Advanced International Journal of Material Science and Engineering
Volume.7, Number 1; January-March, 2022;
ISSN: 2837-3928 | Impact Factor: 7.19
http://zapjournals.com/Journals/index.php/aijmse
Published By: Zendo Academic Publishing

TEMPORAL EXTRAPOLATION IN VIDEOS: DEEP LEARNING APPROACHES

Farhan Rahman Chowdhury[1]

Article Info

Keywords: artificial intelligence, machine learning, deep learning, representation learning, neural networks, future frame prediction

Abstract

The field of artificial intelligence (AI) has long pursued the goal of creating machines that can replicate human-like thinking and behavior. While early AI focused on solving complex problems that were difficult for humans, it became evident that tasks requiring intuition, such as image recognition or video understanding, posed significant challenges for traditional algorithmic approaches. This led to the emergence of machine learning (ML), where knowledge is acquired by extracting patterns from raw data. However, ML still requires manual feature selection and does not effectively handle multi-dimensional data like images and videos.

To address these limitations, representation learning and deep learning have been introduced. Representation learning aims to automatically construct meaningful representations of data, while deep learning leverages artificial neural networks (ANNs) inspired by the human brain to learn hierarchies of representations. The rise of computational power has enabled the development of deep neural networks that can handle increasingly complex tasks. Deep learning approaches have been applied to video analysis, including action recognition, video classification, and optical flow prediction, but they often require large amounts of labeled data, which is time-consuming and limits their applicability.

This paper makes several contributions to the field. Firstly, it provides an extensive overview of existing deep learning approaches for future frame prediction in videos. Secondly, it presents a novel neural network architecture that combines batch normalization, convolutional LSTM, and scheduled sampling to improve the training of recurrent models. This architecture achieves significantly reduced prediction errors compared to state-of-the-art models on the

[1] Daffodil International University, Dhaka, Bangladesh

Moving MNIST dataset. Moreover, the paper makes all TensorFlow implementations, including the specialized components and evaluation metrics, freely available to the research community. Additionally, a lightweight, high-level, open-source framework for TensorFlow is introduced, simplifying the development of deep learning applications by providing abstractions for common tasks.

Introduction

Since the classical era, people have dreamed of inventing machines that can act and think like humans. This opened the field of artificial intelligence (AI), which is still an active research topic and is used in many practical applications. The main focus of AI in the early days was to solve problems that are hard to solve for humans, such as finding the shortest path to an arbitrary destination using the well-known Dijkstra algorithm. Ironically, it turned out that tasks which can be solved by humans using pure intuition are extremely hard for computers to solve. As an example, it is hard or even impossible to write a program from scratch that can detect objects in pictures, recognize words in spoken text, or describe the events in a video scene. The reason is that classical computer programs in contrast have to be algorithmically expressed as a sequence of commands or a list of mathematical rules [1]. But it is quite tough to apply this to multi-dimensional data such as pictures or videos that consists of an incoherent set of pixels including different color channels with a lot of noise and countless possibilities. Humans handle this kind of data differently. They learn to recognize objects by experi- ence and implicitly build hierarchies of relationships in their mind. This basic principle opened a new subfield, known as machine learning (ML). It covers a methodology where knowledge is acquired by extracting patterns from raw data and consequently allows one to make reasonable decisions [1]. But while this can cover many previously unsolvable problems, it requires that one can tell which features we would like to investigate, for instance, to build a decision tree out of it. Coming back to our previous example, this is still hard to be applied to images or video data where it is known which features we are actually looking for, but still cannot formally describe how these are represented. A field that deals with this issue are called representation learning, which tries to automatically build the representation by itself.

Having just a high-level representation might still not be enough. To break down the problem, artificial neural networks (ANN) have been introduced. They are biologically inspired by the structure of the human brain [2] and can be trained to learn hierarchies of representations. For object recognition on images, one can think of edges that are detected on a very low level, which will be further composed of curves or shapes. Furthermore, these simple structures might be compounded in a specific Way, so that the neural network can identify distinct complex objects in it. The rise of computational power allows us to create networks that are even deeper and therefore learn more and more complex representations hierarchies. This principle caused its contemporary name, known as deep learning. Early deep learning approaches dealing with video data or simple image sequences address problems like human action recognition [3], [4], [5] or video classification [6]. Another example is optical flow prediction [7] to detects the visual flow from one frame to the next. Most of these approaches require lots of labeled data to be able to train a network. The effortful labeling process and thus the resulting low availability of such data might be the main reason why this topic has not been covered that well so far. On the contrary, online services like YouTube provide a seemingly endless, but unlabeled source of videos to learn from.

This paper consists of several contributions. Firstly, it provides a dense overview of existing deep learning approaches that deal with the problem of future frame prediction in videos. Secondly, it presents a neural

network architecture that combines modern practices like batch normalization with a novel convolutional LSTM implementation and scheduled sampling to improve the training of recurrent models. Thereby, it can cut in half the prediction error of other state-of-the-art models in the Moving MNIST dataset. Last but not least, all TensorFlow implementations are freely available to the research community, including the scheduled sampling and batch normalization enabled convolutional recurrent cell, as well as several metrics and loss functions to measure perceptual image similarity at training time. Furthermore, a lightweight, high-level, and open-source framework for TensorFlow is contributed to that can radically reduce the boilerplate code of deep learning applications. This is facilitated by providing an abstraction for many recurring or complex tasks that have to be faced while building and training neural network models.

1 Related work

1.1 Neural network approach

The first approaches to predict future frames in an image sequence were made in [2] and its follow-up publication [8]. Probably due to the lower computational power and the weak development of CNNs at that time, they tried to perform single-frame prediction based on an artificial neural network. They performed several preprocessing steps to train such a model using image data. Firstly, the image data was partitioned by the R, G, and B color channels into three parts. Next, the dimension of data was reduced from the order of 104 to 100 in each part using techniques like principal component analysis (PCA). The final training and inference were then accomplished on three separate neural networks of the same architecture for each color channel. After each prediction, the PCA process was inverted to obtain the initial dimensionality, as well as all three outputs were combined to obtain the final image.

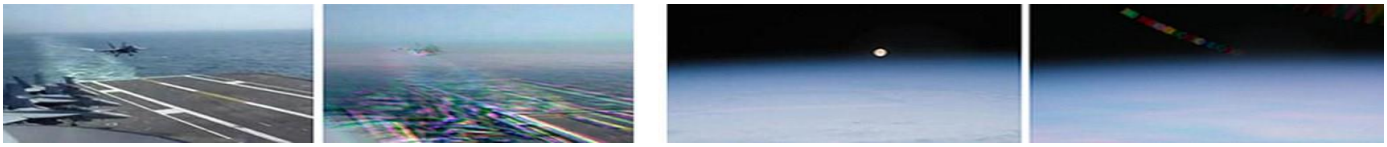

Fig. 1. Single frame predictions using an ANN model with two hidden layers. Left: ground truth target frame. Right: generated prediction [2]

The loss during the training process is measured in terms of the MS-SSIM index to optimize the network to preserve the luminance, contrast, and structure of the image. Since the data of the used Fighter and NASA datasets have a high image size, applying the multi-scale version of SSIM is a reasonable choice. When we take a look at the presented prediction results in Figure 1, it can be seen that the network more or less averages over the input sequence. This effect is visible in Figure 1b, where the movement of the moon from the top-left towards the earth's horizon is predicted like being composed of previous moon positions. As a result, this simple architecture does not sufficiently capture the temporal correlations of the input data.

1.2 LSTM encoder-decoder predictor model

A huge step forward was made when the recurrent encoder-decoder framework was applied in [9] to perform unsupervised learning of video representations. The fact that the same operation should be applied at each time step to produce the next state was their key idea to use this framework in that context. They presented an LSTM autoencoder model that is trained to reconstruct an entire input sequence of about ten image frames.

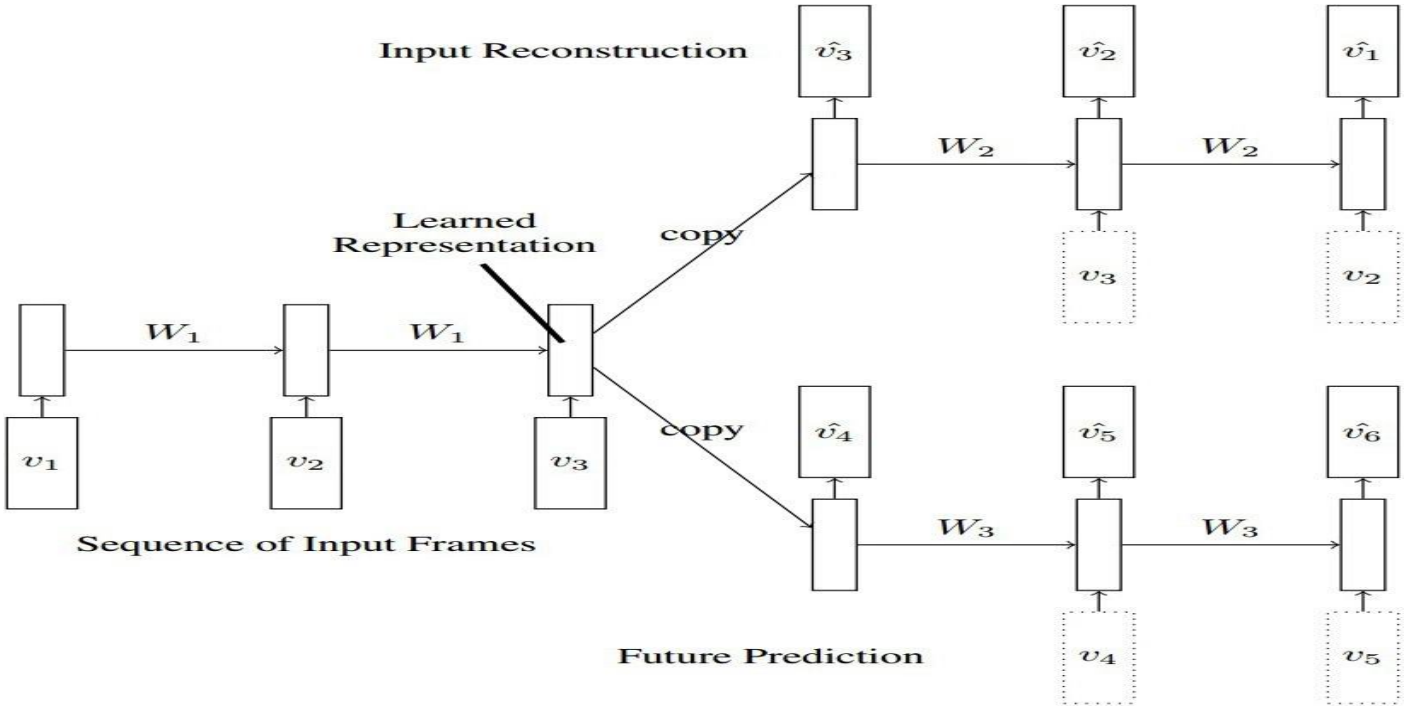

Fig. 2. The composite LSTM autoencoder model. The top branch reconstructs the input sequence backward in time, while the bottom branch performs frame predictions forward in time [9]

1.3 Convolutional LSTM encoding-forecasting model

The previously described model was extended in [10] with the general goal to develop a deep learning approach for precipitation nowcasting. But to moderate the tremendous redundancy of spatial correlations in standard FC-LSTM cells, they invented a modified version of LSTM that features a convolutional structure for both input-to-state and state-to-state transitions. To put it in a nutshell, they exchanged each matrix multiplication by a convolution operation whereby the internal states become three-dimensional tensors and preserve spatial information. These convolutional LSTM (ConvLSTM) cells are then used in the same decoder-encoder framework as before as illustrated in Figure 3. In the bottom line, these cells can capture spatiotemporal properties of the data much better than FC-LSTM cells and have shown to outperform them with even containing way fewer model parameters.

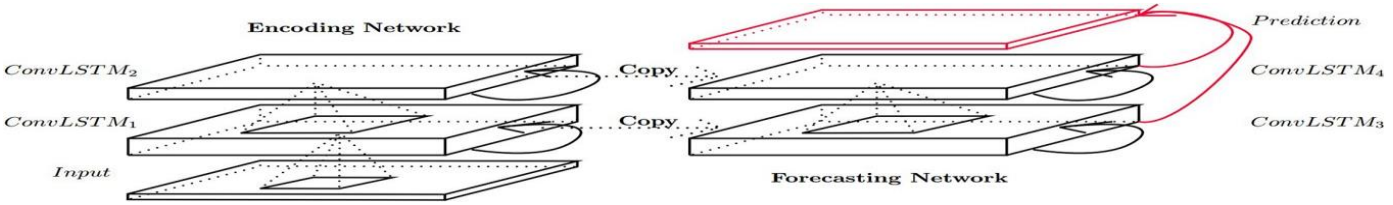

Fig. 3. The ConvLSTM encoding-forecasting model that was used in the paper in the context of frame prediction and precipitation nowcasting. [10]

1.4 Spatio-temporal video autoencoder model

A second work using ConvLSTM cells was published in [11]. It is based on the fundamental idea that a video autoencoder should differ from spatial autoencoders by being able to encode the significant differences instead of the entire video sequence by heart. With such an encoding at hand, neural networks should be able to learn the generation process of the future for any given frame. Compared to both previous models, this one does not rely on the standard recurrent decoder-encoder framework. Instead, it uses convolutional layers followed by a ConvLSTM encoder to produce a dense transformation map as its learned representation. Afterward, a semi-

detached optical flow module serves as a temporal decoder based on the learned transformation map. The estimated optical flow is then applied to the current image to generate the next frame using a convolutional decoder.

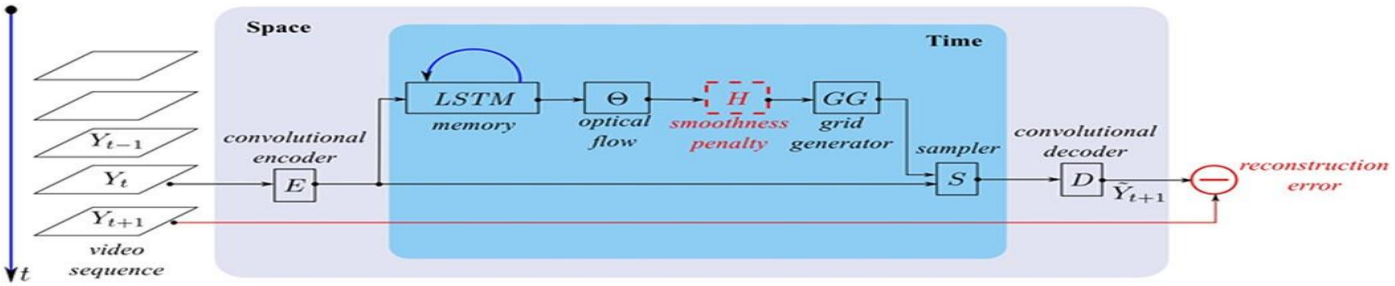

Fig. 4. The spatio-temporal video autoencoder model with its components that are separated by the space and time domain [11]

2 Approach and experiments

The model is trained on a synthetic dataset of black and white images with flying handwritten digits. The Moving MNIST has been introduced in [9] and applied in the context of video frame prediction. Since then, it was used several times in different follow-up works like [11] or [10].

2.1 Characteristics and data generation for MNIST dataset

In the proposed setting, each sequence consists of 20 image frames at a size of 64 × 64 with two random moving digits from the MNIST dataset in it. One major advantage of this simple dataset is that it exhibits a nearly unlimited size because it can be generated on the fly. When training a model, two random digits are therefore randomly chosen from the first 55,000 digits of the training set and placed on any location of the first image patch. For the generation of subsequent frames, a velocity is assigned to each digit, whose direction is chosen uniformly from a unit circle. Further, the simple physical rule is applied that the angle of incidence is equal to the angle of reflection when any digit at a size of 28 × 28 touches the wall. This enables other interesting properties of the dataset, such as basic dynamics due to having to predict the right trajectory after bouncing off a wall, as well as multiple occlusion effects of overlapping digits. Consequently, even though the generation process of the dataset is that simple, it is hard for a model to generate accurate predictions in the test set without learning a representation that encodes the internal motion of the system [10, p. 6]. Last but not least, having a simpler dataset at hand allows us to gain a better understanding of the model's behavior concerning its hyperparameters. Especially in consideration of the very long training time when more complex or even natural videos are used.

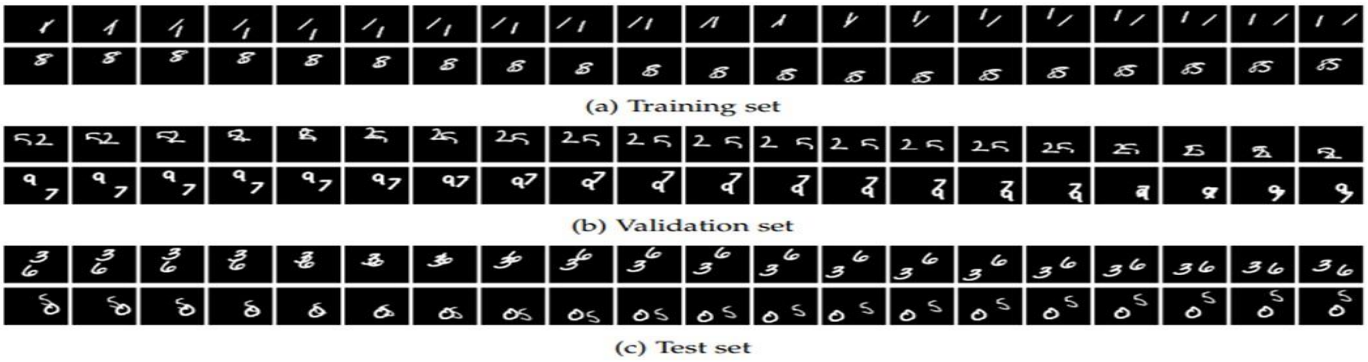

Fig. 5. Randomly chosen samples of generated image sequences of size 64 × 64 from the moving MNIST dataset

The generation procedure of the validation set is equal to the previously described process to create the training data, but with the difference that the last 5000 digits of the original MNIST training split are used. On the contrary, the test set is not generated by using the MNIST test split. Instead, the pre-generated test set of [11] that contains exactly 20 frames long sequences with 10,000 examples has been used. In this way, more comparable results can be achieved with at least one competing model. A random sample from each of these splits is presented in Figure 5.

2.2 Data preprocessing

Since the original pixel values of the MNIST dataset are in a range of [0, 255], a simple rescaling to [0, 1] is performed to normalize the data. Further, subtracting the mean pixel values has been considered as well because gray-scale images have the stationarity property. But because this is not often done in practice when the MNIST dataset is used [12] and no noticeable improvements could be seen when applying it, caused that mean subtraction is not applied while preprocessing the data. Moreover, one can believe that since most parts of any image are black (and therefore zero), further processing of the data is not necessarily required. Additionally, instead of feeding the model with continuous floating values in the normalized range, binary pixel values are used only. This decision results from the use of binary cross-entropy as the main loss function for this dataset, which has shown to be the favorable choice for image-generating models with MNIST [9], [10]. Therefore, every pixel p is set to zero if $p < 0.5$, and a value of one is assigned to all other pixels. Moreover, the sigmoid activation function is used in the output layer to support the saturation of all pixels into either zero or one.

2.3 Characteristics and data generation for MsPacman dataset

To accommodate the request of assessing our model on more complex data compared to the previously presented dataset, but which still has negligible dynamics compared to natural clips, a second data collection is used that consists of video game recordings. The MsPacman dataset has been used in an independent TensorFlow implementation of [13] for adversarial video generation and future frame prediction. It contains several interesting dynamics and properties that make it a reasonable choice to be considered in the evaluation of our model.

Characteristics

This video game dataset contains about half a million single images at 160 × 210. These images can be grouped into 517 game recordings for the training set and 51 recordings for the test set. Each recording has a variable length and ranges from about 500 to 1500 frames per sequence. The last 51 sequences of the training data are taken for the validation set to have it roughly the same size as the testing set. Thus, it ends up with a total number of 418,287 frames for training, 45,007 images for validation, and 46,380 images for testing. The action inside each recording acts in a closed world with various game rules that the network has to understand. Just to name some examples, all objects follow the path between the walls of the game world, with the two exceptions that ghosts in the center might exit the cave through the top barrier, as well as that Pacman can teleport himself by leaving the world using its left or right exit. Further, the game's main character opens and closes its mouth, and can eat the distributed dots, fruits, or blue ghosts.

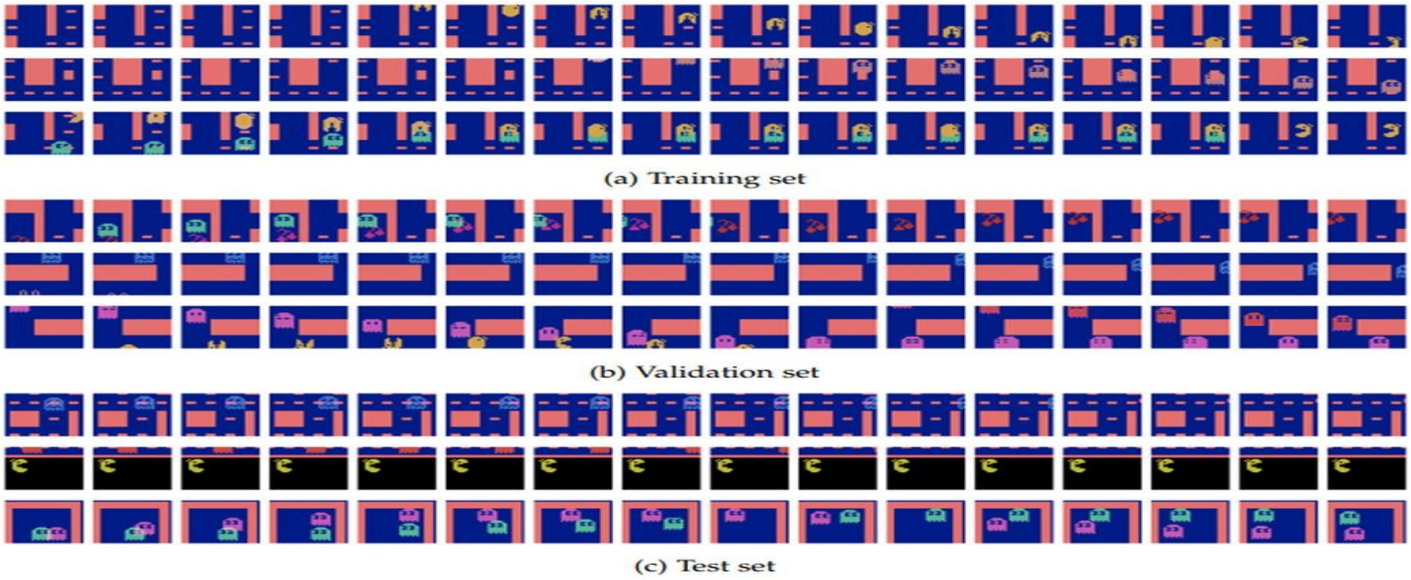

Fig. 6. Example sequences from different splits of the used MsPacman dataset.

These randomly selected frames have been cropped to 32 × 32 and filtered to ensure enough movement while training the model

2.4 Data preprocessing

The images of every sequence are rescaled to be in the range of [–1, 1]. To ensure that the predicted outputs exhibit the same value scale, a "tanh" activation function is used after the last transposed convolutional layer. Since every single image is quite big, the advantage of the fully-convolutional approach is taken; hence the network is only trained on random crops at a size of 32 × 32. Some random samples of these cropped images are shown in Figure 6. Unfortunately, the usage of small image crops follows that most of the used training sequences show no motion at all. After experiencing a strong preference for predicting the background of the world over the small moving objects only, a technique similar to [13] is used to ensure that each temporal sequence shows enough movement. Therefore, each time when a random part of the sequence is selected from the training set, the 2 difference between each consecutive frame pair is calculated. Afterward, this randomly chosen cropped sequence is rejected until the overall movement of the input sequence is higher than a given threshold of $25 \cdot n_{frames}$, or a specified repetition limit is reached to prevent an endless loop. Additionally, it is checked that there is also movement at the end of the input sequence to ensure that the whole movement is not just taking place at the beginning of the whole sequence only. Further, it reduces the chance that the network has to guess the movement of objects that enter the patch after the prediction-decoder has taken over, which it obviously cannot predict. Last but not least, the input and output sequence length has been slightly shortened compared to the other two datasets to 8 frames each. The reason for this is that this performed motion detection is almost inefficient when being applied on too long sequences, because of the high chance that all these fast-moving objects within the selected crop have already left the scene too quickly. However, even though this filtering of the final training examples sounds quite radical, it is to highlight that a high fraction of the input space is still static content. This can be attributed to the use of a convolutional layer with small window sizes that slide over the input space.

2.5 Data augmentation

In terms of data augmentation, the brightness or contrast of the image examples is not randomly modified, because it does not make sense in the context of this video game which uses only a fixed set of colors. But since the game world is mirrored horizontally, random horizontal flipping is performed in case the chosen

crop is not showing any parts of the status display at the bottom. Furthermore, it iterates over all sequences 256 times per epoch, reasoned by the fact that the dataset consists of only a few but very long sequences. A second reason is that a very short clip and a small random crop from these frame sequences are used only.

2.6 Characteristics and data generation for UCF-101 dataset

Finally, a third dataset is used to examine if the model can also deal with complex, natural videos. Therefore, the UCF-101 dataset is used. With its 13,320 clips and about 27 hours of video data, it belongs to the largest labeled dataset for human action recognition. The dataset is made of user-uploaded videos that contain both overloaded background and camera movement. All 101 categories can be divided into 25 main groups, where each video from the same group shares similar features, such as a roughly equal viewpoint. The action categories can also be separated into five types, namely human-object interaction, body-motion only, human-human interaction, playing musical instruments, and sports. Even though this dataset originated for human action recognition, it can be used in the context of frame prediction as well by simply using the raw video data only.

Before taking a deeper look into the characteristics and applied to preprocess steps, it should also be briefly mentioned that there exists an even larger video dataset for representation learning. This huge dataset is called Sports-1M and contains over 1.1 million YouTube video links of 478 classes that have been annotated in an automated process [6]. But due to infrastructural issues with such a huge dataset, as well as a tremendous time exposure regarding data preprocessing, it is not used in this paper.

2.7 Characteristics

The average video length of the whole dataset is about 6.2 seconds, while each single can range from about 1 second to a maximum of 71 seconds. Even though the initial paper states that each video has a fixed resolution and frame rate of 320 × 240 and 25 FPS respectively, a handful of videos exhibit a slightly different resolution nevertheless. Consequently, these frames are padded with zeros or cropped in the center to end up in an equal size for all videos.

The dataset provides three standard train/test splits intended to be used for either action recognition or action detection. The third standard split for action recognition is used in this work because it consists of the most videos for training, and allows the simplest divisibility of the test split. Since a huge training set can be expected to be fundamental for the network in order deeply explore the inner dynamics, the validation data is taken from the test split instead of the training data as otherwise customary. For the avoidance of doubt, other previous works using UCF-101 for frame prediction either have only used 10% of the test set for actual testing [13, p. 12] or did not make any comment about which data partitions they have chosen for validation and testing. Finally, it, therefore, ends up with 9624, 1232, and 2464 videos for the training, validation, and test set, respectively.

2.8 Data preprocessing

First, the width and height of all images are cut in half, resulting in downscaled videos at 160 × 120 using linear interpolation. This is done to compensate for the noisy, pixelated artifacts in the videos. Also, it increases the chance of finding a random crop at a size of 32 × 32 that has some true motion in it, instead of just flickering caused by noise. Second, the constraint regarding motion filtering when selecting the crop region of the randomly selected clip is slightly weakened. While a sequence that has a very low motion in the input frames is still rejected, it does not dismiss a sequence that has no motion at the end.

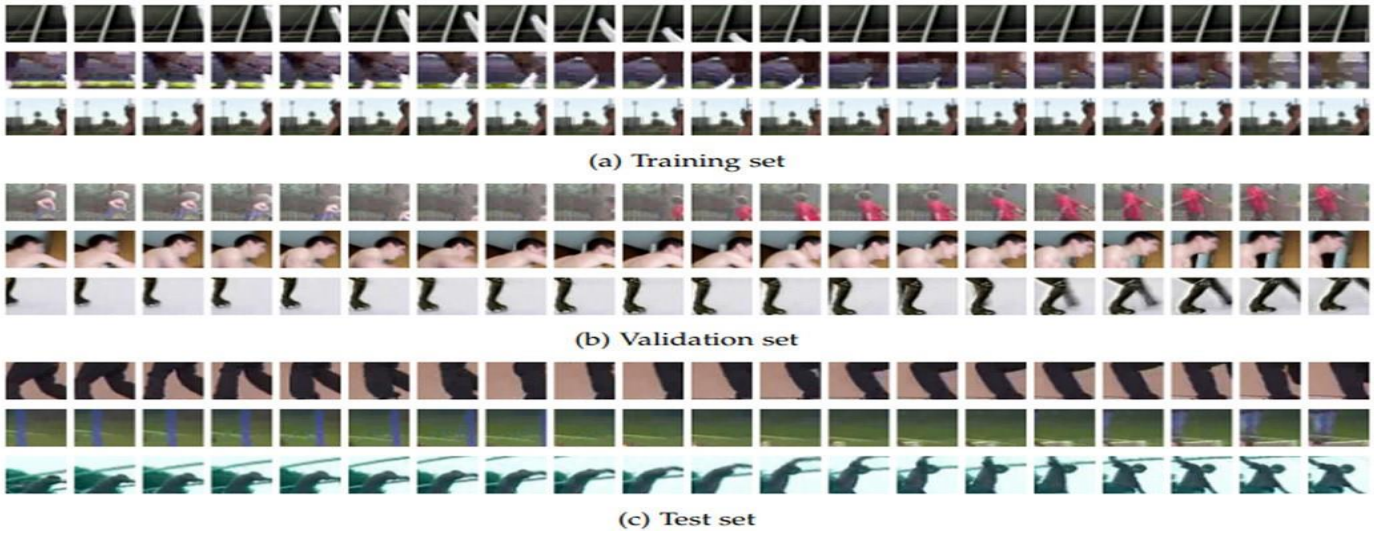

Fig. 7. Sequence examples from UCF-101. These frames have been randomly selected from the different splits, cropped to 32 × 32, and filtered to ensure they contain at least a small proportion of motion

Furthermore, because it is wasteful to load the whole video file into memory, especially in regard that only a very small portion of the data is used to generate a single example for the next batch, an additional offline preprocessing of the data is performed before the actual training process starts. For that reason, it iterates over all raw video files and generates non-overlapping binary video sequences with a length of 30 frames each. Fortunately, the generated files can be reused after doing this process once. Finally, it ends up with 55,150 clips for the training set, 7183 clips for the validation set, and 14,451 clips for the test set.

Data augmentation

Regarding data augmentation, the contrast and brightness of the overall image sequence are randomly modified by a delta of ± 20%. Random horizontal flipping is also performed for the training data. While the contrast or brightness in the validation or test data is not randomly changed, their size is doubled by using both the normal and the flipped instances. Four crops from every video are used in each evaluation iteration to have a balance between more consistent evaluations and still an acceptable processing time. These augmentation steps are performed on-the-fly. To facilitate this, an advanced double-buffered input queue is used. The first filename queue is randomly filled with references to the binary sequence files generated before. Afterward, 16 CPU threads dequeue a reference from this queue, load the sequence into memory and then perform all preprocessing steps in parallel. This is to generate a single training example. Finally, this example is then pushed to the shuffled batch queue, from which the model loads its batches in every iteration. Consequently, there is no waiting time between each training step.

3 Methodology and experiments

3.1 Data augmentation

The synthetic Moving MNIST dataset is used as a starting point for the evaluation of our model. Given an input sequence of ten frames, the network has to understand and encode the motion of this input to predict the next ten frames of the future. In this section, the results are qualitatively and quantitatively compared with other network models, because this dataset was used in several previous works as well. The loss layer used in this section sets $\lambda_{ssim} = 0$ by default. The consequences of the used scheduled sampling technique are examined first. Figure 8 visualizes the training and validation binary cross-entropy using our standard model with either scheduled sampling (SS) or always sampling (AS). The latter method means the approach to always sample from the previously generated frame, which is comparable to SS with a constant sampling probability of p =

0. It can be seen that there is a huge discrepancy between the training and validation error in the starting phase, where the SS approach is mainly trained on input samples taken from the ground truth. This can be explained that sampling on the ground truth in every time step of the recurrent network tremendously speeds up the convergence of the training loss because each cell does not have to correct errors of the previous cells. On contrary, the validation loss is very bad in this phase, since the results represent the average out of ten predicted frames and the Spatio-temporal decoder suddenly receives its feature space predictions as input to predict the next frame, which is in contrast to the procedure while training at that point. But the interesting insight here is that as soon as the scheduled sampling component completely changed the input behavior to inference mode, the achieved prediction performance is continuously better compared to the approach of always sampling from previously generated frames directly at the very beginning.

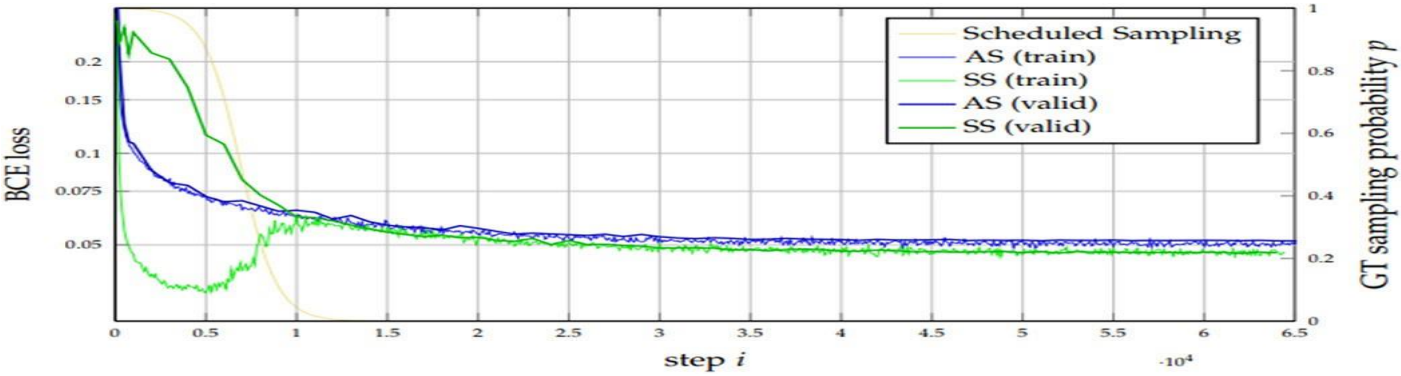

Fig. 8. Influences of scheduled sampling regarding the training and validation error in the context of recurrent networks and future frames prediction on moving MNIST

This behavior could be explained that the SS approach introducing a form of pre-training phase, where the network can learn to predict the next frame when a perfect current frame is given. Hence, it has not dealt with errors made in the previous time step. By slowly changing this behavior to the mode as it is used during inference, it then also starts to learn robustness against imperfect input frames. Similar behavior can be seen when comparing the results of the PSNR metric between AA and SS in Figure 8b. However, when we look at the sharpness difference metric in Figure 8a, it can be seen that the sharpness of the predictions is continuously better when scheduled sampling is used.

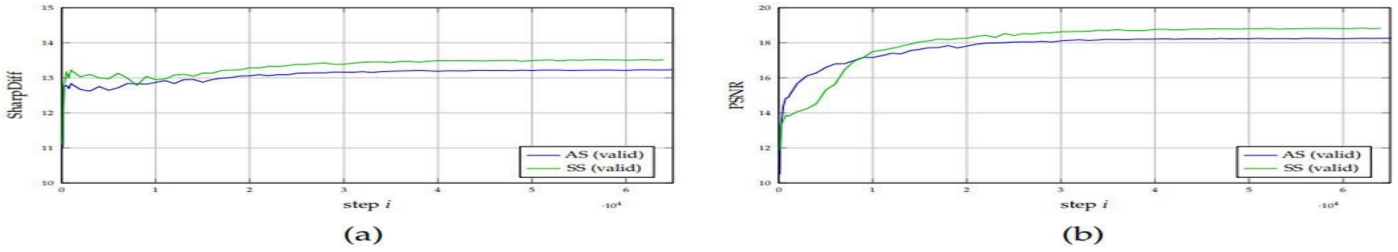

Fig. 9. Comparison of validation results based on a model either using the scheduled sampling or the always sampling training technique

3.2 Batch normalization

The use of batch normalization in the spatial encoder and decoder components is investigated next. Figure 10 shows a clear advantage of using BN layers. It can be argued that our model can profit strongly from batch normalization because it does not only support the convolutional encoder and decoder to learn faster by compensating for the internal covariate shift, but also both recurrent networks in between benefit from it. This can be reasoned by the fact that frame representations in feature space, which are produced by the spatial encoders, tend to have a more stable distribution. Consequently, the Spatio-temporal encoder has fewer

problems extracting useful patterns from these representations while consuming them in the course of the training process.

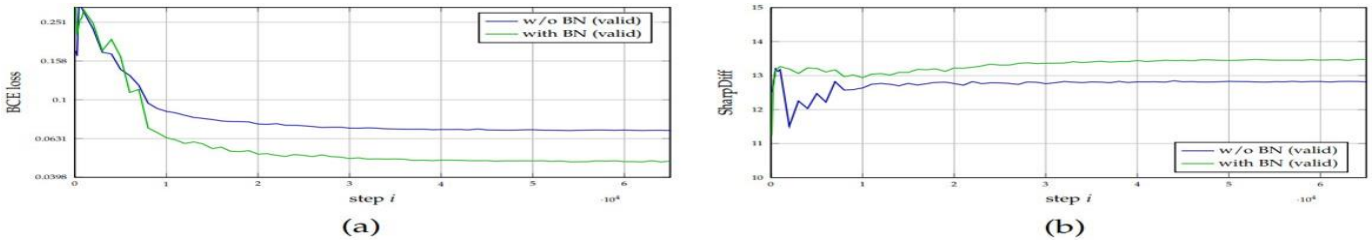

Fig. 10. Comparison of the validation results from two model instances where one uses batch normalization layers in the spatial encoder and decoder components

3.3 Learning rate decay

The network's learning behavior using two different exponential decay rates is illustrated in Figure 11. In this example, the learning rate is slightly decayed after each epoch by a specified factor. Because the Moving MNIST dataset is generated on-the-fly, its total size is specified to be virtually 32,000. In this way, the learning rate is effectively decayed after each validation step, since a batch size of 32 is used. The network denoted with blue has a clear disadvantage because its learning rate at the end becomes so small that it almost stops learning. However, throughout the experiments, it is determined that a very low learning rate at the end of the training is beneficial regarding the image quality and finer details of the generated images.

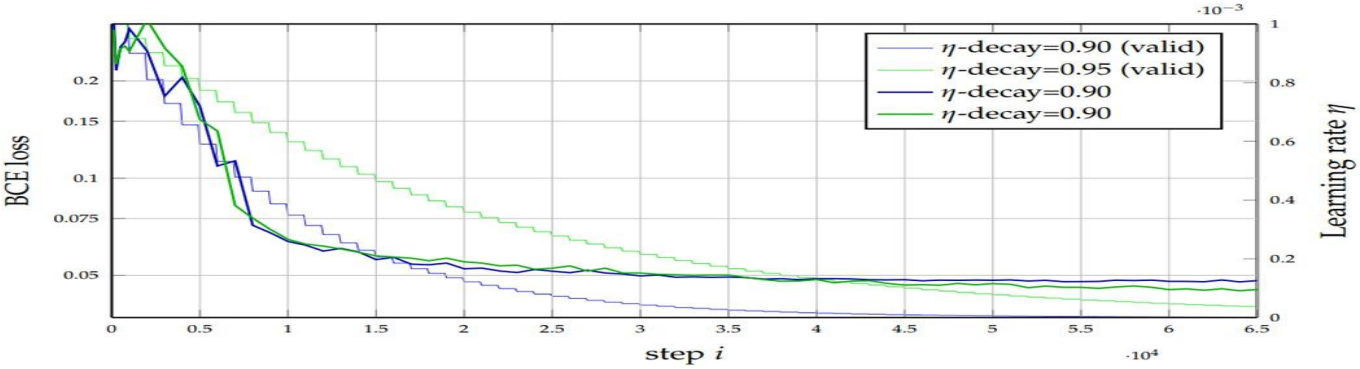

Fig. 11. Validation losses of two identical network models using different learning rate decay rates in combination with adam optimizer

3.4 Loss layer

As a last experiment of the test series, the impact of different loss terms is examined. Three networks with identical standard configurations, but different objective functions have therefore been trained. Starting from a network using binary cross-entropy only, perceptual-motivated loss terms like GDL and SSIM are added one after another until it ends up in the triplet loss. The validation results are depicted in Figure 12.

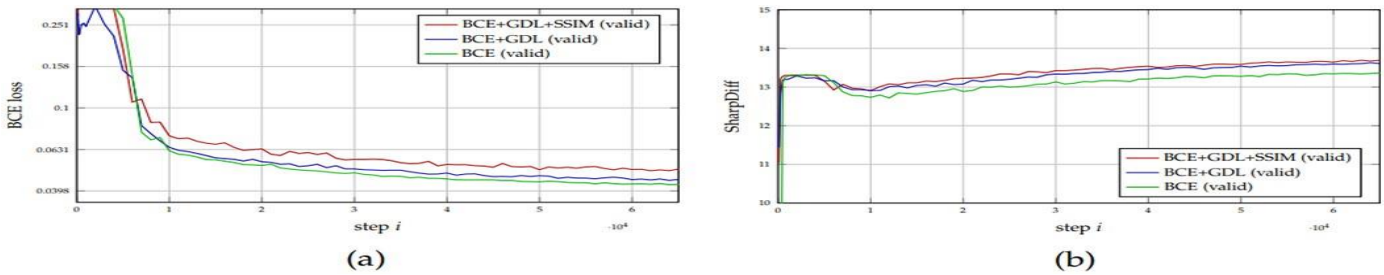

Fig. 12. Comparison of the validation loss using different objective functions

While the network purely trained on the BCE loss function performs best regarding the binary cross-entropy on the validation set, it nevertheless performs worst according to the sharpness difference metric shown in

Figure 12b. These results are not surprising, because the network using the triplet loss function for example is putting more emphasis on other properties, in contrast to specifically optimizing it regarding the binary cross-entropy. However, after the qualitative analysis of many generated prediction samples in the course of the experiments, it seemed that networks using the simple BCE loss produced the best results. Especially in samples where one handwritten number crosses the other number, predictions of the farther future kept slightly more stable. All in all, perceptual-motivated loss terms seem to be useless or even marginally counter-productive when applied on binary Moving MNIST image frames.

4 Result and discussion

4.1 Test results

For the final evaluation of our model with Moving MNIST, the qualitative and quantitative performance of the best configuration is assessed on the test set. Inspired by the findings of the model exploration, as well as a rudimentary grid search, the model for this final test ends up using the default settings as described earlier, but with an exponential learning rate decay of $\alpha = 0.95$ and a loss layer that utilizes no perceptual motivated loss term. Three models of this network with different numbers of recurrent layers are trained for 100,000 steps.

4.2 Quantitative results

A comparison of the best-performing model with experimental results of other works is given in Table 1. It can be seen that the proposed model exhibits way fewer model parameters, especially compared to FC-LSTM approaches. Nevertheless, our ConvLSTM model can cut the average pixel-wise test error of the best-performing competitive model in halves. The listed results are taken from several published papers. However, the results of FC-LSTM-Combo published in [9] could not be reproduced using their open-source code. After training the model for about 1,000,000 training steps, which takes almost one week, the results are not nearly as good as in the paper. Even though about 20,000 training iterations are already sufficient for the proposed model in this paper to generate even better results.

Table 1. Comparison with other networks on moving MNIST

Model	**Trainable Parameters**	**BCE Test Error**
FC-LSTM-Combo(2048–2048) [9]	214,001,664	0.0855
FC-LSTM(2048–2048) [10]	142,667,776	0.1180
ConvLSTM(5/128-64-64) [10]	7,585,296	0.0896
2enc-ConvLSTM(7/45-45)^3 [11]	6,132,203	0.0835
3enc-ConvLSTM-SS-3dec(5/64)	1,841,793	0.0524
3enc-ConvLSTM-SS-3dec(5/64-64)	3,570,817	0.0414
3enc-ConvLSTM-SS-3dec(5/64-64-64)	5,299,841	**0.0407**

Notes: The numbers in brackets identify the number of hidden units per layer in the case of FC-LSTMs, and for ConvLSTMs the hidden-to-hidden kernel size followed by the feature maps per layer are listed. Further, 3enc denotes tree convolutional layers in the spatial encoder. The model of this paper is called 3enc-ConvLSTM-SS-3dec.

5 Conclusion and future work

We presented in this paper, existing unsupervised deep learning approaches for future frame prediction in videos and incorporated their mentioned insights to build a neural network model that combines its strengths. Our finally proposed network architecture utilizes the recurrent decoder-encoder framework using

ConvLSTM cells that can preserve the Spatio-temporal correlations of the data. Further, the extensive use of batch normalization and the scheduled sampling training strategy enables our model to outperform many existing approaches in at least synthetic or simple videos, even though our network contains a lower model complexity and is trained for fewer iterations. An identified key driver to obtaining accurate prediction results has been the choice of an appropriate loss function that considers human perception. However, the best composition of different objective functions strongly depends on the underlying data. The network was further evaluated quantitatively and qualitatively on three datasets of different complexity in several scenarios and investigated the model's behavior regarding hyperparameter changes. The best performing model in each case was then compared to results from related works. Additionally, a high-level framework for TensorFlow projects was presented that enables to use of advanced features out-of-the-box and radically reduces boilerplate code. In the future, the proposed network model should be examined and fine-tuned in more detail. We strongly believe that this architecture can obtain even better results after performing a more extensive hyperparameter search, training it for many more iterations, or when using a larger dataset like Sports-1M. Unfortunately, this is beyond the timeframe of this paper, which is why some evaluations were performed on networks that still had further potential in case of more training iterations.

6 References

I. Goodfellow, Y. Bengio, and A. Courville. "Deep Learning." Book in preparation for MIT Press. 2016.

A. Kar. "Future Image Prediction using Artificial Neural Networks." In: 2012.

S. Ji, W. Xu, M. Yang, and K. Yu. "3D Convolutional Neural Networks for Human Action Recognition." In: IEEE Trans. PAMI. 2013, pp. 221–231. https://doi.org/10.1109/ TPAMI.2012.59

K. Simonyan and A. Zisserman. "Two-Stream Convolutional Networks for Action Recognition in Videos." In: Proc. NIPS. 2014, pp. 568–576.

J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. "Long-term Recurrent Convolutional Networks for Visual Recognition and Description." In: 2014. https://doi.org/10.21236/ADA623249

A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. "Large- scale Video Classification with Convolutional Neural Networks." In: CVPR. 2014, pp. 1725–1732. https://doi.org/10.1109/CVPR.2014.223

P. Fischer, A. Dosovitskiy, P. Häusser, C. Hazırbas, and V. Golkov. "FlowNet: Learning

Optical Flow with Convolutional Networks." In: 2015. https://doi.org/10.1109/ ICCV.2015.316

N. K. Verma. "Future Image Frame Generation using Artificial Neural Network with Selected Features." In: AIPR. Oct. 2012. https://doi.org/10.1109/AIPR.2012.6528189

N. Srivastava, E. Mansimov, and R. Salakhutdinov. "Unsupervised Learning of Video Representations using LSTMs." In: ICML. 2015.

X. Shi, Z. Chen, H. Wang, and D.-Y. Yeung. “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting.” In: 2015.

V. Pătrăucean, A. Handa, and R. Cipolla. “Spatio-Temporal Video Auto Encoder with Differentiable Memory.” In: ICLR. 2016.

A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, and C. Suen. Standford: Unsupervised Feature Learning and Deep Learning. 2013. http://ufldl.stanford.edu/wiki/index.php/Data_Preprocessing (visited on 10/03/2016).

M. Mathieu, C. Couprie, and Y. LeCun. “Deep Multi-Scale Video Prediction Beyond Mean Square Error.” In: ICLR. Feb. 2016.