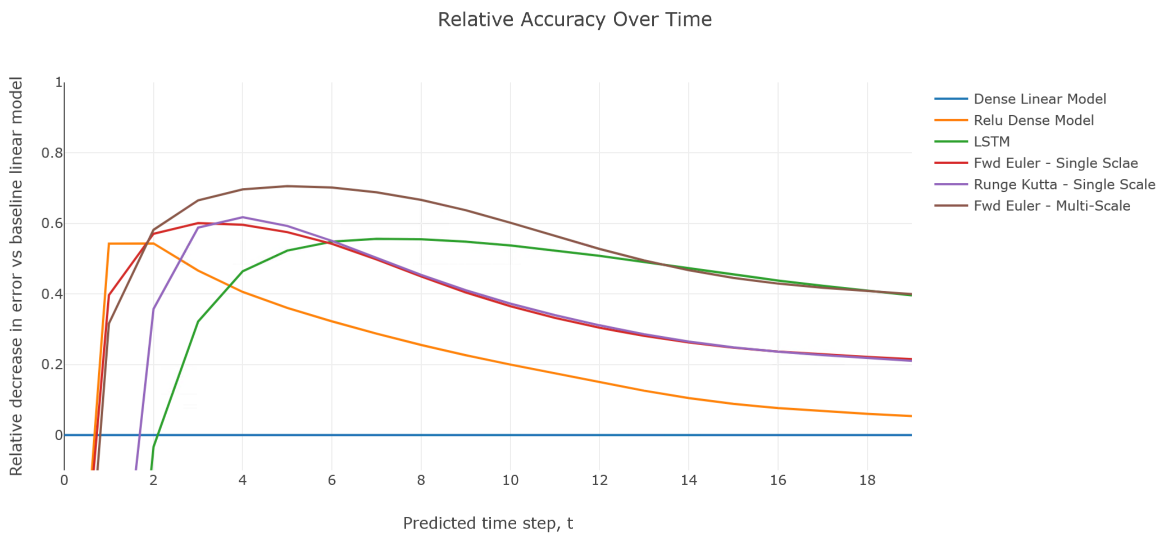

Figure 1: Relative baseline performance of various architectures as compared to a fully connected linear model (6.25 million parameters). A relative error decrease of 1.0 represents perfect prediction for a given time-step. Note the advantage of the foward-Euler multi-scale architecture across most time-steps.

1 Executive summary

Previously, we developed a forward Euler inspired network architecture that rivaled the performance of the previous LSTM based architectures. We also demonstrated the similarity of this architecture to numeric solvers for partially differential equations (PDEs).

To evaluate new models in a standard way, we first establish a baseline performance using a fully connected linear network, and introduce a relative baseline performance metric, where σ represents the mean squared error of the model for a given time-step t.

$$RBP = \frac{1}{n}\sum_{t=1}^{n}\frac{\sigma_{baseline}-\sigma_{model}}{\sigma_{baseline}}$$

This gives us a score of 1 for models that perfectly explain the test data, and a score of 0 for models as good as a linear model (negative scores indicate models perform worse than the fully connected linear baseline).

In this period we significantly increased the performance of model architectures discovering new learning procedures and model architectures. As compared to our previous LSTM models with relative baseline performance (RBP) of 0.358, further improvements to training procedures (hypothesis 2) increased the RBP of the previously discovered forward Euler architecture to 0.386. Furthermore, by introducing multi-scale networks (hypothesis 1,) we see a 28.7% reduction in mean squared error over 20 time steps resulting in a new architecture with an RBP of 0.548. Finally, we introduce a curriculum learning approach (hypothesis 3) that leverages different scales to train architectures that were previously impossible to train given their sensitivity to co-variate shifts.

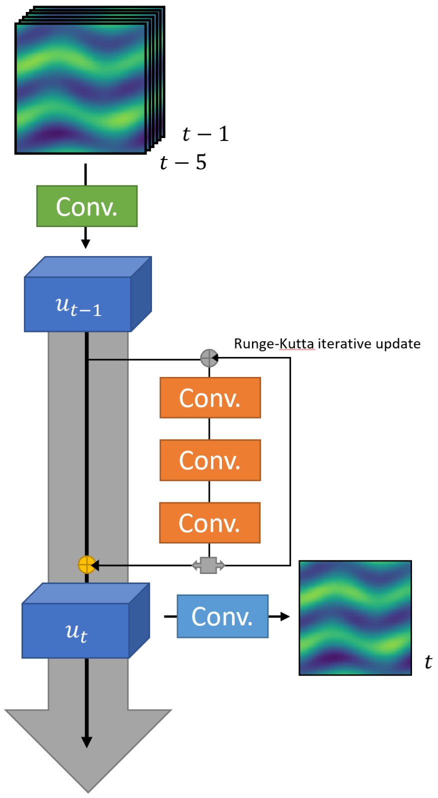

Figure 2: Illustration of the iterative Runge-Kutta architecture. The encoder $\phi(x)$ (shown in green) maps the input x into the feature space u_t. Three sequential convolutional layers (shown in orange) are applied, using the activation from the previous time-step to calculate the next update as described in the RK4 method. This generates a more precise estimate of the dynamics, taking into account the curvature at the mid-point as well as either end rather than the one-sided explicit update used by the Euler method. The learned dynamics update $f(u_t)$ is then summed with u_t (shown in yellow) to give u_{t+1}. At each time step t, the feature vector u_t is decoded by $\theta(u_t)$ (as shown in light blue) predicting the vorticity at time t, x_t.

1.1 Hypothesis 1 - Multi-scale embedded dynamics

We proposed using convolutions at varying scales to increase the fidelity of the dynamics model while maintaining the symmetry-constrained properties of a convolutional architecture. We evaluate the performance of the forward-Euler architecture (illustrated in figure 3)with and without multi-scale features, figure 5, and see significant improvement in relative performance resulting in a 28.7% decrease in mean squared error as compared to the previous best architecture.

1.1.1 What was to be tested? What was the expected outcome prior to testing? We test the addition of convolutional features at different scales to learn the problem dynamics more accurately. The resulting architecture is illustrated in figure 4. Previous architectures used only a single scale for convolutional features meaning the network had access to only local information. By adding multiple scales to the dynamics update we hope to enable the network to learn to use both local and global features in the region to predict more accurately.

1.1.2 High-level summary of main results. We see strong performance gains by the addition of the additional scales. As shown in figure 1, for the multi-scale architecture prediction is accurate over both the short-term and long-term predictions, beating LSTM based methods in short-term prediction.

1.1.3 Discuss relevance to project and DARPA concerns. In numeric PDE solvers, choosing the proper simulation scale can dramatically effect the accuracy of simulations even for known PDEs. Given the increase in performance we see that with the addition of multi-scale convolutional layers, networks learn the proper scale for accurate prediction dynamically. This results in networks that are general and adapt well to new problems.

1.2 Hypothesis 2 - Iterative approximation methods

We implemented the fourth-order Runge-Kutta method by iteratively sampling the dynamics model (shown in orange in figure 2) to obtain a more accurate estimate of the next time-step, u_t. In testing this method directly on single-scale architectures, performance of the model both computationally and numerically were reduced. Further investigation is needed to determine if a different formulation of the loss function could be effective in lowering compounding error and enabling better long-term predictions.

1.2.1 What was to be tested? What was the expected outcome prior to testing? Given the similarity between our network architecture and the update step used by forward-Euler approximations, we proposed integrating techniques effective in reducing the compounding error approximate solutions for ordinary differential equations. Using the existing architecture to predict the next point, taking a small gradient step in that direction, then updating our prediction. We expected this method to increase the accuracy of the learned model by adjusting for local curvature.

1.2.2 High-level summary of main results. The fourth order Runge-Kutta method as applied (see figure 2) did not improve performance and increased the computational requirements during training. Future work is needed to explore if this method could be effective if introduced into the curriculum learning approach outlined in hypothesis 3. Runge-Kutta as tested in these experiments reduces to an averaging of an iterative procedure which may be intractable given the non-linear nature of the model architecture.

1.2.3 Discuss relevance to project and DARPA concerns. If successfully applied, iterative methods can leverage traditional understanding of time-discretized ordinary differential equations to improve the accuracy of models without requiring additional training steps or more training data.

1.3 Hypothesis 3 - Scale adapted curriculum learning

In experiments concerning hypothesis 2, we experienced an explosion of gradient updates - where small shifts in network parameters were amplified making learning impossible for these interesting network formulations. To address this issue, we proposed a dynamic adjustment of the prediction horizon while training, forming a curriculum-learning approach where simpler short-term dynamics are learned before introducing longer-term dynamics which have higher sensitivity to rapid shifts in network dynamics. This approach was effective in enabling experiments on iterative descritized approximation methods.

1.3.1 What was to be tested? What was the expected outcome prior to testing? We evaluate models over increasingly distant time-steps to determine if this procedure regularized the network and prevented explosion of gradients. Given the short reward horizon we expected this method to decrease the variability of gradient updates and allow for stable gradient decent.

1.3.2 High-level summary of main results. This approach expanded the domain of model architectures to experiment on and increased the stability of training allowing for more aggressive gradient steps as well as reducing the amount of data required for early training steps.

1.3.3 Discuss relevance to project and DARPA concerns. By varying the prediction length early on, the decrease in computation dramatically speeds up training time and increases the stability of the learning procedure.

2 Achievements

2.1 Scientific Breakthroughs

None

2.2 Technology developments

None

2.3 Application results

Predictive performance has demonstrated that despite the difficult problem of predicting chaotic systems, using scale-adapted and symmetry constrained architectural components are able to predict better than LSTM based architectures and significantly better then baseline linear models.

2.4 Transitions achieved

None

3 Lessons Learned

Problems encountered/risks that occurred, and corresponding solutions/mitigations

None

Open Issues

None

4 Next Steps

We would like to further refine the multi-scale architecture, specifically testing networks that include additional scales of convolutional features.
Current results reflect only one ground truth scale. Testing with super-sampling and sub-sampling to establish scale invariance would be an interesting exploration for the current model architecture.
When used naively, fourth order Runge-Kutta did not improve the performance of the model. Instead of using Runke-Kutta directly on the same $f(u_t)$ we would like to explore learning separate iterative midpoints to use in the dynamics update step.

5 Technical details

5.1 Formulating Multi-Scale Architecture

Consider learning a PDE of the form $u_t = f(u)$. The simplest approximation of $u_t = f(u)$ is to discretize the time derivative u_t by $\frac{u_{n+1}-u_n}{\triangle t}$ and approximate the right hand side by $f(u_n)$. This leads to the forward Euler scheme

$$u_{n+1} = u_n + \triangle t f(u_n)$$

where $f(u_n)$ is learned by a deep neural network. Instead of learning the dynamics of the system directly, we use a learned feature representation u_n composed of linear convolutions of the input history. This avoids the need for approximating the local partial derivatives of the system and instead allows the network to learn the optimal feature representation.

Let x represent h patches of history $x = [x_{t-h}, ..., x_{t-1}]$ and y represent the target sequence $y = [x_t, x_{t+1}, ..., x_{t+l-1}]$. We learn an encoder, $\phi(x) = u_0$, a decoder $\theta(u_n) = \hat{x}_n$ as well as a dynamics model $f(u_n)$ from equation 5.1.

In the new multi-scale architecture shown in figure 4 we utilize a feature space of dimension m = 64, and reduce the spacial dimensions of the image patch by a factor of 4 therefore, $u_n \in \mathbb{R}^{64,25,25}$.

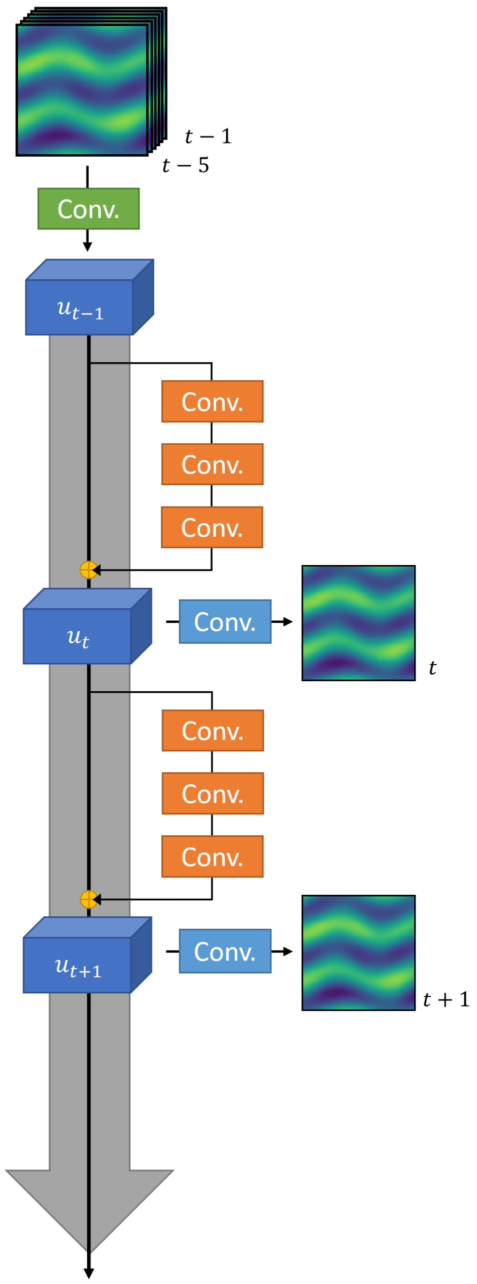

Figure 3: Illustration of the previous forward Euler architecture. The encoder $\phi(x)$ (shown in green) maps the input into the feature space. Three convolutional layers (shown in orange) are used to approximate $f(u_t)$ and summed with u_t (shown in yellow) to give u_{t+1} as outlined in equation 5.1. At each step, the feature vector u_t is decoded by $\theta(u_t)$ (as shown in light blue) predicting x_t.

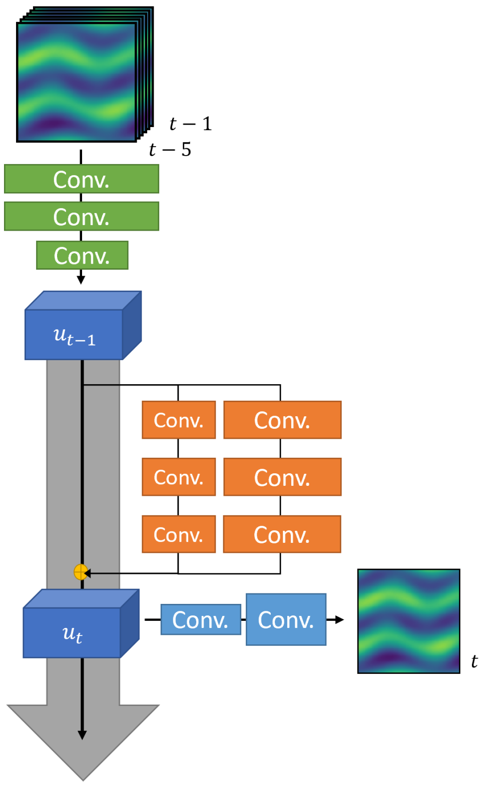

Figure 4: Illustration of the multi-scale forward Euler architecture. The dynamics $f(u_t)$ are enriched with filters at different scales (shown in orange). Additionally, the encoder $\phi(x)$ reduces the dimensionality of the input image by a factor of 4 in the encoding step $\phi(x)$ to increase the receptive field of the dynamics update calculation, $f(u_t)$.

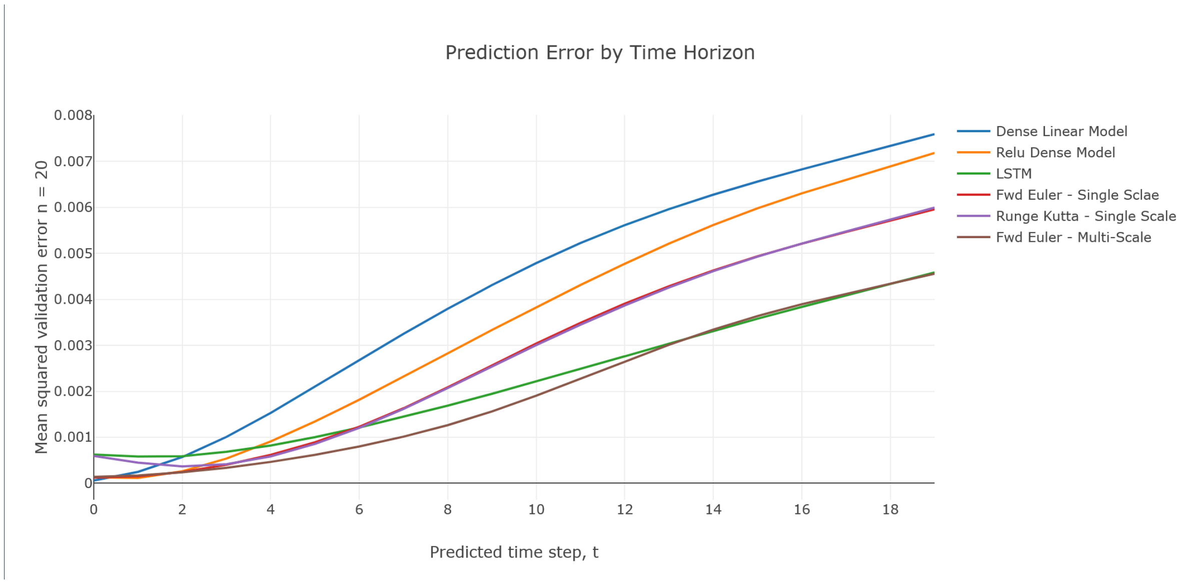

Figure 5: Relative baseline performance of various architectures. Note the advantage of the multi-scale architecture across most time-steps.

In constructing the multi-scale element of $f(u_n)$, the first layer of each scale reduces the feature space (64) down to $\frac{1}{num\_scales}$ features (32 as implemented in the multi-scale architecture outlined in figure 4). Each scale is then computed in parallel and then the activations are concatenated. This allows each scale to signal other scales while lowering the number of learnable parameters as show to be successful in [F. Iandola et al. 2016]. Finally, the decoder $\theta(u_n)$ uses a de-convolution of size 2 to increase the spacial dimension of the embedded dynamics u_n to 50x50x8 and a final linear convolution smooths the output of the network.

5.2 Scale adapted curriculum learning

Implementing multi-scale curriculum learning simple anneals the prediction horizon during training. By loading the model weights from previous runs, we start training the model to predict 2 time-steps into the future, increasing the prediction horizon by 2 until the desired prediction length is achieved. This simple modification both increased the number of training batches per second, and thus training speed, but also prevented the iterative R4 models from diverging during training which resulted because of the repeated application of model parameters causing the explosion of resultant gradients over long reward horizons.