Spiking ConvLSTM for Semantic Music Generation

Anonymous Submission

Abstract

Spiking neural networks received a lot of attention in a scientific literature due to low memory and energy consumption, which makes them suitable for edge computing and off-line deployment of AI models on low power devices. The use of spiking neural networks in a multimodal generative context was not yet explored and in this paper the focus in made on comparison of performance between classical and spiking versions of shallow Convolutional Long-Short Term Memory (ConvLSTM) network architecture. A new encoding strategy based on the system of graphs allowed dimensionality and data augmentation of one-dimensional sequential data, resulting in better generalization capacities of trained models. To the best of our knowledge, this is the first attempt of implementing spiking ConvLSTM model for semantic music generation task. We release the resources with the article (*upon paper acceptance*).

Introduction

Recent advances in gradient calculation in spiking neural networks with surrogate approaches (Kheradpisheh, Mirsadeghi and Masquelier 2022, Zhang et al. 2021) allowed amelioration of the learning efficiency of this type of networks, making them interesting for practical use. The integration of spiking neurons has been previously done with feedforward neural networks (She 2020), recurrent neural networks (Demirag et al. 2021, Kim and Sejnowski 2019), convolutional neural networks (Guan & Mo 2020) and belief neural networks (O'Connor et al. 2013). The angle of research focused mostly on classification tasks, however there are several examples of spiking neurons use in a generative adversarial network architecture (Molano-Mazon et al. 2018, Kotariya and Ganguly 2021, Rosenfeld, Simeone and Rajendran 2021). The generative prospective in a multimodal setting was not yet explored and is mostly related to the absence of benchmark data on generative abilities of spiking neural networks, along with a lack of systematic research measuring the differences of performance between classical and spiking neural networks in a generative setting. There were attempts to develop a benchmark dataset for spiking neural networks, however each time only one modality is covered (Cramer et al. 2022, Liu et al. 2016), therefore the data for multimodal research were not yet proposed.

In this article we propose a multimodal dataset with parallel data, referred further as seqGraph, and compare the performance of classical shallow convolutional Long Short-Term Memory network (ConvLSTM) with its spiking version (SConvLSTM). The data of seqGraph comprise 216 one-dimensional harmonic sequences (their analytical representation, which uses Roman and Arabic numerals to represent the chord fundamental and its inversion), their respective audio representation (generated with *music21* Python library for computational musicology, out of the initial harmonic sequences) and a 2D representation of chords, received via original encoding technique based on the system of graphs. The dimensionality augmentation allows to match the dimensionality of existing datasets and evaluate the network performance on data for which the benchmark results are known. We also apply a varying normalization term to the 2D partition of data, which results in considerable data augmentation (up to 2160 entries).

The models in this paper are trained on 2D partition of the seqGraph dataset, being compared with the existing Moving MNIST dataset and a custom selection of sequences of images from classical MNIST dataset, referred further as seqMNIST. We exploit the implementation of ConvLSTM model (Xingjian et al. 2015), modifying it to its shallow version, and build our shallow spiking ConvLSTM model.

The contribution of this paper consists in the following:

1. Exploration of the generative abilities of spiking neural networks;
2. Cross modal generation exploration (the use of one modality to generate another modality);
3. Multimodal data, which includes semantic, visual and audio information, as a tool for further multimodal research in spiking neural networks.
4. Original encoding method based on a frame theory, and the system of graphs, which has several advantages over classical 2D data, such as:
    1) Data augmentation with the use of varying normalization term;
    2) Reduced information density, which is better processed by spiking neural networks;
    3) Easy transition of unsupervised visual information to a labelled 1D sequences for inputs, targets and predictions, which facilitates multimodal and cross-modal research.

Encoding method

Methodology

The encoding strategy proposed in this paper is based on an original system of graphs in music harmony, which was build following the frame and graph theory by Marvin Minsky (1974). The system has been shown useful in building of mobile (Anonymous 2016), VR (Anonymous and Darkazanli 2020) and web-based graphical interfaces for learning music harmony, tested with success in several pedagogical experiments (Anonymous 2019a).

The system of graphs presents an analytical representation of functional correlations between chords, visualized as the degrees of the diatonic scale (Roman numerals), placed in a specific order. The placement order is defined by the possibility of connection between the chords and expresses the variation potential within a multitude of chord progressions. The slots are reserved for a specific type of information, where the slots reserved for the seventh chords cannot contain the information about triads (Figure 1). This limit comes from the theory of graphs, defining a graph as a graphical representation of frames, corresponding to a specific knowledge representation.

Figure 1: Unfilled graph structure (on the left) along with the graph filled with the information about three passing progressions, between the seventh chord of the tonic and its first inversion (on the right). The tonic seventh chords are inside the tops of the graph and three triads are inside the edges of the graph.

The graphs are organized in horizontal triads, as shown in Figure 2, and vertical triads (Figure 3). A harmonic sequence is therefore represented as a path within the system, connecting the nearest instances of the chord in the system of graphs.

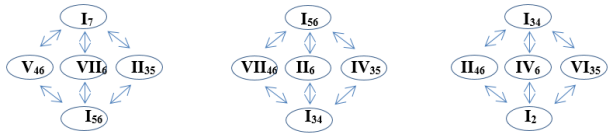


Figure 2: Horizontal triad regroups the graphs containing tops sharing the same degree. In this example, a horizontal triad embraces all possible passing progressions between the tonic seventh chord and its inversions.

Data conversion process

The value of the graph representation consists in a possibility of the encoding of semantic music data. Such encoding allows training and evaluation methods application, inherent to a visual domain. This way, the harmonic sequences features learning may be done with the 2D convolutional LSTM layers, discussed below. Another advantage of such encoding is the possibility of considerable data augmentation with the application of a small normalization term. Finally, the modality switch is useful in fostering multimodal studies in general and particularly within spiking neural networks.

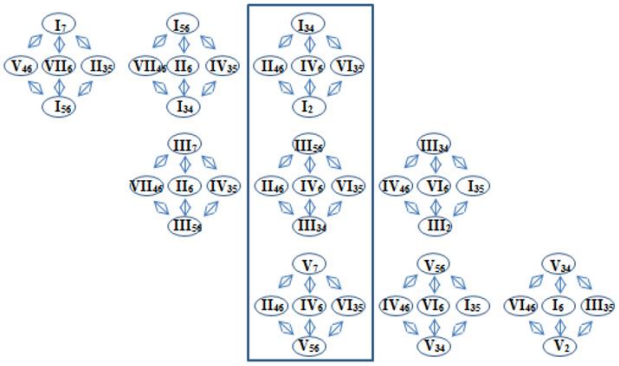


Figure 3: A vertical triad groups the graphs, sharing the same information inside edges.

To obtain the feature maps out of the arrays of harmonic sequences, the following transformation steps must be performed:

- Chords dictionary creation.
- Two-dimensional matrix creation using the graph system frame and replacing the chords with the values from previously created dictionary.
- Creation of 28x28 feature maps of harmonic paths in 2D space using harmonic sequences mapped to the chords dictionary in a progressive way: one sequence of 5 chords results into 5 matrices, gradually filled in with the chord values.
- Normalization of the matrices with a changing normalization term (in a range between 0.01 and 0.1).

Using this data conversion strategy, it was possible to obtain 2160 data entries out of the initial handcrafted 216 harmonic sequences.

To illustrate a described transition process, let us take a sequence of three chords II_7-VI_{46}-II_{56} and present it as a graph (Figure 4a), then replace a semantic chord designation with numerical values – indexes we chose to represent the chords (Figure 4b): we thus receive a 3x3 matrix. If we repeat the described procedure, but taking the dimensionality of the whole system, we receive a matrix ready to be processed by a neural network (49 chords of the system of graphs are giving precisely 27x21 matrix, however we added one column and seven rows of a padding filled with

zeros to facilitate computing of 2D convolutional operations).

After the data conversion is finished and each chord of the harmonic sequence is represented as a 28x28 tensor, we can visualize the matrices using *matplotlib* library. Let's visualize a harmonic sequence from our dataset, consisting of 5 chords: II_7-VI_{46}-II_{56}-V_{35}-I_{35}, in Figure 5.

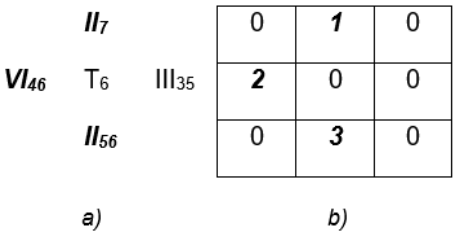
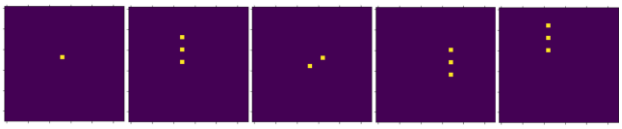


Figure 4: Harmonic sequence in a graph representation (a) and in a matrix representation (b).



1) II_7 2) VI_{46} 4) II_{56} 5) V_{35} 6) I_{35}

Figure 5: Converted harmonic sequence visualization.

The shape of the chord in presented visualization is defined by its position within the system of graphs – the chord II_7 can only start the passing progression sequence (appear only in a top of the graph), therefore it is represented as a single point, whereas II_{56} can start and finish the passing progression sequences and as a result, appearing in both tops of the graph within the same horizontal triad of graphs (therefore a single chord is represented with two dots). Finally, the triadic structures (VI_{46}, V_{35} and I_{35}) appear in the graph vertices only and are common for a given vertical triad of graphs, which produces three positions and, consequently, three dots to appear in a matrix visualization.

Experiment

The new data encoding scheme has been tested in an unsupervised setting, using classical and spiking neural networks. The 2D data required implementation of an architecture capable to process sequences of frames, therefore, a hybrid ConvLSTM architecture was chosen. In ConvLSTM each element of the sequence passes through a convolutional layer followed by LSTM layer. Previous experiment has shown that this type of architecture captures well two-dimensional sequential data (Anonymous 2019b) and will be exploited in current experiment using novel data encoding scheme.

The first stage of the experiment focused on comparison of ConvLSTM model trained on equivalent 2D data, with and without augmentation. The result received from the models trained on custom data is further compared with the models trained on reduced versions of Moving MNIST dataset (Srivastava et al. 2015), in respect of the quantity of entries from a custom 2D dataset in its original and augmented forms (216 and 2160 sets). The Moving MNIST dataset was used to train the very first implementation of ConvLSTM model and allows to account for the difference between original deep model and its shallow version, as well as the number of data points, considerably reduced in our experiment. The very nature of the data in the Moving MNIST dataset, consisting of the sequence of 2D frames, representing two moving digits in each data entry, doesn't fit the diversity of our data, since each 2D frame of the seqGraph dataset represents a distinct chord, which has a different shape than other frames of the sequence. Besides, the chords in each sequence are distinct (the repetition is only allowed for a target), where four non-repeating chords in an input sequence are followed either by non-repeating target or a target, similar to one of the chords from the input sequence. Therefore, we selected distinct digits from a MNIST dataset to form 216 sequences, which matches the degree of diversity proposed in seqGraph.

The second stage of the experiment compares performance of spiking versions of neural networks, trained on the data from the first stage. Therefore, the augmented and non-augmented seqGraph, Moving MNIST and seqMNIST datasets were used to train SConvLSTM models. All neural networks were developed using Pytorch framework with additional use of *snnTorch* Python package for SConvLSTM implementation, which allows performing gradient-based learning with spiking neural networks.

Neural Networks Architecture

The shallow ConvLSTM model architecture comprised one LSTM layer, followed by 3D batch normalization (third dimensionality was necessary to account for sequential nature of data) and convolutional 2D layer. The tangent activation function was used for gated mechanism inside ConvLSTM cell, while the sigmoid activation function was applied to the output of the final convolutional 2D layer, generating prediction on the input batch. The entire model architecture is shown in Figure 6.

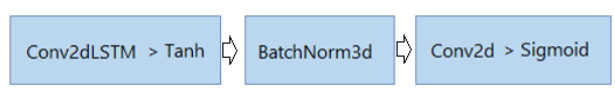


Figure 6: ConvLSTM model architecture.

The spiking ConvLSTM model architecture comprised the same set of layers, with a difference consisting in SConvLSTM cell integration. The implementation of the cell was taken from *snnTorch* API and unlike ConvLSTM

cell, it uses sigmoid activation function for the input, forget and output gates, while preserving tangents for the cell state. The Straight Through Estimator surrogate gradient descend was used to learn the weights. The models were trained using membrane activation potential, representing continuous values, which allowed application of the same loss function for both classical and spiking neural networks.

Training Setup

The similar hyperparameters were applied to all the models, such as mini-batch size (15 sequences per batch), optimizer (Adam), learning rate (1e-4), sequence length (5 chords in a sequence), number of epochs (100 epochs), manual seed (42). The loss function for ConvLSTM and SConvLSTM models was binary cross entropy. Since we used non-neuromorphic data, there was no need for additional time steps parameter, which is required for processing time distributed data, therefore this parameter was set to zero for SConvLSTM models. The measurement of the training efficiency was done using mean value from 5 k-folds.

The initial one-dimensional custom data from seqGraph dataset consisted of 216 sequences of 5 chords each, where 4 chords served as an input and the 5th chord was treated as a target. In two-dimensional data split, each chord of the sequence was converted into a 28x28 matrix form, where the first 4 matrices represented the input features, while the last matrix represented a target. This way, four input chords conditioned the prediction of the final chord of the sequence.

The data from Moving MNIST dataset was resized from 64x64 to 28x28, the sequence length was reduced from 20 frames to 5 frames and the initial 10.000 entries were reduced to 216 and 2160 respectively, in order to match the size and structure of the custom seqGraph dataset.

Finally, as was mentioned above, the seqMNIST was formed by selecting distinct digits for 216 sequences of 5 images each, were 4 first images served as an input and the last 5th image represented a target to predict.

Results

First stage. The result of the first stage of experiment is presented in Table 1, showing the loss values for train and validation (val.) splits in the beginning of training (train start, val. start) and after the training completion (train end, val. end). The data concerns five ConvLSTM models trained on non-augmented seqGraph (seqGraph 216), 216 entries from Moving MNIST (M. MNIST 216), seqMNIST (seqMNIST 216), augmented seqGraph (seqGraph 2160) and 2160 entries from Moving MNIST datasets (M. MNIST 2160) respectively.

ConvLSTM models trained on three datasets with the same amount of data (216) show very different result in terms of the training efficiency. The loss in the beginning of training is almost identical for all three models, however, in the end of training, seqGraph gave the lowest train and validation loss values. The seqMNIST dataset, in opposite, resulted in the highest loss values, therefore, we can conclude that seqMNIST data were the hardest to process by the model.

The result is confirmed by learning curves, shown in figures 7-9, where the best agreement between train and validation loss curves were achieved with seqGraph dataset (Figure 7). The distance between train and validation loss curves appears for the model trained on Moving MNIST (Figure 8) and augments for the model trained on seqMNIST (Figure 9).

Model	Train start	Val. start	Train end	Val. end
ConvLSTM seqGraph 216	401.8	109.3	8.72	2.18
ConvLSTM M. MNIST 216	413.1	109.3	61.45	15.44
ConvLSTM seqMNIST 216	430.8	109.2	176.4	44.22
ConvLSTM seqGraph 2160	142.7	8.21	**4.02**	**1.01**
ConvLSTM M. MNIST 2160	206.0	25.54	53.38	13.39

Table 1: Loss values comparison for ConvLSTM models.

The images generated with the model trained on seqMNIST have undistinguishable shape (Fig. 12), whereas the images generated with the model trained on seqGraph, although blurred, contain quite distinct shapes (Fig. 10). The images generated with the model trained on 216 entries from Moving MNIST are the clearest (Fig. 11), however, the model was trained to generate a position of the target digits, not a distinct digit. Therefore, the charge of generating the shapes of digits is lifted by the repetition of the same shapes in a sequence of 4 input images, provided at the training and inference phases.

The table data for the ConvLSTM models trained on 2160 entries from augmented seqGraph and Moving MNIST show a significant difference of the training efficiency, expressed in much lower training and validation losses for seqGraph in comparison to Moving MNIST. This allows suggesting that dimensionality augmentation of data helps with the generalization capacities of the model.

The learning curves (Figures 13-14) confirm that hypothesis, as the training and validation curves of the ConvLSTM model trained on seqGraph have the best agreement, while the same distance between train and validation loss curves appears for the model trained on 2160 entries of Moving MNIST, as it was the case for the model trained on 216 entries of the same dataset.

The generated images of these two models are shown in Figures 15-16.

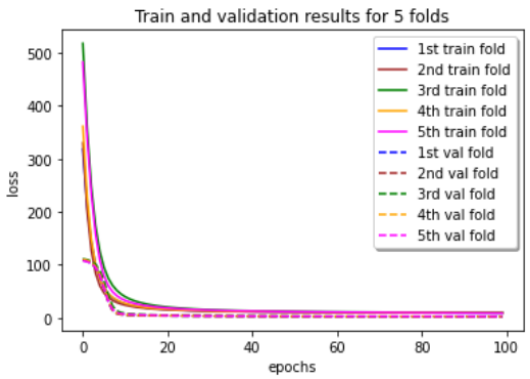
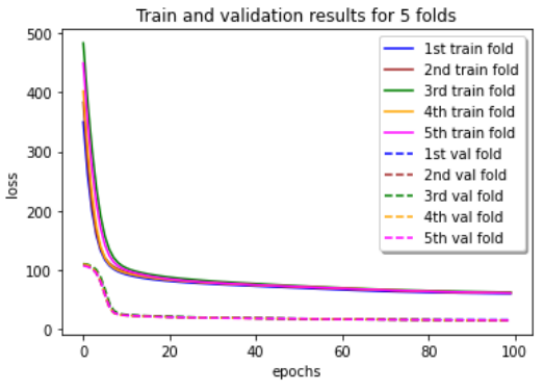


Figure 7: Train and validation learning curves for the ConvLSTM model trained on seqGraph (216 entries).



<

Figure 8: Train and validation learning curves for the ConvLSTM model trained on 216 entries from Moving MNIST.

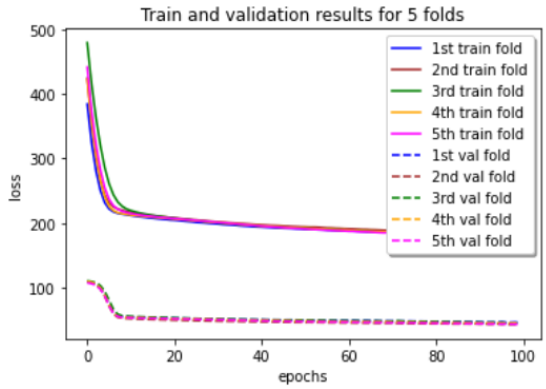


Figure 9: Train and validation learning curves for the ConvLSTM model trained on seqMNIST (216 entries).

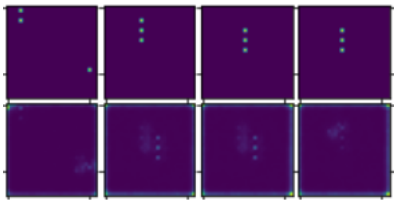


Figure 10: Comparison of generated samples (bottom raw) with targets (upper row). Inference is done with the ConvLSTM model trained on 216 entries from seqGraph.

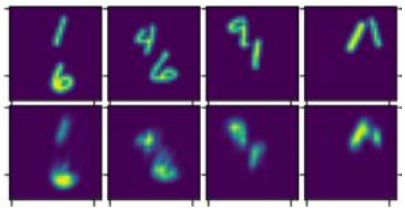


Figure 11: Comparison of generated samples (bottom raw) with targets (upper row). Inference is done with the ConvLSTM model trained on 216 entries from Moving MNIST.

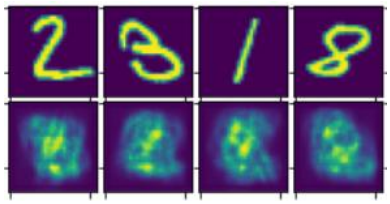


Figure 12: Comparison of generated samples (bottom raw) with targets (upper row). Inference is done with the ConvLSTM model trained on seqMNIST (216 entries).

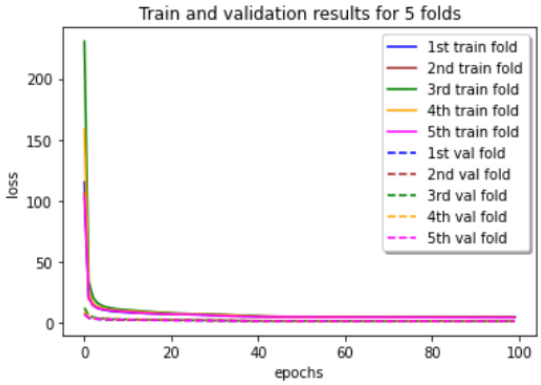


Figure 13: Train and validation learning curves for the ConvLSTM model trained on augmented seqGraph (2160 entries).

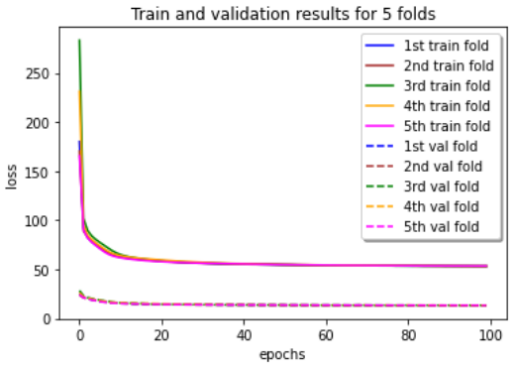


Figure 14: Train and validation learning curves for the ConvLSTM model trained on 2160 entries from Moving MNIST.

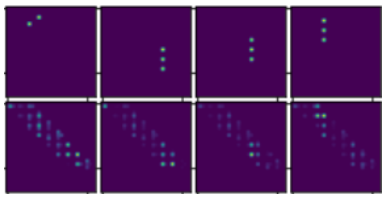


Figure 15: Comparison of generated samples (bottom raw) with targets (upper row). Inference is done with the ConvLSTM model trained on augmented seqGraph (2160 entries).

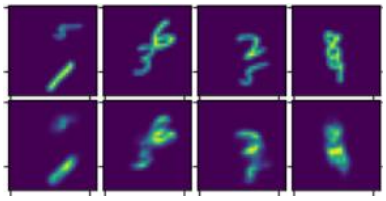


Figure 16: Comparison of generated samples (bottom raw) with targets (upper row). Inference is done with the ConvLSTM model trained on 2160 entries from Moving MNIST.

Second stage. The result of the second stage of the experiment is presented in Table 2. As in case with ConvLSTM models, the SConvLSTM models trained on seqGraph data (augmented and non-augmented versions) show the lowest training and validation losses comparing to SConvLSTM models trained on Moving MNIST or seqMNIST datasets. The similar tendency is preserved for learning curves, for the group of datasets containing 216 entries (Figures 17-19), as well as for the datasets containing 2160 entries (Figures 23-24).

Although the loss values overall are slightly higher for the SConvLSTM models, the quality of generated images (Figures 20-22 and 25-26) is better than the generations made by ConvLSTM models. The observation is particularly truthful for images generated with SConvLSTM trained on 2160 entries of the augmented seqGraph dataset (Figure 25), which are much cleaner and exact in terms of the pixel position, than the images generated by the non-spiking counterpart trained on the same data (Figure 15). This allows suggesting that spiking ConvLSTM model architecture better processes sparse data.

Model	Train start	Val. start	Train end	Val. end
SConvLSTM seqGraph 216	416.1	108.1	8.67	2.17
SConvLSTM M. MNIST 216	446.6	108.1	63.79	16.04
SConvLSTM seqMNIST 216	450.3	108.2	184.5	46.30
SConvLSTM seqGraph 2160	89.29	6.01	**5.88**	**1.49**
SConvLSTM M. MNIST 2160	166.3	24.56	53.65	13.42

Table 2. Loss values comparison for spiking versions ConvLSTM models.

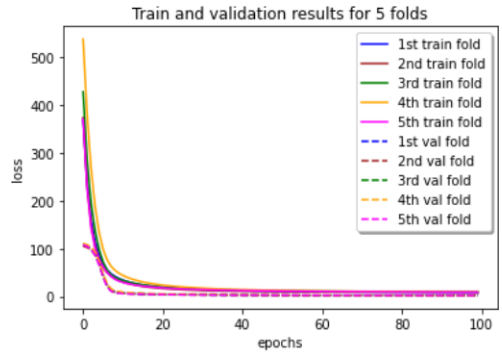


Figure 17: Train and validation learning curves for the SConvLSTM model trained on seqGraph (216 entries).

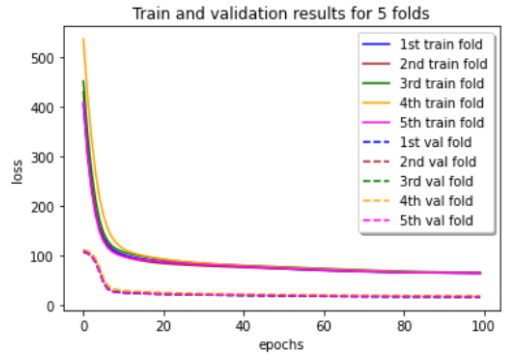


Figure 18: Train and validation learning curves for the SConvLSTM model trained on 216 entries from Moving MNIST.

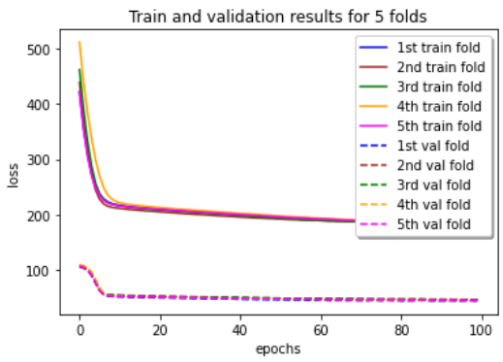


Figure 19: Train and validation learning curves for the SConvLSTM model trained on seqMNIST (216 entries).

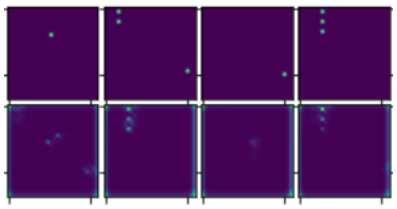


Figure 20: Comparison of generated samples (bottom raw) with targets (upper row). Inference is done with the SConvLSTM model trained on seqGraph (216 entries).

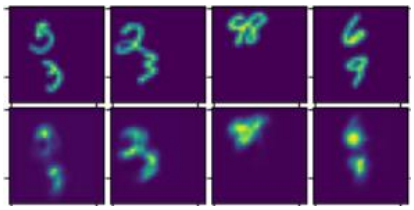


Figure 21: Comparison of generated samples (bottom raw) with targets (upper row). Inference is done with the SConvLSTM model trained on 216 entries from Moving MNIST.

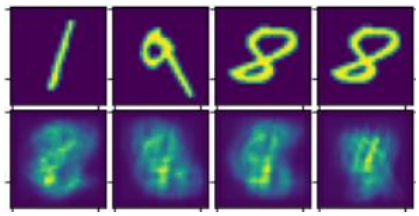


Figure 22: Comparison of generated samples (bottom raw) with targets (upper row). Inference is done with the SConvLSTM model trained on seqMNIST (216 entries).

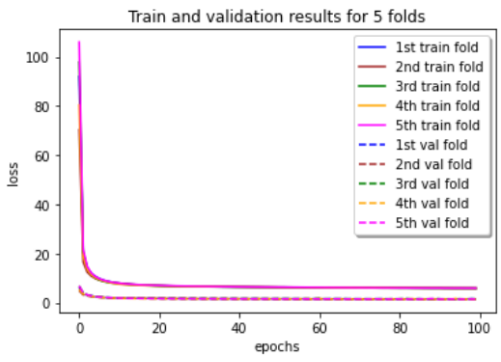


Figure 23: Train and validation learning curves for the SConvLSTM model trained on augmented seqGraph (2160 entries).

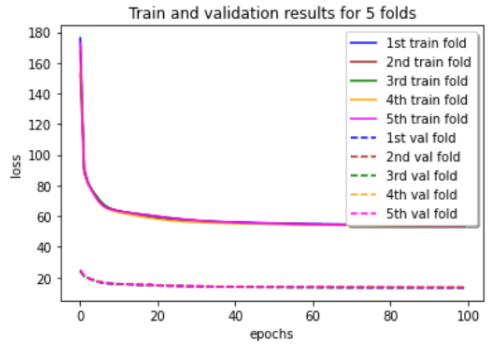


Figure 24: Train and validation learning curves for the SConvLSTM model trained on 2160 entries from Moving MNIST.

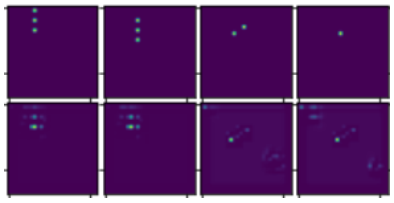


Figure 25: Comparison of generated samples (bottom raw) with targets (upper row). Inference is done with the SConvLSTM model trained on augmented seqGraph (2160 entries).

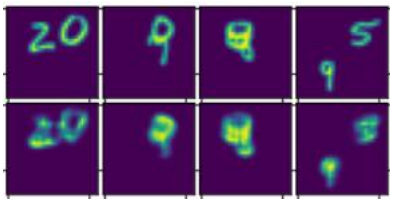


Figure 26: Comparison of generated samples (bottom raw) with targets (upper row). Inference is done with the

Ablation studies

We tested images generation without batch normalization and without application of Sigmoid activation to the last frame. Thus, without batch normalization layer no visual information was generated, although the loss values indicated that the networks trains well. The ablation of Sigmoid activation to the last frame resulted in blurriness augmentation in generated images, compared to the images generated with a full version of the network.

We also tried adding spiking layers to the network, however the training was unpredictable and the generated images lacked pixel information. The possible cause of the unpredictable training might be the dead neurons problem, to which the spiking neural networks are particularly sensible. A possible solution of the problem can be an artificial simulation of neurons that did not fire (Kheradpisheh, Mirsadeghi and Masquelier 2021), however the solution implementation is beyond the scope of this experiment.

Conclusions

The article proposed a shallow spiking ConvLSTM architecture, the performance of which was compared to a shallow ConvLSTM architecture. The network was tested with a custom dataset seqGraph, sequences of digits from MNIST and sequences of frames from Moving MNIST datasets. It has been shown that sparse nature of seqGraph data consistently gives better results in terms of the training efficiency, agreement of training and validation learning curves and the clarity of generated images for spiking and classical models, with the best result received for spiking version of ConvLSTM trained on augmented version of seqGraph. The multimodal nature of constructed data allows further research in multimodal generation context.

References

Anonymous, 2016. The System of Graphs in Music Harmony: A user interface for mobile learning game development. In Proceedings of the Electronic Visualisations and the Art. London: British Computer Society, pp. 193-194.

Anonymous, 2019a. Contemporary methods of functional harmony teaching in a high school context. In Proceedings of the Electronic Imaging and the Visual Arts. Florence: Florence University Press, pp. 142-150.

Anonymous, 2019b. Structural harmony method in the context of deep learning on example of music by Valentyn Sylvestrov and Philipp Glass. In Proceedings of the Electronic Visalisations and the Art. London: British Computer Society. pp. 318–320.

Anonymous and Darkazanli, S. 2020. Graphs in Harmony Learning: AI assisted VR application. In Proceedings of the Electronic Visalisations and the Art. London: British Computer Society, pp. 104-105.

Cramer, B.; Stradmann, Y.; Schemmel, J. and Zenke, F. 2022. The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks, *IEEE Transactions on Neural Networks and Learning Systems* 33(7): 2744-2757, doi:10.1109/TNNLS.2020.3044364.

Demirag, Y.; Frenkel, C.; Payvand, M. and Indiveri, G. 2021. Online Training of Spiking Recurrent Neural Networks with Phase-Change Memory Synapses. arXiv:2108.01804.

Guan, X. and Mo, L. 2020. Unsupervised conditional reflex learning based on convolutional spiking neural network and reward modulation. *IEEE Access*, 8: 17673-17690, doi: 10.1109/ACCESS.2020.2968240.

Kheradpisheh, S.R.; Mirsadeghi, M.; and Masquelier, T. 2022. BS4NN: Binarized Spiking Neural Networks with Temporal Coding and Learning, *Neural Processing Letters*. 54: 1255–1273, doi:10.1007/s11063-021-10680-x.

Kim, R.; Li, Y. and Sejnowski, T. J. 2019. Simple framework for constructing functional spiking recurrent neural networks. In Proceedings of the national academy of sciences, 116(45), pp. 22811-22820, doi: 10.1073/pnas.1905926116.

Kotariya, V. and Ganguly, U. 2021. Spiking-GAN: A Spiking Generative Adversarial Network Using Time-To-First-Spike Coding. arXiv:2106.15420.

Liu, Q.; Pineda-García, G.; Stromatias, E.; Serrano-Gotarredona, T. and Furber, S. B. 2016. Benchmarking spike-based visual recognition: a dataset and evaluation, *Frontiers in neuroscience* 10(496).

Minsky, M. 1974. *A framework for representing knowledge*. Massachusetts: Massachusetts Institute of Technology.

Molano-Mazon, M.; Onken, A.; Piasini, E. and Panzeri, S. 2018. Synthesizing realistic neural population activity patterns using generative adversarial networks. In Proceedings of ICRL 2018, doi: 10.48550/arXiv.1803.00338.

O'Connor, P.; Neil, D.; Liu, S. C.; Delbruck, T., and Pfeiffer, M. 2013. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in neuroscience (7)*, p. 178.

Rosenfeld, B.; Simeone, O. and Rajendran, B. 2021. Spiking Generative Adversarial Networks with a Neural Network Discriminator: Local Training, Bayesian Models, and Continual Meta-Learning. arXiv:2111.01750.

She, X.; Saha, P.; Kim, D.; Long, Y. and Mukhopadhyay, S. 2020. Safe-DNN: A deep neural network with spike assisted feature extraction for noise robust inference. In 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1-8, doi: 10.1109/IJCNN48605.2020.9207274.

Srivastava, N.; Mansimov E. and Salakhudinov, R. 2015. Unsupervised learning of video representations using LSTM. In International conference on machine learning. pp. 802–810.

Xingjian, S. H. I.; Chen, Z.; Wang, H.; Yeung, D. Y.; Wong, W. K. and Woo, W. C. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Advances in neural information processing systems. pp. 802–810.

Zhang, M.; Wang, J.; Amornpaisannon, B.; Zhang, Z.; Miriyala, V.; Belatreche, A.; Qu, H.; Wu, J.; Chua, Y.; Carlson, T.E. and Li, H. 2021. Rectified Linear Postsynaptic Potential Function for Backpropagation in Deep Spiking Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5): 1947-1958, doi: 10.1109/TNNLS.2021.3110991.