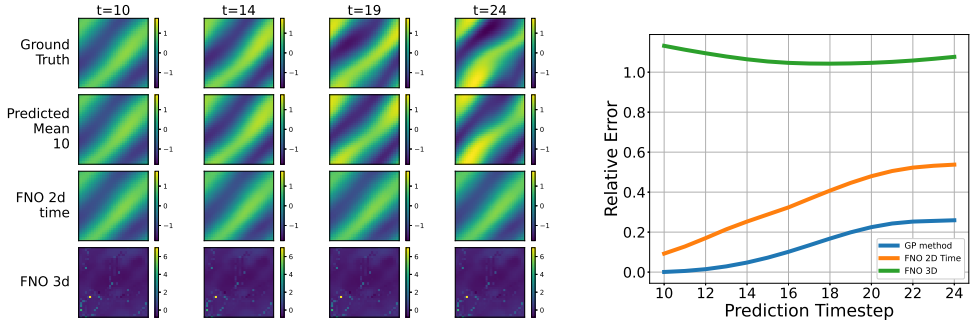
$$StdE(z, \tilde{z_\mu}, \tilde{z_\sigma}) = \frac{1}{Q^2}\sum_{i=0}^{Q-1}\sum_{j=0}^{Q-1}\left(\frac{|z[i,j] - \tilde{z_\mu}[i,j]|}{\sqrt{\tilde{z_\sigma}[i,j]}}\right) \tag{9}$$

$z \in \mathbb{R}^{Q\times Q}$ is the ground truth image, $\tilde{z_{mu}}, \tilde{z_\sigma} \in \mathbb{R}^{Q\times Q}$ is the predicted mean and variance images outputted by our model, and $|.|$ is the absolute value function. This metric returns the average absolute standard deviations between our model's predicted mean image and the ground truth. This metric is inversely proportional to the how closely the predicted distribution captures the truth.

Forward Prediction Experiment: In this experiment, we train our model using the first $t_0 = 10$ frames of a video $[z_0, \dots z_9]$. This model is used to predict the next 15 frames $[z_{10}, \dots, z_{24}]$, from input images $[z_7, z_8, z_9]$. The results of this experiment are shown in Figure 1. Our model's predicted mean images track the complex dynamics of the ground truth with very little training data. The low magnitude of the variance images in Figure 1a indicates our model's confidence in its predictions. This confidence is well founded given the low relative error in Figure 1b and low $StdE$ in Figure 1c which illustrate our ability to predict distributions that capture the ground truth.

Sequential Prediction Experiment: Having established our model's predictive capabilities, we examine the benefits of incorporating recent data into our model. In this experiment we incrementally train models with the first $5, 10$ and 15 images of a video sequences. Each of these models is used to predict 15 frames into the future, starting from their last training image. In Figure 2 we can visually see the improvement in prediction accuracy as we update our models. In the relative error graph (Figure 2c) we also show the results of starting predictive rollouts for the lower data models, trained with 5 and 10 images, later in the sequence. This provides a fair evaluation to compare the impact of adding recent data, by mitigating the added error compounded as a result of the predictive rollouts. The graph shows a large improvement in accuracy as we incorporate data into our model.

Variance Experiments: The variance outputted by our model acts as confidence measure around the value of the corresponding predicted pixel mean. This variance is based on a kernel based



(a) Predictive Comparison Images

(b) Predictive Comparison Errors

Figure 5: Predictive Comparison Experiment: This figure compares the predictions on Navier Stokes simulations from our method to the neural network based methods, FNO-2d-time and FNO-3d, discussed in Li et al. (2021) trained on similarly low data. Details are discussed in section A.4.

metric evaluating the similarity between the model's training data and the test input. To understand the variance output we look at a predictive rollout started with inputs far from our model's training distribution. We use the same model trained in the 'Forward Prediction' experiment to predict frames $[z_{85}, \ldots, z_{99}]$ from input images $[z_{82}, z_{83}, z_{84}]$. Since our environment changes slowly and smoothly over time, a large temporal gap between frames increases distance with respect to our kernel based similarity measure. Looking at the results in Figure 3 we notice that despite the high relative error, in Figure 3b, the mean standard deviations off ($StdE$) in Figure 3c remain low. The model recognizes that its inputs are too far from its training distribution to make a confident prediction and outputs higher variance values. This increase in variance magnitudes is very noticeable when contrasted with the variances in Figure 1a, where the inputs start close to the training distribution. We can better understand the variance by analyzing our model's failure modes. Figure 4 shows the results of a 'Forward Prediction' experiment on a sequence where the predicted mean images had a higher error. Recognizing the decrease in predictive accuracy, the model increases its variance to track the increase in error, keeping the $StdE$ low. The initial jump in this and other mean standard deviations off graphs indicates an initial overconfidence that the model quickly corrects for. Towards the end of longer predictive steps the $StdE$ decreases despite an increasing relative error, as the variance outputs balloon to reflect the model's decreased confidence. These results demonstrate our model's ability to understand its own failure modes with a interpretable confidence metric.

Predictive Comparison Experiment: We compare our model's performance to the FNO-2d-time and FNO-3d neural network methods presented in Li et al. (2021). All methods are trained using a similarly low number of training images. This experiment serves solely as a visual comparison. The use cases of the contrasted models are widely different, with Networks not being suited to little training data. The results in Figure 5 show our model outperforming the neural network methods. Both networks fail to learn the complex dynamics from such little data, outputting either identical frames or noise. Details on the experiment are in section A.4 of the appendix.

6 Conclusion and Discussion

In this paper we provided a novel method using non-parametric GP models to predict probability distributions over future video frames in smooth dynamic environments with limited training data. Our use of the RBF kernel and direct prediction of future pixels caters towards the smoothness of the zero mean images of our test environment. We can expand our prediction equations to other kernel equations and predict zero mean image differences to generalize our method beyond smooth dynamic environments. Another future direction is to find methods to efficiently track covariances between pixels, to increase the accuracy of our distributions for longer term predictions. The problem of accurately learning to accurately predict in any environment with little data is very difficult. We seek to expand upon our work by combining our method with parametric learned approaches. Such a combination can leverage the predictive accuracy of neural networks and switch to non-parametric approaches when encountered with new environments far from the network training distribution.

REFERENCES

Sandra Aigner and Marco Korner. Futuregan: Anticipating the future frames of video sequences using spatio-temporal 3d convolutions in progressively growing gans. *arXiv: Computer Vision and Pattern Recognition*, 2018.

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Royhttps://arxiv.org/help/api/index H. Campbell, and Sergey Levine. Stochastic variational video prediction, 2018.

Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. Bayesian prediction of future street scenes using synthetic likelihoods, 2019.

Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In David Forsyth, Philip Torr, and Andrew Zisserman (eds.), *Computer Vision – ECCV 2008*, pp. 44–57, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-88682-2.

Wonmin Byeon, Qin Wang, Rupesh Kumar Srivastava, and Petros Koumoutsakos. Contextvp: Fully context-aware video prediction, 2018.

Xiongtao Chen, Wenmin Wang, Jinzhuo Wang, and Weimian Li. Learning object-centric transformation for video prediction. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, pp. 1503–1512, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349062. doi: 10.1145/3123266.3123349. URL https://doi.org/10.1145/3123266.3123349.

Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 465–472, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Emily Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video, 2017.

Emily Denton and Rob Fergus. Stochastic video generation with a learned prior, 2018.

Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 304–311, 2009. doi: 10.1109/CVPR.2009.5206631.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction, 2016.

Katerina Fragkiadaki, Jonathan Huang, Alex Alemi, Sudheendra Vijayanarasimhan, Susanna Ricco, and Rahul Sukthankar. Motion prediction under multimodality with conditional stochastic networks, 2017.

Hang Gao, Huazhe Xu, Qi-Zhi Cai, Ruth Wang, Fisher Yu, and Trevor Darrell. Disentangling propagation and generation for video prediction, 2019.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

Daniel Greenfeld, Meirav Galun, Ron Kimmel, Irad Yavneh, and Ronen Basri. Learning to optimize multigrid pde solvers, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction, 2018.

Chiyu Max Jiang, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, Karthik Kashinath, Mustafa Mustafa, Hamdi A. Tchelepi, Philip Marcus, Prabhat, and Anima Anandkumar. Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

Benjamin Klein, Lior Wolf, and Yehuda Afek. A dynamic convolutional layer for short rangeweather prediction. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4840–4848, 2015. doi: 10.1109/CVPR.2015.7299117.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error, 2016.

Vincent Michalski, Roland Memisevic, and Kishore Konda. Modeling deep temporal dependencies with recurrent grammar cells"". In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/cd89fef7ffdd490db800357f47722b20-Paper.pdf.

Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos, 2020.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games, 2015.

Sergiu Oprea, Pablo Martinez-Gonzalez, Alberto Garcia-Garcia, John Alejandro Castro-Vargas, Sergio Orts-Escolano, Jose Garcia-Rodriguez, and Antonis Argyros. A review on deep learning techniques for video prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020. ISSN 1939-3539. doi: 10.1109/tpami.2020.3045007. URL http://dx.doi.org/10.1109/TPAMI.2020.3045007.

Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Fitsum A. Reda, Guilin Liu, Kevin J. Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao, and Bryan Catanzaro. Sdcnet: Video prediction using spatially-displaced convolution, 2021.

D. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms, 2016.

Ilya Sutskever, Geoffrey Hinton, and Graham Taylor. The recurrent temporal restricted boltzmann machine. volume 20, pp. 1601–1608, 01 2008.

Casper Kaae Sønderby, Lasse Espeholt, Jonathan Heek, Mostafa Dehghani, Avital Oliver, Tim Salimans, Shreya Agrawal, Jason Hickey, and Nal Kalchbrenner. Metnet: A neural weather model for precipitation forecasting, 2020.

Jilin Tang, Haoji Hu, Qiang Zhou, Hangguan Shan, Chuan Tian, and Tony Q. S. Quek. Pose guided global and local gan for appearance preserving human video prediction. In *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 614–618, 2019. doi: 10.1109/ICIP.2019.8803792.

Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration in finite markov decision processes with gaussian processes, 2016.

Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration for interactive machine learning, 2019.

Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction, 2018.

Susana Vinga. Convolution integrals of normal distribution functions. 01 2004.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics, 2016.

Jacob Walker, Kenneth Marino, Abhinav Gupta, and Martial Hebert. The pose knows: Video forecasting by generating pose futures, 2017.

Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li. Fei-Fei. Eidetic 3d lstm: A model for video prediction and beyond. In *ICLR*, 2019.

Nevan Wichers, Ruben Villegas, Dumitru Erhan, and Honglak Lee. Hierarchical long-term video prediction without supervision, 2018.

Brian Wilcox and Michael C. Yip. Solar-gp: Sparse online locally adaptive regression using gaussian processes for bayesian robot model learning and control. *IEEE Robotics and Automation Letters*, 5(2):2832–2839, 2020. doi: 10.1109/LRA.2020.2974432.

Wei Yu, Yichao Lu, Steve Easterbrook, and Sanja Fidler. Efficient and information-preserving future frame prediction and beyond. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1eY_pVYvB`.

A Appendix

A.1 Predictions: The first 3 Predictions in a sequence of predictions

The first three predictions of a rollout involve predicting from a combination of known and predicted random variable sets of input images. We treat these predictions as special cases of the more general case presented in Section 4.3. Each input $x_j^{'}$ is composed of $3p^2$ independent random variables, with p^2 random variables contributed from each input image. We separate the input $x_j^{'}$ along the dimensions of the input that are known and random, to handle the different components separately. $x_{j,Rn}^{'}$ and $x_{j,Kn}^{'}$ are random variables that denote the random and known components of the input respectively. X_{Rn}, X_{Kn} and $x_{i,Rn}, x_{i,Kn}$ reference the corresponding known and random dimensions in all the training inputs and a single training input respectively.

The kernel functions k_a and the probability distribution over the input $p(x_j^{'})$ are the only forms of interaction with the inputs while predicting the output distribution $p(f(x_j^{'})[a])$. The structure of these functions and our independence assumptions allow us to cleanly split up our inputs into known and random components. We use the Radial Basis Function (eq. 3) as our kernel function $k_a(x, y)$. In this function the inputs interact with one another along the same dimension, allowing

us to re-write the kernel as:

$$k_a(x,y) = \alpha_a^2 \exp\left(-\frac{(x-y)^T\Lambda_a^{-1}(x-y)}{2})\right) = \alpha_{a,Kn}^2 \cdot \exp\left(-\frac{(x_{Kn}-y_{Kn})^T\Lambda_{a,Kn}^{-1}(x_{Kn}-y_{Kn})}{2}\right) \cdot \alpha_{a,Rn}^2 \exp\left(-\frac{(x_{Rn}-y_{Rn})^T\Lambda_{a,Rn}^{-1}(x_{Rn}-y_{Rn})}{2}\right) = k_{a,Kn}(x_{Kn},y_{Kn}) \cdot k_{a,Rn}(x_{Rn},y_{Rn}) \tag{10}$$

The subscripts on the inputs correspond to their known and random dimensions. $k_{a,Rn}$ and $k_{a,Kn}$ are the kernel functions that act on the known and random dimensions respectively. Each are parameterized by their own set of kernel parameters: $\alpha_{a,Kn}, \Lambda_{a,Kn}$ and $\alpha_{a,Rn}, \Lambda_{a,Rn}$ respectively. $\alpha_{a,Rn} \cdot \alpha_{a,Kn} = \alpha_a$. $\Lambda_{a,Kn}$ and $\Lambda_{a,Rn}$ are block diagonal matrices sampled from Λ_a, a large diagonal matrix, along the known and random dimensions. Using our assumption that all predicted pixels within and across images are independent, we separate the probability distribution:

$$p(x_j^{'}) = p(x_{j,Kn}^{'}) \cdot p(x_{Rn}^{'}) \tag{11}$$

$p(x_{j,Rn}^{'})$ is a multivariate gaussian distribution of the random components of the input and $p(x_{Kn}^{'})$ denotes the joint distribution of all the known input pixels.

We construct our inputs according to this split input structure. The random component of each input is a multivariate gaussian distribution, $x_{j,Rn}^{'} \sim \mathcal{N}(x_{j,\mu}^{'}, \Sigma_{x_{j,\sigma}^{'}})$, specified by its mean and covariance matrix. To create $x_{j,\mu}^{'}$ and $\Sigma_{x_{j,\sigma}^{'}}$ for the random test inputs, we use the method described in Section 4.3. We sample and concatenate patches from the mean and variance images corresponding to the subset of input images that are random variables. Pixels in the analogous patches of the known input images are flattened and concatenated to form $x_{j,Kn,\mu}^{'}$, the observed values of $x_{j,Kn}^{'}$ the random variable that is the known component of the test input. If all the input images are known, the test inputs are created in the same manner as the training inputs, as described in Section 4.2

Our model output is the predicted random variable $f(x_j^{'})[a]$. To solve for our model's output, we must find the distribution $p(f(x_j^{'})[a])$ by solving the intractable integral in eq: 4. We approach a solution with the same moment matching procedure presented in Section 4.3, using the split kernel (eq: 10) and probability distribution (eq: 11) derived above. The values of the known pixels have been definitively observed with the assumption of no noise. The joint probability distribution of all known pixels can be substituted with a delta function at the observed values: $\delta(x - x_{j,Kn,\mu}^{'})$. This allows us to integrate out certain contributions from the known components during the moment matching steps. Solving for the mean of $f(x_j^{'})[a]$ using these hybrid inputs we get:

$$\mu(x_j^{'}) = d_{a,hybrid}^T \beta_a$$
$$d_{a,hybrid}[i] = k_{a,Kn}(x_{i,Kn}, x_{j,Kn,\mu}^{'}) \cdot \frac{\alpha_{a,Rn}^2}{\sqrt{|\Sigma_{x_{j,\sigma}^{'}}\Lambda_{a,Rn}^{-1} + I|}} \cdot e^{-\frac{1}{2}v_{i,Rn}^T\left(\Sigma_{x_{j,\sigma}^{'}} + \Lambda_{a,Rn}\right)^{-1}v_{i,Rn}} \tag{12}$$

We use the $\cdot$ operator to denote element wise multiplication. β_a is defined in eq: 5 and $v_{i,Rn} = x_{j,\mu}^{'} - x_{i,Rn}$. Solving for the variance of $f(x_j^{'})[a]$ from the hybrid inputs we get:

$$\Sigma(x_j^{'})[a,a] = \alpha_a^2 - trace((k_a(X,X) + \sigma_{n,a}^2 I)^{-1}Q_{aa,hybrid}) + \beta_a^T Q_{aa,hybrid}\beta_a - \mu(x_j^{'})[a]$$
$$Q_{aa,hybrid} = Q_{aa,Kn} \cdot Q_{aa,Rn}$$
$$Q_{aa,Kn} = k_{a,Kn}(x_{j,Kn,\mu}^{'}, X_{Kn})^T k_{a,Kn}(x_{j,Kn,\mu}^{'}, X_{Kn}) \in \mathbb{R}^{n \times n}$$
$$Q_{aa,Rn}[i,k] = \frac{1}{\sqrt{|R_{Rn}|}} \cdot k_{a,Rn}(x_{i,Rn}, x_{j,\mu}^{'}) k_{a,Rn}(x_{k,Rn}, x_{j,\mu}^{'}) \cdot e^{\frac{1}{2}z_{ik,Rn}^T R_{Rn}^{-1}\Sigma_{x_{j,\sigma}^{'}}z_{ik,Rn}} \tag{13}$$

$R_{Rn} = \Sigma_{x_{j,\sigma,}^{'}}(2\Lambda_{a,Rn}^{-1}) + I$ and $z_{ik,Rn} = \Lambda_{a,Rn}^{-1}v_{i,Rn} + \Lambda_{a,w}^{-1}v_{k,Rn}$. We use these equations to predict every pixel in the future image. Specific details for each step of the rollout are discussed in Section A.3 A walkthrough derivation of these equations along with the specifics for each step of the rollout can also be found in section A.3.

A.2 PREDICTION WITH ALL RANDOM INPUTS: DERIVATIONS

In this section we discuss the derivations for the equations 5 and 6. These equations predict the mean and variance of the output distribution of a single pixel in a future image $p(f(x_j^{'})[a])$. These predictions are done from input $x_j^{'}$ which is a multivariate gaussian random variable $x_j^{'} \sim \mathcal{N}(x_{j,\mu}^{'}, \Sigma_{x_{j,\sigma}^{'}})$ defined in Section 4.3.

Mean Prediction Derivation: We begin by walking through the derivation of the mean $\mu(x_j^{'})[a]$ of the output distribution $p(f(x_j^{'})[a])$ presented in equation 5. For the following derivations we simplify the notation from $p(f(x_j^{'})[a]|x_j^{'}, X, f(X)[:,a])$ to $p(f(x_j^{'})[a]|x_j^{'})$ as X and $f(X)[:,a]$ are known quantities. We begin our moment matching based derivation by taking the mean of the intractable integral presented in equation 4.

$$\begin{aligned}\mu(x_j^{'})[a] = E_f[\int_{-\infty}^{\infty} p(f(x_j^{'})[a]|x_i^{'})p(x_j^{'})dx_i^{'}] = E_{f,x_j^{'}}[p(f(x_j^{'})[a]|x_j^{'})] \\ = E_{x_j^{'}}[E_f[p(f(x_j^{'})[a]|x_j^{'})]]\end{aligned} \tag{14}$$

$E_f[p(f(x_j^{'})[a]|x_j^{'})]$ is the analytical form of the mean during Gaussian Process Regression from equation 1. Substituting this formula into the above equations we get:

$$\mu(x_j^{'})[a] = E_{x_j^{'}}[k_a(x_j^{'}, X)[k_a(X,X) + \sigma_{n,a}^2 I]^{-1} f(X)[:,a]] \tag{15}$$

We denote $\beta_a \in \mathbb{R}^n$ to be $[k_a(X,X) + \sigma_n^2 I]^{-1} f(X)[:,a]]$. We denote $d_a \in \mathbb{R}^n$ to be $E_{x_j^{'}}[k_a(x_j^{'}, X)]$.

$$d_a[i] = \int_{-\infty}^{\infty} k_a(x_j^{'}, x_i)p(x_j^{'})dx_j^{'} \tag{16}$$

$$\mu(x_j^{'})[a] = d_a^T \beta_a = \beta_a^T d_a \in \mathbb{R} \tag{17}$$

Expanding k_a to the RBF kernel equations and $p(x_j^{'})$ to the multivariate gaussian pdf, we solve for d_a using Vinga (2004). This gives us the mean prediction equations listed in 5 and relisted below in equation 18:

$$\begin{aligned}\mu(x_j^{'})[a] = d_a^T \beta_a \\ d_a[i] = \frac{\alpha_a^2}{\sqrt{|\Sigma_{x_{j,\sigma}^{'}} \Lambda_a^{-1} + I|}} e^{-\frac{1}{2} v_i^T (\Sigma_{x_{j,\sigma}^{'}} + \Lambda_a)^{-1} v_i} \\ \beta_a = [k_a(X,X) + \sigma_n^2 I]^{-1} f(X)[:,a]\end{aligned} \tag{18}$$

Variance Prediction Derivation: We now walk through the derivation of the predicted variance $\Sigma(x_i^{'})[a,a] \in \mathbb{R}$. Let $\Sigma(x_i^{'}) \in \mathbb{R}^{(p-b)^2 \times (p-b)^2}$ be the covariance matrix of the predicted output, where $\Sigma(x_i^{'})[a,a] \in \mathbb{R}$ is the variance of output $f(x_i^{'})[a]$. Due to our independence assumptions between outputted pixel distributions, we assert that the covariance between outputs $f(x_j^{'})[a]$ and $f(x_j^{'})[b]$, representing different output dimensions, $\Sigma(x_i^{'})[a,b] = 0, \ \forall a \neq b$.

$$\Sigma(x_j^{'}) = E_{x_j^{'}}\left[\left(f(x_j^{'}) - \mu(x_j^{'})\right)^T \left(f(x_j^{'}) - \mu(x_j^{'})\right)\right] \tag{19}$$

This is simplified using the law of total variance.

$$\Sigma(x_j^{'})[a,a] = E_{x_j^{'}}\left[var_f(f(x_j^{'})[a]|x_j^{'})\right] + E_{f,x_j^{'}}\left[f(x_j^{'})[a]f(x_j^{'})[a]\right] - \mu(x_j^{'})[a]^2 \tag{20}$$

$$E_{f,x_j^{'}}\left[f(x_j^{'})[a]f(x_j^{'})[a]\right] = \int_{-\infty}^{\infty} E_f\left[f(x_j^{'})[a]|x_j^{'}\right] E_f\left[f(x_j^{'})[a]|x_j^{'}\right] p(x_j^{'})dx_j^{'} \tag{21}$$

$E_f\left[f(x_j^{'})|x_j^{'}\right]$ is the mean output of standard Gaussian Process Regression from equation 1.

Substituting this into the above equations we have:

$$\begin{aligned} E_{f,x_j^{'}}\left[f(x_j^{'})[a]f(x_j^{'})[a]\right] &= \int_{-\infty}^{\infty} \beta_a^T k_a(x_j^{'}, X)^T k_a(x_j^{'}, X)\beta_a p(x_j^{'})dx_j^{'} \\ &= \beta_a^T \int_{-\infty}^{\infty} k_a(x_j^{'}, X)^T k_a(x_j^{'}, X)p(x_j^{'})dx_j^{'}\beta_a \end{aligned} \tag{22}$$

We define $Q_{aa} = \int_{-\infty}^{\infty} k_a(x_j^{'}, X)^T k_a(x_j^{'}, X)p(x_j^{'})dx_j^{'} \in \mathbb{R}^{(p-b)^2\times(p-b)^2}$. β_a is defined in the above sections. This gives us:

$$\begin{aligned} E_{f,x_i^{'}}\left[f(x_i^{'})[a]f(x_i^{'})[a]\right] &= \beta_a^T Q_{aa}\beta_a \\ Q_{aa}[i,k] &= \frac{k_a(x_i, x_j^{'})k_a(x_k, x_j^{'})}{\sqrt{|R|}} e^{z_{ik}^T \mathbb{R}^{-1}\Sigma_{x_{j,\sigma}^{'}} z_{ik}} \end{aligned} \tag{23}$$

$z_{ik} = \Lambda_a^{-1}v_i + \Lambda_a^{-1}v_k$ and $R = \Sigma_{x_{j,\sigma}^{'}}[\Lambda_a^{-1} + \Lambda_a^{-1}] + I$ where $I \in \mathbb{R}^{n\times n}$ is the identity matrix.

In the first term of equation 20, $E_{x_j^{'}}\left[var_f(f(x_j^{'})[a]|x_j^{'})\right]$, $var_f(f(x_j^{'})[a]|x_j^{'})$ is the variance output of Gaussian Process Regression from equation 2. Simplifying this term we get:

$$E_{x_j^{'}}\left[var_f(f(x_j^{'})[a]|x_j^{'})\right] = \alpha^2 - trace\left((k_a(X,X) + \sigma_{n,a}^2 I)^{-1}Q_{aa}\right) \tag{24}$$

We substitute the equations from 24, 23 and 5 into 20 to compute the variance for each outputted pixel corresponding to output dimensions $a \in [0, (p-b)^2-1]$. This results in the variance prediction formula presented in equations 6 and 7 and relisted below as equations 25 and 26

$$\Sigma(x_j^{'})[a,a] = \alpha_a^2 - trace\left((k_a(X,X) + \sigma_{n,a}^2 I)^{-1}Q_{aa}\right) + \beta_a^T Q_{aa}\beta_a - \mu(x_j^{'})[a] \tag{25}$$

$$Q_{aa}[i,k] = \frac{k_a(x_i, x_j^{'})k_a(x_k, x_j^{'})}{\sqrt{|R|}} e^{\frac{1}{2}z_{ik}^T R^{-1}\Sigma_{x_{j,\sigma}^{'}} z_{ik}} \tag{26}$$

A.3 Prediction with Hybrid and Fully Known inputs: Derivations and Additional Details

In this section we discuss the derivations for the equations 12 and 13. These equations predict the mean and variance of the output distribution of a single pixel in a future image $p(f(x_j^{'})[a])$. These predictions are done from input $x_j^{'}$ which is composed of two random variables $x_{j,Rn}^{'}$ and $x_{j,Kn}^{'}$ explained in Section A.3.

Mean Prediction Derivation: In this section we discuss the derivation of the mean prediction $\mu(x_j^{'})[a]$ of the output distribution $p(f(x_j^{'})[a])$ when using a combination of known and random input images, for the first 3 predictions of a rollout. We show the derivation for equation 12. Since this is a special case of prediction from all random inputs we begin our derivation from the derivation of the mean prediction equations for all random inputs in section A.2 We begin our derivation from equations 15, 16 and 17.

In this derivation we use the split kernel and probability density functions in equations 10 and 11 to deal with the hybrid, random and known, nature of the inputs. The joint probability distribution of all known pixels can be substituted with a delta function at the known values: $\delta(x - x_{j,Kn,\mu}^{'})$. This allows us to integrate out certain contributions from the known components. β_a, being a constant, remains unchanged and we re derive d_a as $d_{a,hybrid}$.

$$\begin{aligned} d_{a,hybrid}[i] &= E_{x_j^{'}}[k_a(x_j^{'}, X[i])] \\ &= E_{x_{j,Rn}^{'}, x_{j,Kn}^{'}}[k_{a,Rn}(x_{j,Rn}^{'}, X_{Rn}[i])k_{a,Kn}(x_{j,Kn}^{'}, X_{Kn}[i])] \\ &= E_{x_{j,Rn}^{'}}[k_{a,Rn}(x_{j,Rn}^{'}, X_{Rn})]E_{x_{j,Kn}^{'}}[k_{a,Kn}(x_{j,Kn}^{'}, X_{Kn})] \\ &= \int_{-\infty}^{\infty} k_{a,Kn}(x_{j,Kn}^{'}, X_{Kn})\delta(x_{j,Kn,\mu}^{'} - x_{j,Kn}^{'})dx_{j,Kn}^{'} \int_{-\infty}^{\infty} k_{a,Rn}(x_{j,Rn}^{'}, X_{Rn})p(x_{j,Rn}^{'})dx_{Rn}^{'} \end{aligned} \tag{27}$$

Solving this yields the mean prediction for the third rollout given in 12 relisted below as equation 28:

$$\mu(x_j^{'}) = d_{a,hybrid}^T \beta_a$$
$$d_{a,hybrid}[i] = k_{a,Kn}(x_{i,Kn}, x_{j,Kn,\mu}^{'}) \cdot \frac{\alpha_{a,Rn}^2}{\sqrt{|\Sigma_{x_{j,\sigma}^{'}} \Lambda_{a,Rn}^{-1} + I|}} \cdot e^{-\frac{1}{2} v_{i,Rn}^T \left(\Sigma_{x_{j,\sigma}^{'}} + \Lambda_{a,Rn}\right)^{-1} v_{i,Rn}} \tag{28}$$

Variance Prediction Derivation: Here we discuss the derivation of the variance $\Sigma(x_i^{'})[a,a] \in R$ in equation 13 from hybrid and random inputs. We follow the method outlined in the derivation for all random inputs. We use the split kernel and probability density functions in equations 10 and 11 to separately deal with the random and known components of the inputs. With this we arrive at an identical formulation to the case with all random inputs where $Q_{aa,hybrid}$ is used in place of Q_{aa}.

$$\begin{aligned} Q_{aa,hybrid} &= E_{x_j^{'}}[k_a(x_j^{'}, X)^T k_a(x_j^{'}, X)] \\ &= E_{x_{j,Rn}^{'}}[k_{a,Rn}(x_{j,Rn}^{'}, X_{Rn})^T k_{a,Rn}(x_{j,Rn}^{'}, X_{Rn})] \cdot \\ &\quad E_{x_{j,Kn}^{'}}[k_{a,Kn}(x_{j,Kn}^{'}, X_{Kn})^T k_{a,Kn}(x_{j,Kn}^{'}, X_{Kn})] \\ &= \int_{-\infty}^{\infty} k_{a,Kn}(x_{j,Kn}^{'}, X_{Kn})^T k_{a,Kn}(x_{j,Kn}^{'}, X_{Kn}) \delta(x_{j,Kn}^{'} - x_{j,Kn,\mu}^{'}) dx_{j,Kn}^{'} \\ &\quad \int_{-\infty}^{\infty} k_{a,Rn}(x_{j,Rn}^{'}, X_{Rn})^T k_{a,Rn}(x_{j,Rn}^{'}, X_{Rn}) p(x_{j,Rn}^{'}) dx_{Rn}^{'} \end{aligned} \tag{29}$$

Here $\cdot$ denotes the element wise multiplication operator. The integrals with the multivariate gaussian pdfs result in the same solution as elaborated in the random variance derivation. Solving this yields the variance prediction given in equation 13 relisted below as equation 30:

$$\Sigma(x_j^{'})[a,a] = \alpha_a^2 - trace((k_a(X,X) + \sigma_{n,a}^2 I)^{-1} Q_{aa,hybrid}) + \beta_a^T Q_{aa,hybrid} \beta_a - \mu(x_j^{'})[a]$$
$$Q_{aa,hybrid} = Q_{aa,Kn} \cdot Q_{aa,Rn}$$
$$Q_{aa,Kn} = k_{a,Kn}(x_{j,Kn,\mu}^{'}, X_{Kn})^T k_{a,Kn}(x_{j,Kn,\mu}^{'}, X_{Kn}) \in \mathbb{R}^{n \times n}$$
$$Q_{aa,Rn}[i,k] = \frac{1}{\sqrt{|R_{Rn}|}} \cdot k_{a,Rn}(x_{i,Rn}, x_{j,\mu}^{'}) k_{a,Rn}(x_{k,Rn}, x_{j,\mu}^{'}) \cdot e^{\frac{1}{2} z_{ik,Rn}^T R_{Rn}^{-1} \Sigma_{x_{j,\sigma}^{'}} z_{ik,Rn}} \tag{30}$$

Rollout Discussion: In this section we discuss the composition of our inputs and additional details of each step in our predictive rollout. In a predictive rollout we are predicting T time steps into future starting from 3 known, consecutive input images $[z_i, z_{i+1}, z_{i+2}]$. With each prediction we predict a single time step into the future before incorporating our prediction into our next set of inputs. We continue this process until we predict the desired number of time steps.

First Step: The first step of the predictive rollout predicts z_{i+3} from input images $[z_i, z_{i+1}, z_{i+2}]$. For this first prediction all the input images are known quantities. As a result for each test input $x_j^{'}$ $j \in [0, m-1]$, the entire test input is known, $x_j^{'} = x_{j,Kn}^{'}$, and $x_{j,Rn}^{'}$ does not exist. $x_{j,Kn,\mu}^{'} \in \mathbb{R}^{3p^2}$ is formed in a manner identical to the training inputs. When plugging these inputs into the hybrid mean prediction equation 12 and variance prediction equation 13 we remove the random components of the equations giving us:

$$d_{a,hybrid}[i] = k_{a,Kn}(x_{i,Kn}, x_{j,Kn,\mu}^{'}) \tag{31}$$
$$Q_{aa,hybrid} = Q_{aa,Kn} \tag{32}$$

Substituting these back into the equations 12 and 13 we get the formulas to predict a single output dimension of for a single test input for the first prediction. These equations equivalent to the basic Gaussian Process Regression equations for mean and variance prediction. Using these formulas we predict the distribution for every pixel in z_{i+3} from the m test inputs. This gives us the final predicted image z_{i+3} which is stored as a mean, variance image tuple (M_{i+3}, V_{i+3}).

Second Step: The second step of the predictive rollout predicts z_{i+4} from input images $[z_{i+1}, z_{i+2}, z_{i+3}]$. z_{i+3} is a random variable, from the first prediction, represented by the mean and variance image tuple (M_{i+3}, V_{i+3}). z_{i+1} and z_{i+2} are known. When constructing each test input, $x^{'}_{j,Kn,\mu}$ is constructed from flattened and concatenated patches of $[z_{i+1}, z_{i+2}]$. $x^{'}_{j,\mu}$ is constructed from flattened patches of M_{i+3} and $x^{'}_{j,\sigma}$ is constructed from flattened patches of V_{i+3} to create the random input $x^{'}_{j,Rn}$. These components together form a single test input $x^{'}_{j}$. We plug these inputs into the hybrid mean prediction equation 12 and variance prediction equation 13 to compute the model's output distribution for a single output dimension. We repeat this to predict the output distributions for each pixel in the future image z_{i+4} which is stored as a mean and variance image tuple (M_{i+3}, V_{i+3}).

Third Step: The third step of the predictive rollout predicts z_{i+5} from input images $[z_{i+2}, z_{i+3}, z_{i+4}]$. z_{i+3} and z_{i+4} are random variables, from the first and second predictions, represented by the mean and variance image tuples $(M_{i+3}, V_{i+3}), (M_{i+4}, V_{i+4})$. z_{i+2} is known. When constructing each test input, $x^{'}_{j,Kn,\mu}$ is constructed from flattened patches of $[z_{i+2}]$. $x^{'}_{j,\mu}$ is constructed from flattened concatenated patches of M_{i+3}, M_{i+4} and $x^{'}_{j,\sigma}$ is constructed from flattened patches of V_{i+3}, V_{i+4} to create the random input $x^{'}_{j,Rn}$. These components together form a single test input $x^{'}_{j}$. We plug these inputs into the hybrid mean prediction equation 12 and variance prediction equation 13 to compute the model's output distribution for a single output dimension. We repeat this to predict the output distributions for each pixel in the future image z_{i+5} which is stored as a mean and variance image tuple (M_{i+5}, V_{i+5}).

Fourth Step and Onwards: The third step of the predictive rollout predicts z_{i+6} from input images $[z_{i+3}, z_{i+4}, z_{i+5}]$. For this and all subsequent predictions, all the input images are random variables outputted by our model. To predict the future image we utilize the approach detailed in the section 4.3 on Prediction with all random inputs.

A.4 Predictive Comparison Experiment: Additional Details

In this section we highlight additional details on the methodology used to generate the results for the 'Predictive Comparison' Experiment in Section 5. To compare all three methods, we predict frames $[z_{10}, \ldots, z_{24}]$ given frames $[z_0, \ldots, z_9]$. For our method we train our model using $[z_0, \ldots, z_9]$ and begin our prediction with input images $[z_7, z_8, z_9]$, identical to the approach outlined in the 'Forward Prediction' Experiment.

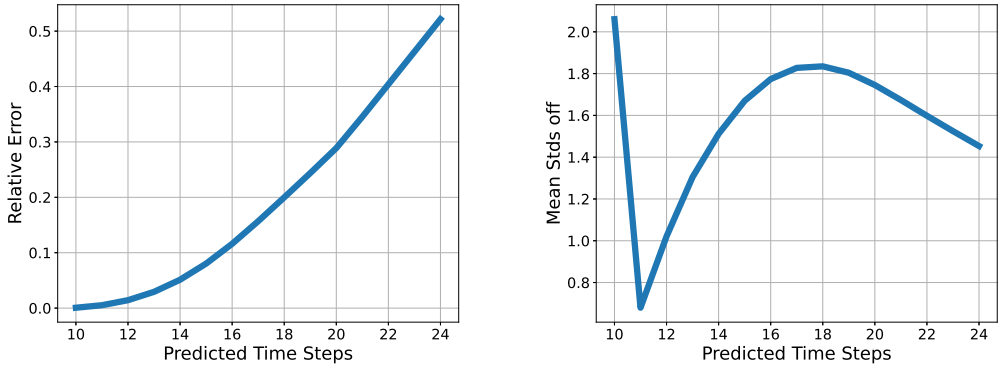
The FNO-2d-time model convolves across the two spatial dimensions to predict a single future image with a recurrent structure in time. This model uses a rollout method to predict longer video sequences. The model predicts one future frame at a time and incorporates its last prediction into its next input. We continue this until we have predicted the desired number of frames. The model is structured to take an input of three images and predict a single future image. We train this method in a manner similar to ours. The training data is created using the first 10 frames $[z_0, \ldots, z_9]$. These frames are separated into data points of 4 consecutive images $[z_i, \ldots, z_{i+3}]; \forall i \in [0, 6]$, where the first three images form the input and the fourth is the output. The model is trained over 20 epochs. The trained model is then used to predict 15 frames $[z_{10}, \ldots, z_{24}]$ of the same sequence.

The FNO-3d model is a neural network that convolves in space and time to directly output several frames from a set of input frames. We train this model using the first 25 frames from two unique sequences, generated using the same simulation parameters. The first 10 images are used as the inputs, with the remaining 15 serving as the training outputs. Once trained over 20 epochs this model is given a set of 10 consecutive frames $[z_0, \ldots, z_9]$ to predict the next 15: $[z_{10}, \ldots, z_{24}]$.

The results of this comparison are shown in fig. 5.

A.5 Average Result Metrics

We showcase the average relative error and average mean standard deviations off of our model's predictions evaluated on 100 separate video sequences in Figure 6. For each video sequence our model is trained using the first 10 frames $[z_0, \ldots, z_9]$ and used to predict the next 15 frames, $[z_{10}, \ldots, z_{24}]$. We use the parameters discussed in the 'Forward Prediction' experiment.



(a) Average Relative Errors

(b) Average Mean Standard Deviations off

Figure 6: Averaged Metrics: Fig (6a) Shows the average relative error metrics using our method over 100 results on separate video sequences. (6b) Shows the average mean standard deviations off the predicted image is from the ground truth using the mean and variance generated using our method. This result is also averaged over predictions on 100 different video sequences. To generate the above results our model was trained on frames $[z_0, \ldots, z_9]$ of each sequence and used to predict the next 15 frames, $[z_{10}, \ldots, z_{24}]$

A.6 Training Input Creation Graphic

The Figure 7 graphically demonstrates the process of creating a training datapoint from patches of 4 consecutive video frames.

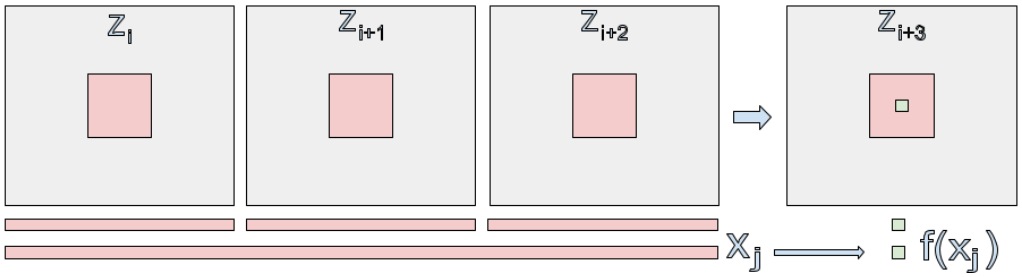


Figure 7: Graphic to visualize the process of generating a training data point from a sequence of 4 consecutive video frames. Each grey figure represents a labelled video frame. The red squares represent the patches that are sampled to create the input output pair. In the fourth image the green pixel represents the output patch after cropping the patch with patch border b. The patches from the first 3 images are flattened and concatenated to form the input. The patch from the fourth image is flattened and used as the output. In this case the output patch is a single pixel.

A.7 Method Overview Graphic

The Figure 8 shows a graphical overview of the method.

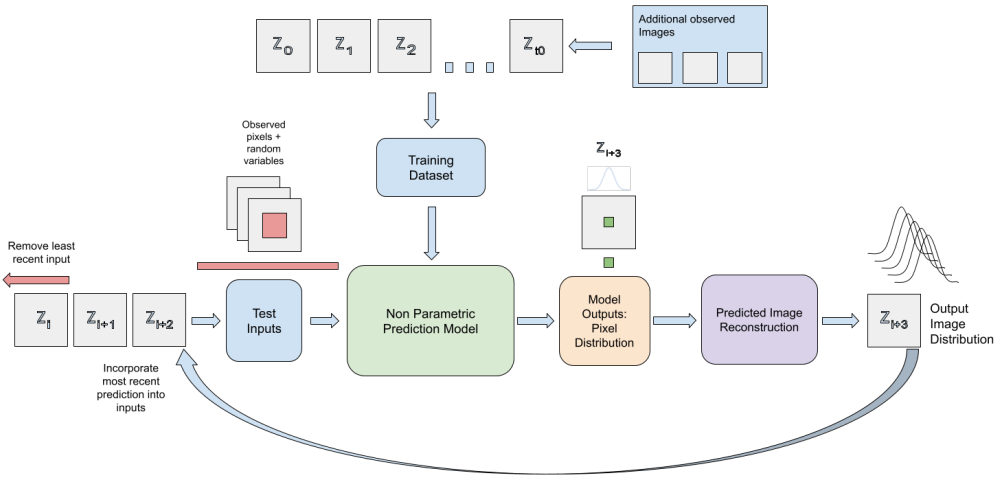


Figure 8: Graphic to visualize the overall method