**Sample Plots**

The following set of plots summarize the fitted and predicted lightcurve using CARMA, GP, Vanilla LSTM and Attention LSTM of the QSO with index . For each lightcurve, of the data points (green) are used for training, (blue) for validation and the rest for testing.

|  |  |
| --- | --- |
|  |  |
| **Figure 1.1 CARMA Simulated Path** | **Figure 1.2 GP Simulated Path** |
| **A screenshot of a video game  Description automatically generated** | **A screenshot of a cell phone  Description automatically generated** |
| **Figure 1.3 Vanilla LSTM** | **Figure 1.4 Vanilla LSTM Residual** |
| **A screenshot of a video game  Description automatically generated** | **A screenshot of a computer  Description automatically generated** |
| **Figure 1.5 Attention LSTM** | **Figure 1.6 Attention LSTM Residual** |

**Result Summary**

The models are evaluated using the MSE between the predicted and the observed magnitude in the training and the cross-validation set of each lightcurve. The mean and standard deviation of the MSE for the entire dataset (containing ~50 lightcuevs) is summarized in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Training Loss | Validation Loss  (Single-step) | Validation Loss  (Multi-step) |
| Gaussian |  |  |  |
| CARMA |  |  |  |
| Vanilla Standard |  |  |  |
| Vanilla Phased |  |  |  |
| Attention Standard |  |  |  |
| Attention Phased |  |  |  |

**Figure. Summary of Training & Validation Loss**

There are serval observations from this table:

* The validation loss is always greater than the training loss, suggesting that the models might be overfitted. However, this is expected as our time-series is stochastic, and we do not expect our model to capture the uncertainties in the out-of-sample data
* The single-step prediction always performs better than the multi-step prediction, this is also due to the stochastic nature of our lightcurve. The uncertainty in the magnitude diverges as we move further away from the current time-step.
* The vanilla standard LSTM achieves the best performance on the training set and the best single-step prediction on the validation set, whereas the CARMA model has the best multi-step prediction on the validation set

The deep learning based models, as they stand now, does not do as a good job as the CARMA model in the multi-step prediction on the validation set. Several reasons might account for this

* Overfitting: The LSTM models has more parameters (about ) than the number of training data (about data points) available, the models are prone to overfitting [Solution: Look for dataset that have more observation for QSOs]
* Gradient Vanishing: By looking at the predictions from the LSTM based models for other QSOs, we observe cases where the prediction gives a constant value. This is likely due to the LSTM learning only the bias, while other parameters remains untrained [Solution: Retrain the model if it get stuck]

A picture containing indoor

Description automatically generated

**Figure. Gradient Vanishing**

* Outliers: For training data with outliers, the prediction shows larger variations and deviations from the observed magnitude values. In comparison, the predictions from the CARMA models are smoother and more stable. [Solution: remove outliers]

**A screenshot of a video game

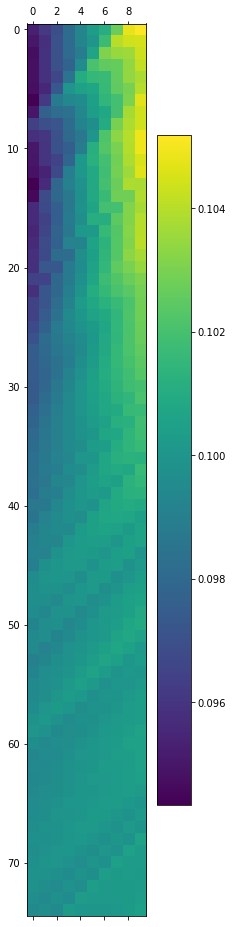
Description automatically generated**

**Figure. Lightcurve with Outliers**

**Attention Weight**

The figure below shows the attention weights of the attention LSTM for the lightcurve after the training phase. The vertical axis corresponds to different training data (each with a window length of ). The horizontal axis shows the time-step, from left to right, corresponding to time-steps ago to one time-step ago.

From the attention plot, we can observe several “diagonal” patterns. Those suggest important features that carries a big weight for several predictions whose input includes that point.

****

**Figure.** Attention Weight for Lightcurve (x-axis: look back; y-axis: sample),

**Result on A Sinusoidal Time Series**

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated | A picture containing object  Description automatically generated |
| **Figure 2.1 Vanilla LSTM** | **Figure 2.2 Vanilla LSTM Residual** |
| A picture containing sky, indoor  Description automatically generated | A picture containing indoor, object  Description automatically generated |
| **Figure 2.3 Attention LSTM** | **Figure 2.4 Attention LSTM Residual** |

(All the data above are generated from single step prediction)

* **Phased Layer**

|  |  |
| --- | --- |
| A picture containing sky  Description automatically generated | A screenshot of a cell phone  Description automatically generated |
| **Figure 2.5 Vanilla LSTM with Phased Layer** | **Figure 2.6 Vanilla LSTM with Phased Layer Residual** |

* **Multiple Step Prediction**

|  |  |
| --- | --- |
| A picture containing sky, indoor  Description automatically generated | A picture containing sky, indoor  Description automatically generated |
| **Figure 3.1 Vanilla LSTM Multiple Step Prediction** | **Figure 3.2 Phased LSTM Multiple Step Prediction** |

* **Reduced Number of Data Points**

|  |  |
| --- | --- |
| **A screenshot of a computer  Description automatically generated** | A screenshot of a cell phone  Description automatically generated |
| **Figure 4.1 Vanilla LSTM** | **Figure 4.2 Vanilla LSTM Residual** |

* **Highly Oscillatory Data**

|  |  |
| --- | --- |
| **A screenshot of a social media post  Description automatically generated** | A screenshot of a cell phone  Description automatically generated |
| **Figure 4.1 Vanilla LSTM** | **Figure 4.2 Vanilla LSTM Residual** |

**Sinusoidal Time Series Attention Map**

**A screenshot of a cell phone

Description automatically generated**

**Figure.** Attention Weight for Sinusoidal Time Series (x-axis: look back; y-axis: sample),

**TODO:**

Tune the hyper-parameters for LSTM models

Use evaluation metrics that accounts for uncertainty in observation / prediction

Build Bayesian RNN

Train the LSTM using a sine wave and see whether it works