Time Series Gap Reconstruction using (Conv)LSTMs

Marcel Plutat and Wilhelm Jochim

University of Bremen, Bremen, Germany
{mplutat,wjochim}@uni-bremen.de

Abstract. This work proposes solutions to two distinct time series gap reconstruction problems. Firstly, we employed two ConvLSTM models to reconstruct missing video frames approaching each other from two temporal directions. This strategy reduced the compounding errors from sliding window methods. Secondly, we used an LSTM-based model to fill in gaps in motion capture data. This set of problems was the task for the Bremen Big Data Challenge 2022, a yearly data analysis competition set to be solved by students. The proposed solution was the winning submission with an achieved motion capture SSIM of 1.0 and a video SSIM of 0.61.

Keywords: Big data · Time Series · ConvLSTM · LSTM · Motion Capture · Video Prediction · Bremen Big Data Challenge

Thanks: to Fabian Wetjen for his contribution to our implementation as our group member in the Bremen Big Data Challenge.

1 Introduction

The Bremen Big Data Challenge is a competition encouraging students to solve problem statements in the field of Big Data. The Challenge is held annually and the task of 2022 revolved around reconstructing time series data. Time series is a difficult class of problems that requires complex machine learning approaches. The proposed solution was developed by a group of three students attending the University of Bremen. The time frame of the contest spanned from the 1st of March to the 10th of April 2022.

2 Challenge

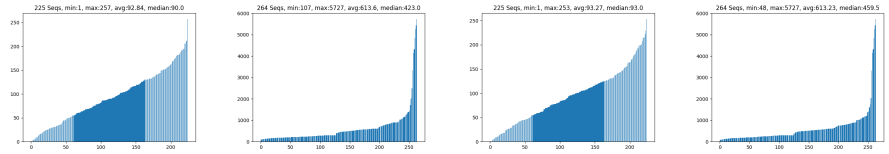
The dataset tracks a person interacting with common household items and contains two separate data formats:

1. Motion capture data with three tracked points in the form of 9 column CSV files. The tracking points are mounted to the head of the person.

2. 30 frames per second of grayscale video data with 160 pixels in width and 96 pixels in height. The camera capturing this is mounted to the head of the person.

There are multiple gaps of varying lengths in each data stream. The goal is to reconstruct the missing frames. To get acquainted with the dataset, a detailed analysis was employed which yielded more information on the occurrences and lengths of the gaps.

2.1 Data Exploration



(a) (b) (c) (d)

Fig. 1: Sequence length distribution for motion capture (a)–(b) and video (c)–(d)

The first question while exploring is the distribution of valid and missing data. Missing data is represented by a sequence of frames containing only 0 values. Checking for the existence of frames where both data streams were missing was the first step of data exploration. Approaches using valid data from one data stream to more accurately reconstruct the gap in the other data stream might be hindered by the existence of such frames. The data exploration confirmed the occurrence of this problem.

To determine the distribution of valid sequence and gap lengths, two graphs were generated for each model (see Figure 1). Of special interest was the minimum, maximum and average length. The minimum valid sequence length is required to determine the possible input and output lengths of the model. In training, valid sequence chunks need to be split into input and output data, restricted by the upper bound of minimum valid sequence length. The minimal valid data sequences are 107 for motion capture data and 48 for video.

The decision on a particular output length for the model is influenced by the distribution of the missing data gap length. Using short output lengths results in a bigger accumulation of errors through the usage of a sliding window to cover the whole gap. The average gap for motion capture data was 92 and for video 93.

An outlier value was identified in the motion capture data. There are multiple instances of the value -999999000 occurring. Our initial hypothesis for the existence of these values is measurement errors. This hypothesis was later confirmed by the organizers of the Bremen Big Data Challenge.

2.2 Our Approach

After the previously alluded to problem of frames missing in both data streams, in combination with the time constraints of a competition, an early decision was made to approach motion capture and video reconstruction as two separate problems. The implementation of a motion capture solution is described in section 3, while the video implementation is described in section 4.

3 Motion Capture

With each motion capture frame being one-dimensional it was the easier starting point and implemented first. Related solutions to the problem of reconstructing missing data sequences were researched at the beginning.

3.1 Related Work

The reconstruction of missing frames in the motion capture data is a time series prediction problem. Time series prediction can be implemented using *Long Short-Term Memory* (LSTM). **LSTM** is useful for longer input sequences as it uses gates to solve the problem of long-term dependencies[3].

Batch Normalization was introduced to normalize the activities of neurons which consequently reduces the training time for feed-forward neural networks. However, one challenge of batch normalization is that it is "not obvious how to apply it to recurrent neural networks". **Layer Normalization** builds upon batch normalization by normalizing the neurons for each sample in the batch independently. The authors claim that this is effective at stabilizing the hidden state dynamics in recurrent networks[1].

3.2 Preprocessing

Preprocessing needs to generate the input and output sequences for the training of the motion capture model. The input size referred to as history length was chosen to be 105 and the output size referred to as prediction length was chosen to be 1. This results in a total chunk size of 106 which is lower than the possible maximum of 107. All blocks of valid data are split into chunks of history and prediction length to be used in training and validation. For the prediction the valid history in front of the missing data is needed, which is also generated. Lastly, the samples are split into training and validation data with a ratio of 90/10.

Additionally, during preprocessing each chunk gets filtered to remove the outlier value and the data is normalized by subtracting the mean and dividing by the standard deviation.

3.3 Implementation

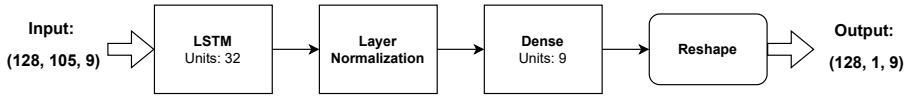


Fig. 2: Motion capture sequence prediction architecture

Our model was implemented in TensorFlow using the Keras library. We used a model checkpoint to save the best model based on the validation loss. The model, which is depicted in Figure 2, has an input shape of (128, 105, 9). Where the batch size is 128 and the history length is 105 samples. The trailing 9 of the input refers to the 9 columns of measurement data. Firstly, our model features an LSTM layer with 32 units. Afterward, we use Layer Normalization to normalize our data. The third layer is a fully-connected layer with 9 units, responding to the 9 measurements. Finally, we reshape the data to match our desired prediction length of one sample.

We trained our model for 10 epochs using the Adam optimizer. The learning rate was set to 0.01. As a loss function, we used mean squared error since it fits our target domain.

3.4 Postprocessing

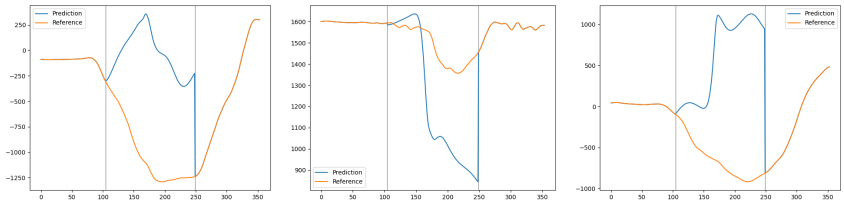
```
mocap = mocap.replace(-2147483648, -999999000)
```

Fig. 3: Code replacing the outlier value

The first step of our postprocessing pipeline is to reverse our normalization by multiplying the data by the standard deviation and adding the mean.

Using the outlier value as an input to the prediction generated an output of the minimal value representable by a signed 32-bit integer. While a more favorable solution is possible, a sufficient approach was replacing the neural networks output value with the known error value of -999999000. This was the only postprocessing necessary for the motion capture prediction data. The outlier processing is listed in Figure 3.

3.5 Evaluation



(a) Head top x (b) Head top y (c) Head top z

Fig. 4: Comparison between predicted output and reference data

Submitting the prediction generated by this implementation resulted in a score of 1.0. This means no improvement in the leaderboard position could be gained by improving the accuracy of the prediction.

Figure 4 shows an example of the actual discrepancy between the prediction and the original data, which was released after the conclusion of the challenge. The first missing data gap was chosen and the x-, y- and z-coordinates of the top-mounted tracking point are plotted. An immediate deviation from the reference data at the start of the gap can be observed, with the inaccuracy increasing over time. The model's lack of awareness of valid data following the gap is noticeable by the discontinuous nature of the prediction, here visible by a sudden and drastic change in the curve at the end of the gap. A further discussion of the discrepancy between the measured score and observed predicted values follows in section 5.

4 Video

After achieving the best possible score on the motion capture data, the focus changed to the video frame prediction.

4.1 Related Work

A similar problem is the prediction of future frames on the **Moving MNIST** dataset. Srivastava et al. created a dataset of moving MNIST digits to aid in the visualization of their proposed model [8]. "In this dataset, each video was 20 frames long and consisted of two digits moving inside a 64 x 64 patch" [8]. The Moving MNIST dataset is popular among video prediction papers with several evaluations on said dataset (see Moving MNIST leaderboard on Papers with Code [6]).

One of the first network architectures to be used on the Moving MNIST dataset is the *Convolutional LSTM* (ConvLSTM). The aim of the **ConvLSTM**

was precipitation nowcasting which is to "predict the future rainfall intensity in a local region over a relatively short period of time". Shi et al. used radar map images which will be extrapolated to generate the weather forecast. Precipitation nowcasting is a "spatiotemporal sequence forecasting problem with the sequence of past radar maps as input and the sequence of a fixed number (usually larger than 1) of future radar maps as output". The authors mention the *fully connected LSTM* (FC-LSTM) layers' shortcomings in taking spatial correlation into consideration. The solution to this problem is "stacking multiple ConvLSTM layers and forming an encoding-forecasting structure, [through which] an end-to-end trainable model for precipitation nowcasting" can be built. A major advantage of the ConvLSTM is that contrary to the FC-LSTM spatiotemporal data does not need to be unfolded to one-dimensional inputs and therefore no spatial information will be lost [7].

The challenge used the *Structural Similarity Index Measure* (SSIM) to evaluate both motion capture data and video submissions. **SSIM** measures the perceptual quality of the image and not the strength, such as the MSE does [9]. While MSE was sufficient for motion capture data the added complexity of accurately assessing the visual quality of the video prediction made SSIM the choice for the loss function. An added benefit is that the output will be optimized towards the same score the challenge evaluation uses.

4.2 Preprocessing

For the history, the past 15 frames were used and the prediction consists of the following 5 frames. These values stem from our experiments where they were found to be among the best performing. The main function of preprocessing is twofold. Firstly, the set of complete blocks without missing frames in between is split into smaller subsets of size history length + prediction length. The resulting data is then split into the training and validation sets using a 90/10 ratio. The second goal is to create the initial data for the missing frame prediction, where the history before and after a block is needed to make the first predictions to fill the missing frames. Lastly, the data is normalized by dividing by 255, which is the maximum value a single pixel can take.

4.3 Implementation

Fig. 5: Network architecture for the video sequence prediction

Our implementation consists of two separate models, one traversing forward and the other one traversing backward through the missing frames. The gaps

are filled by using a sliding window to predict the next frames and subsequently using those to predict the next set. The major drawback of this approach is that there are compounding errors in the prediction and the end of the prediction does not contain any useful information. To solve this issue we propose using a second model which traverses the missing frames from the back and meets the forward model in the middle of the gap. While video motion should be reversible, making the reuse of the forward model for backward predictions possible by reversing the model's input, experimentation yielded higher scores with a model specifically trained on reversed data.

Our model was implemented in Pytorch based on the ConvLSTM implementation from Palazzi [5] which differs from the original paper. This implementation uses a list of ConvLSTMCells, where each element of the input sequence has an accompanying ConvLSTMCell. Moreover, these lists can be stacked with varying parameters for each.

The inputs to our models have the shape (16, 15, 1, 96, 160), where the batch size is 16, the length of the input sequence is 15, the number of color channels is 1, and the image dimensions are 96 by 160 pixels. Our network as shown in Figure 5 consists of stacking 3 ConvLSTM layers, each with kernel size (5,5) and the number of filters increasing from 16 to 18 to 20. Finally, a Conv2D operation is performed to reduce the number of channels from 20 to 5, which is the desired sequence length for the prediction.

We trained our models for 10 epochs and used the Adam optimizer with a learning rate of 0.001. The best-performing epochs were measured by their validation loss and exported using a checkpoint system. For our loss function, we built a wrapper around the Kornia implementation of the `ssim_loss` [4] where we calculated the SSIM loss for each element of the batch.

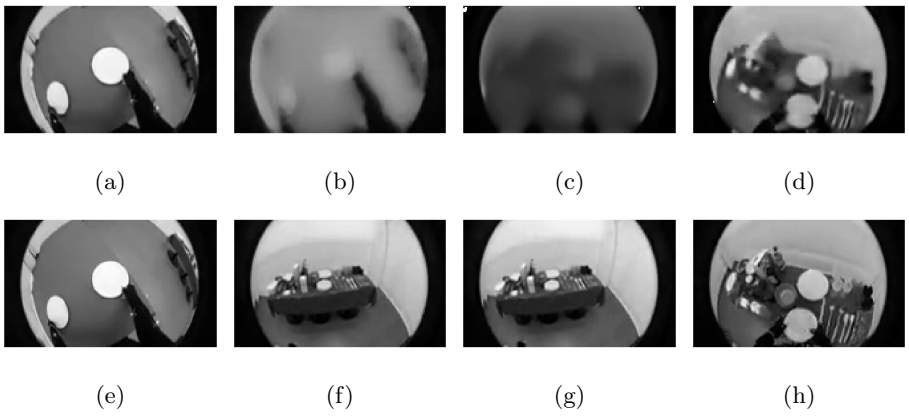
4.4 Postprocessing

During visual inspection of our predicted frames, we noticed white blocks that would appear sporadically in the black borders. These can be seen to some extent in the top left corner of Figure 6c or in the left middle of Figure 6d. We tried to programmatically remove these blocks, however, no attempt yielded an improvement in our scores. Consequently, we discarded the replacement of white blocks.

The only step in our postprocessing pipeline is to reverse the normalization by multiplying the data by 255.

4.5 Evaluation

After tuning the following parameters: history length, prediction length, amount of ConvLSTM layers, learning rate and filter per layer. The final submission achieved a SSIM of 0.61.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 6: Comparison between predicted frames and reference frames

Figure 6 compares four exemplary predicted video frames to the original frames. The frames chosen are the first frames and last frames of the forward and backward trained models respectively. This means frames (b) and (c) are a single frame apart and in the middle of the gap. This results in a visual cut, seeing as the two frames are generated by separate models without a shared context. The sliding window results in compounding errors which reduces the visual detail in the output. The prediction does not contain a lot of movement but does retain structural information. Especially the black borders to the left and right edges of the video are recreated faithfully.

5 Discussion and Future Work

The most influential point of discussion for the Bremen Big Data Challenge 2022 is the submission scoring based on SSIM. There seemed to be a problem with the challenges' evaluation script handling of the motion capture outlier value. It favored solutions that could accurately fill in gaps containing the outlier value. As a consequence, there was a miss-match in incentives. Spending more time on improving the prediction capabilities of the motion capture data without gaining leaderboard positions was not productive if this could be time spent on improving the video model.

One prospective change could be fixing the scoring for the motion capture prediction. In return, the model could be further optimized which may lead to more accurate models. Additionally, different architectures for both motion capture and video prediction could be examined in the future. One approach could utilize a transformer-based architecture. Transformers have gained a lot of popularity in recent years and have been adopted for a variety of different tasks, e.g. time series [6]. We believe that a transformer-based architecture could be employed to solve both problems based on their plethora of use cases. Another

potential approach for video prediction is the *Simpler yet Better Video Prediction* (SimVP) architecture. SimVP is a simple convolutional model [2] that was released after the challenge had ended. Due to the simplicity of the model, we believe that it might have been a candidate to solve this challenge.

6 Conclusion

The Bremen Big Data Challenge 2022 posed a difficult and thereby interesting problem. Our solution for the motion capture prediction has a big potential for improvement. Despite the achieved score of 1.0, a visual inspection of the predictions showed that they were not accurate. Since the evaluation is skewed towards submissions that can consistently recreate gaps containing the outlier value, substituting the scoring function for a more meaningful alternative could lead to better results. While ConvLSTM is a simple architecture we still managed to achieve a score of 0.61 for the video prediction by investing considerable time into fine-tuning. Considering the time constraints of the challenge and the difficulty of the problem, the achieved results are a good starting point for future developments. These findings demonstrated the usefulness of bilaterally filling the gaps in reducing sliding window errors.

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization (2016). https://doi.org/10.48550/ARXIV.1607.06450, `https://arxiv.org/abs/1607.06450`
2. Gao, Z., Tan, C., Wu, L., Li, S.Z.: Simvp: Simpler yet better video prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3170–3180 (June 2022)
3. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**, 1735–80 (12 1997). https://doi.org/10.1162/neco.1997.9.8.1735
4. Kornia: kornia.losses (2019), `https://kornia.readthedocs.io/en/latest/losses.html?highlight=ssim#kornia.losses.ssim_loss`
5. Palazzi, A.: ConvLSTM_pytorch (2022), `https://github.com/ndrplz/ConvLSTM_pytorch`
6. Papers With Code: Video prediction on moving mnist (2022), `https://paperswithcode.com/sota/video-prediction-on-moving-mnist`
7. Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.k., Woo, W.c.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1. p. 802–810. NIPS'15, MIT Press, Cambridge, MA, USA (2015)
8. Srivastava, N., Mansimov, E., Salakhutdinov, R.: Unsupervised learning of video representations using lstms. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. p. 843–852. ICML'15, JMLR.org (2015)
9. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004). https://doi.org/10.1109/TIP.2003.819861