SPATIO-TEMPORAL VIDEO AUTOENCODER WITH DIFFERENTIABLE MEMORY

Viorica Pătrăucean, Ankur Handa & Roberto Cipolla
Department of Engineering
University of Cambridge, UK
{vp344,ah781,rc10001}@cam.ac.uk

ABSTRACT

We describe a new spatio-temporal video autoencoder, based on a classic spatial image autoencoder and a novel nested temporal autoencoder. The temporal encoder is represented by a differentiable visual memory composed of convolutional long short-term memory (LSTM) cells that integrate changes over time. Here we target motion changes and use as temporal decoder a robust optical flow prediction module together with an image sampler serving as built-in feedback loop. The architecture is end-to-end differentiable. At each time step, the system receives as input a video frame, predicts the optical flow based on the current observation and the LSTM memory state as a dense transformation map, and applies it to the current frame to generate the next frame. By minimising the reconstruction error between the predicted next frame and the corresponding ground truth next frame, we train the whole system to extract features useful for motion estimation without any supervision effort. We believe these features can in turn facilitate learning high-level tasks such as path planning, semantic segmentation, or action recognition, reducing the overall supervision effort.

1 INTRODUCTION

High-level understanding of video sequences is crucial for any autonomous intelligent agent, *e.g.* semantic segmentation is indispensable for navigation, or object detection skills condition any form of interaction with objects in a scene. The recent success of convolutional neural networks in tackling these high-level tasks for static images opens up the path for numerous applications. However, transferring the capabilities of these systems to tasks involving video sequences is not trivial, on the one hand due to the lack of video labelled data, and on the other hand due to convnets' inability of exploiting temporal redundancy present in videos. Motivated by these two shortcomings, we focus on reducing the supervision effort required to train recurrent neural networks, which are known for their ability to handle sequential input data (Williams & Zipser, 1995; Hochreiter et al., 2000).

In particular, we describe a spatio-temporal video autoencoder integrating a differentiable short-term memory module whose (unsupervised) training is geared towards motion estimation and prediction (Horn & Schunck, 1994). This choice has biological inspiration. The human brain has a complex system of visual memory modules, including visual short-term memory (VSTM), iconic memory, and long-term memory (Hollingworth, 2004). Among them, VSTM is responsible mainly for understanding visual changes (movement, light changes) in dynamic environments, by integrating visual stimuli over periods of time (Phillips, 1974; Magnussen, 2000). Moreover, the fact that infants are able to handle occlusion, containment, and covering events by the age of 2.5 months (Baillargeon, 2004) could suggest that the primary skills acquired by VSTM are related to extracting features useful for motion understanding. These features in turn could generate objectness awareness based on the simple principle that points moving together belong to the same object. Understanding objectness is crucial for high-level tasks such as semantic segmentation or action recognition (Alexe et al., 2012). In this spirit, our approach is similar to the recent work of Agrawal et al. (2015), who show that the features learnt by exploiting (freely-available) ego-motion information as supervision data are as good as features extracted with human-labelled supervision data. We believe that our work is complementary to their approach and integrating them could lead to an artificial agent with enhanced vision capabilities.

Our implementation draws inspiration from standard video encoders and compression schemes, and suggests that deep video autoencoders should differ conceptually from spatial image autoencoders. A video autoencoder need not reproduce *by heart* an entire video sequence. Instead, it should be able to encode the significant differences that would allow it to reconstruct a frame given a previous frame. To this end, we use a classic convolutional image encoder – decoder with a nested memory module composed of convolutional LSTM cells, acting as temporal encoder. Since we focus on learning features for motion prediction, we use as temporal decoder a robust optical flow prediction module together with an image sampler, which provides immediate feedback on the predicted flow map. At each time step, the system receives as input a video frame, predicts the optical flow based on the current frame and the LSTM content as a dense transformation map, and applies it to the current frame to predict the next frame. By minimising the reconstruction error between the predicted next frame and the ground truth next frame, we are able to train the whole system for motion prediction with no supervision effort. Other modules handling other types of variations like light changes could be added in parallel, inspired by neuroscience findings which suggest that VSTM is composed of a series of different modules specialised in handling the low-level processes triggered by visual changes, all of them connected to a shared memory module (Magnussen, 2000). Note that at the hardware level, this variations-centred reasoning is similar to event-based cameras (Boahen, 2005), which have started to make an impact in robotic applications (Kim et al., 2014).

Our contribution resides at the experimental level. We propose a spatio-temporal version of LSTM cells to serve as a basic form of visual short-term memory (temporal encoder), and make available an end-to-end differentiable architecture with built-in feedback loop, that allows effortless training and experimentation with the goal of understanding the role of such an artificial visual short-term memory in low-level visual processes and its implications in high-level visual tasks. All our code (Torch implementation) is available online (Con).

2 Related Work

Architectures based on LSTM cells (Hochreiter & Schmidhuber, 1997) have been very successful in various tasks involving one-dimensional temporal sequences: speech recognition (Sak et al., 2014), machine translation (Sutskever et al., 2014), music composition (Eck & Schmidhuber), due to their ability to preserve information over long periods of time. Multi-dimensional LSTM networks have been proposed to deal with (2D) images (Graves et al., 2007) or (3D) volumetric data (Stollenga et al., 2015), treating the data as spatial sequences. Since in our work we aim at building a visual short-term memory, customised LSTM cells that deal with temporal sequences of spatial data represent a natural choice.

Recently, Srivastava et al. (2015) proposed an LSTM-based video autoencoder, which aims at generating past and future frames in a sequence, in an unsupervised manner. However, their LSTM implementation treats the input as sequences of vectors by flattening the frames or by using 1D frame representations produced after the last fully-connected layer of a convolutional neural network. This results in a large number of parameters, since the activations of the LSTM cells use fully-connected linear operations, which are not necessarily useful for 2D images, since natural image statistics indicate only local correlations. Another difference is in the architecture setup. We are not interested in training a black-box to produce past and future frames in the sequence. Instead we aim at a transparent setup, and train a more generic memory module together with specialised modules able to decode the memory content and on which appropriate constraints (sparsity, smoothness) can be imposed.

Our approach is partially related to optical flow estimation works like DeepFlow (Weinzaepfel et al., 2013) and FlowNet (Fischer et al., 2015). However, these methods use purely supervised training to establish matches between consecutive pairs of frames in a sequence, and then apply a variational smoothing. Our architecture is end-to-end differentiable and integrates a smoothness penalty to ensure that nearby pixels follow a locally smooth motion, and requires no labelled training data.

As mentioned in the previous section, our work is similar in spirit to (Agrawal et al., 2015), by establishing a direct link between vision and motion, in an attempt to reduce supervision effort for high-level scene understanding tasks. However, instead of relying on data provided by an inertial measurement unit from which we can estimate only a global transformation representing the ego-motion, we predict a dense flow map by integrating visual stimuli over time using a memory module,

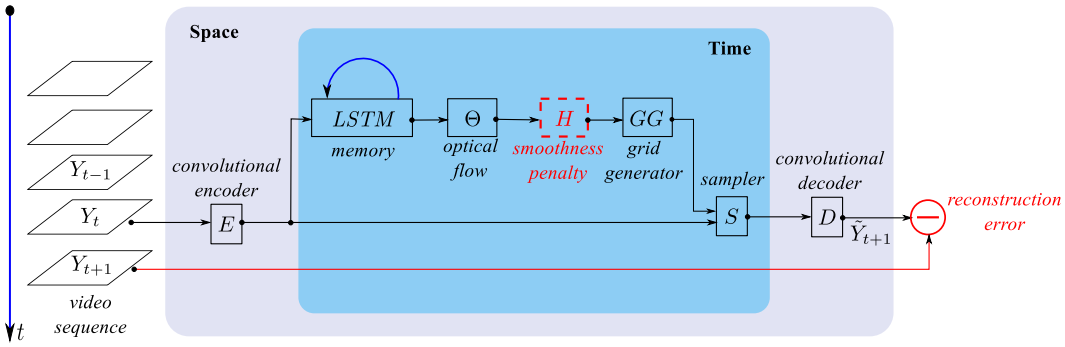

Figure 1: Spatio-temporal video autoencoder.

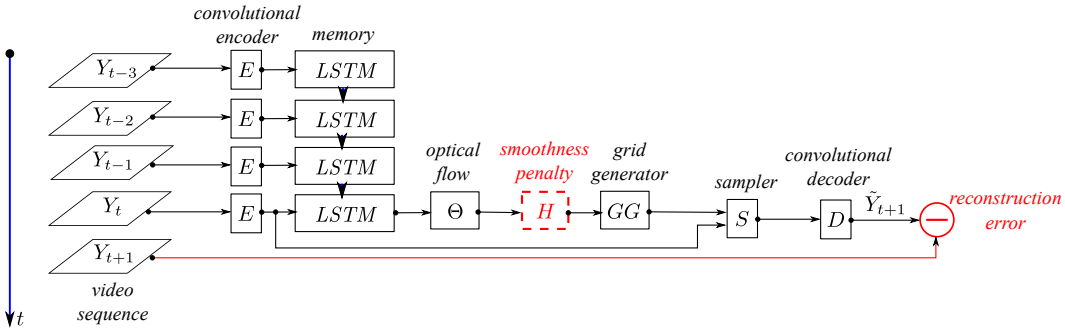

Figure 2: Unfolding of the architecture for $\tau = 4$ time steps. Only the output from the last time step is used in optimisation, *i.e.* we make sure that at the prediction moment, the memory module has a context of at least τ frames. Note that the encoder module preceding the memory module needs to be unfolded as well, even if it is not recurrent, to ensure that the gradients are backpropagated correctly.

and then we use a built-in feedback loop to assess the prediction. The dense flow map is useful to explain dynamic environments, where different objects can follow different motion models, hence a global transformation is not sufficient (Menze & Geiger, 2015).

3 Architecture

Our architecture consists of a temporal autoencoder nested into a spatial autoencoder (see Figure 1). At each time step, the network takes as input a video frame Y_t of size $H \times W$, and generates an output of the same size, representing the predicted next frame, $\tilde{Y}_{t+1}$. In the following, we describe each of the modules in detail.

3.1 Spatial autoencoder E and D

The spatial autoencoder is a classic convolutional encoder – decoder architecture.

In the vanilla version of our architecture, the encoder E contains one convolutional layer, followed by $\tanh$ non-linearity and a spatial max-pooling with subsampling layer. The decoder D mirrors the encoder, except for the non-linearity layer, and uses nearest-neighbour spatial upsampling to bring the output back to the size of the original input. After the forward pass through the spatial encoder $Y_t \xrightarrow{E} x_t$, the size of the feature maps x_t is $d \times h \times w$, d being the number of features, and h and w the height and width after downsampling, respectively. In the experiments section, we show results obtained with deeper versions of the architecture as well.

3.2 TEMPORAL AUTOENCODER

The goal of the temporal autoencoder is to capture significant changes due to motion (ego-motion or movement of the objects in the scene), that would allow it to predict the visual future, knowing the past and the present. In a classic spatial autoencoder (Masci et al., 2011), the encoder and decoder learn proprietary feature spaces that allow an optimal decomposition of the input using some form of regularisation to prevent learning a trivial mapping. The encoder decides freely upon a decomposition based on its current feature space, and the decoder constrains the learning of its own feature space to satisfy this decomposition and to reconstruct the input, using usually operations very similar to the encoder, and having the same number of degrees of freedom. Differently from this, the proposed temporal autoencoder has a decoder with a small number of trainable parameters, whose role is mainly to provide immediate feedback to the encoder, but without the capacity of amending encoder's mistakes like in the spatial case. In optimisation terms, the error during learning is attributed mainly to the encoder, which is now more constrained to produce sensible feature maps.

3.2.1 MEMORY MODULE $LSTM$

The core of the proposed architecture is the memory module playing the role of a temporal encoder. We aim at building a basic visual short-term memory, which preserves locality and layout ensuring a fast access and bypassing more complicated addressing mechanisms like those used by neural Turing machines (Graves et al., 2014). To this end, we use customised spatio-temporal LSTM cells with the same layout as the input. At each time step t, the $LSTM$ module receives as input a new video frame after projection in the spatial feature space. This is used together with the memory content and output of the previous step $t-1$ to compute the new memory activations. This operation can be easily understood by unfolding the recurrent module over a number of time steps (see Figure 2).

Classic LSTM cells operate over sequences of (one-dimensional) vectors and perform biased linear (fully-connected) transformations, followed by non-linearities to compute gate and cell activations. In our case, to deal with the spatial and local nature of the video frames, we replace the fully-connected transformations with spatial local convolutions.

All-in-all, the activations of a spatio-temporal convolutional LSTM cell at time t are given by:

$$\left\{\begin{array}{lcl} i_t & = & \sigma(x_t * w_{xi} + h_{t-1} * w_{hi} + w_{i\text{bias}}) \\ f_t & = & \sigma(x_t * w_{xf} + h_{t-1} * w_{hf} + w_{f\text{bias}}) \\ \tilde{c}_t & = & \tanh(x_t * w_{x\tilde{c}} + h_{t-1} * w_{h\tilde{c}} + w_{\tilde{c}\text{bias}}) \\ c_t & = & \tilde{c}_t \odot i_t + c_{t-1} \odot f_t \\ o_t & = & \sigma(x_t * w_{xo} + h_{t-1} * w_{ho} + w_{o\text{bias}}) \\ h_t & = & o_t \odot \tanh(c_t) \end{array}\right. \tag{1}$$

where x_t represents the input at time t, *i.e.* the feature maps of the frame t; i_t, f_t, $\tilde{c}_t$, and o_t represent the input, forget, cell, and output gates, respectively; c_t, c_{t-1}, h_t, and h_{t-1} are the memory and output activations at time t and $t-1$, respectively; σ and $\tanh$ are the sigmoid and hyperbolic tangent non-linearities; $*$ represents the convolution operation, and $\odot$ the Hadamard product. For input feature maps of size $d \times h \times w$, the $LSTM$ module outputs a memory map of size $d_m \times h \times w$, where d_m is the number of temporal features learnt by the memory. Note that the recurrent connections operate only over the temporal dimension, and use local convolutions to capture spatial context, unlike multi-dimensional LSTM versions that use spatial recurrent connections (Graves et al., 2007; Stollenga et al., 2015). We implemented and experimented with spatial recurrent 2D LSTM cells as well, to gain more flexibility in handling spatial context. However, since the spatial recurrent connections require sequential processing, the execution time becomes prohibitive especially during training; therefore we decided to sacrifice the spatial recurrent connections. Note that a similar convolutional LSTM implementation was recently used by Shi et al. (2015) for precipitation nowcasting.

3.2.2 OPTICAL FLOW PREDICTION Θ WITH HUBER PENALTY H

The optical flow prediction module generates a dense transformation map $\mathcal{T}$, having the same height and width as the memory output, with one 2D flow vector per pixel, representing the displacement in x and y directions due to motion between consecutive frames. $\mathcal{T}$ allows predicting the next

frame by warping the current frame. We use two convolutional layers with relatively large kernels (15×15) to regress from the memory feature space to the space of flow vectors. Large kernels are needed since the magnitude of the predicted optical flow is limited by the size of the filters. To ensure local smoothness, we need to penalise the local gradient of the flow map $\nabla\mathcal{T}$. We add a penalty module H whose role is to forward its input unchanged during the forward pass, and to inject non-smoothness error gradient during the backward pass, towards the modules that precede it in the architecture and that might have contributed to the error. We use Huber loss as penalty (2) with its corresponding derivative (3), due to its edge-preserving capability (Werlberger et al., 2009). The gradient of the flow map is obtained by convolving the map with a fixed (non-trainable) $2\times3\times3$ filter, corresponding to a 5-point stencil, and 0 bias.

$$\mathcal{H}_\delta(a_{ij}) = \begin{cases} \frac{1}{2}a_{ij}^2, & \text{for } |a_{ij}| \leq \delta, \\ \delta(|a_{ij}| - \frac{1}{2}\delta), & \text{otherwise,} \end{cases} \tag{2}$$

$$\nabla\mathcal{H}_\delta(a_{ij}) = \begin{cases} a_{ij}, & \text{for} |a_{ij}| \leq \delta, \\ \delta\text{sign}(a_{ij}), & \text{otherwise.} \end{cases} \tag{3}$$

Here, a_{ij} represent the elements of $\nabla\mathcal{T}$. In our experiments, we used $\delta = 10^{-3}$.

3.2.3 GRID GENERATOR GG AND IMAGE SAMPLER S

The grid generator GG and the image sampler S output the predicted feature maps $\tilde{x}_{t+1}$ of the next frame after warping the current feature maps x_t with the flow map produced by the Θ module. We use similar differentiable grid generator and image sampler as Spatial Transformer Network (STN) (Jaderberg et al., 2015). The output of S, of size $d \times h \times w$, is considered as a fixed $h \times w$ grid, holding at each (x_o, y_o) position a feature map entry of size $1\times1\times d$. We modified the grid generator to accept one transformation per pixel, instead of a single transformation for the entire image as in STN. Given the flow map $\mathcal{T}$, GG computes for each element in the grid the source position (x_s, y_s) in the input feature map from where S needs to sample to fill in the position (x_o, y_o):

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \mathcal{T}(x_o, y_o) \begin{pmatrix} x_o \\ y_o \\ 1 \end{pmatrix}, \mathcal{T}(\cdot,\cdot) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{pmatrix}. \tag{4}$$

The Θ module outputs two parameters (t_x, t_y) for each pixel. The forward pass of GG is given by equation (4). In the backward pass, GG simply backpropagates the derivative of equation (4) w.r.t. the input parameters (t_x, t_y).

3.2.4 LOSS FUNCTION

Training the network comes down to minimising the reconstruction error between the predicted next frame and the ground truth next frame, under specific constraints. To enforce edge preservation, we use a loss function similar to (Brox et al., 2004; Eigen & Fergus, 2015). Namely, we consider the error between the intensities and gradients respectively of the predicted next frame and of the ground truth next frame. Additionally, as mentioned before, we inject Huber penalty gradient during back-propagation on the optical flow map. We do not apply any preprocessing step to the input images (subtracting mean, or local contrast normalisation), since this would discard the low frequencies, which we believe are important for motion estimation. Hence, the overall objective to minimise is given by:

$$\mathcal{L}_t = \|\tilde{Y}_{t+1} - Y_{t+1}\|_2^2 + \|\nabla\tilde{Y}_{t+1} - \nabla Y_{t+1}\|_2^2 + w_H\mathcal{H}(\nabla\mathcal{T}), \tag{5}$$

where w_H is a hard-coded parameter used to weight the smoothness constraints w.r.t. the data term; in our experiments $w_H = 10^{-2}$.

3.3 NETWORK PARAMETERS

The vanilla version of the network has 703,651 trainable parameters distributed as described in Table 1. The spatial encoder and decoder have 32 filters each, size 7×7. The memory module ($LSTM$) has 45 filters, size 7×7, and the optical flow regressor Θ has 2 convolutional layers, each with 2 filters of size 15×15, and a 1×1 convolutional layer. The other modules: GG, H, and S have no trainable parameters.

Module	Number of trainable parameters
Spatial encoder (E)	1,600
Spatial decoder (D)	1,569
Memory ($LSTM$)	679,320
Optical flow regressor (Θ)	21,162
Total	**703,651**

Table 1: Network parameters per module.

4 TRAINING

To train the proposed architecture, we follow a setup similar to (Stollenga et al., 2015); we use *rmsprop*, with parameters $\epsilon = 10^{-5}$, $\rho = 0.9$, and a decaying learning rate following the rule $\eta = 10^{-4}\left(\sqrt[100]{\frac{1}{2}}\right)^{\text{epoch}}$. The training data is represented by binary or gray-scale video sequences. At each time step, we sample a subsequence of τ consecutive frames, and the goal is to reconstruct the next frame, $(\tau+1)^{\text{th}}$, in the sequence. Therefore, the network is unfolded over τ steps (as illustrated in Figure 2). After processing the subsequence, we update the parameters and we reset the memory. The initialisation of all the convolutional layers except those involved in the memory module was done using *xavier* method (Glorot & Bengio, 2010). The parameters of the memory module, except the biases, were initialised from a uniform distribution $\mathcal{U}(-0.08, 0.08)$. The biases of the forget gates are initialised to 1; for the other gates we set the biases to 0. During training, before each parameter update, we clip the gradients to lie in a predefined range, to avoid numerical issues (Graves, 2013).

Our implementation is based on Torch library (Collobert et al.) and extends the *rnn* package (Léonard et al., 2015). All our code (vanilla architecture) is available online (Con). The training was done on an 880M Nvidia GPU (8G of GPU memory).

5 EXPERIMENTS

As a sanity check to confirm the ability of the grid generator GG and image sampler S to generate the next frame given the current frame and the correct optical flow map, we ran simple warping tests using Sintel dataset (Butler et al., 2012), by isolating the two modules from the rest of the architecture. Figure 3 shows an example of warping. Note that since the flow displacement is significant in this dataset, the sampling result can contain artefacts, especially near boundaries, caused by occlusions. The average per-pixel error induced by the sampler on this dataset is of 0.004 (for pixel values in $[0, 1]$). However, this test was done by sampling directly from the input frames, no spatial encoder – decoder was used. In our architecture, these artefacts will be washed out to a certain extent by the convolutional decoder. The optical flow throughout this section is displayed using the colour code from (Baker et al., 2011), illustrated in Figure 3, top-right corner.

To assess qualitatively and quantitatively the behaviour of the proposed architecture and of its components, we ran unsupervised experiments on synthetic and real datasets[1].

[1] Since our system deals with optical flow prediction, one might think that existing datasets in the optical flow community could represent suitable training and/or testing data. However, these datasets contain flow maps for only pairs of images (Fischer et al., 2015), or for very short video sequences (2 or 8 frames in Middlebury dataset (Baker et al., 2011)); hence not suitable for our experiments, where at least 10 or 12 frames are used to make flow predictions. Sintel dataset (Butler et al., 2012) contains indeed ground truth optical flow for longer video sequences. However, it exhibits a complex combination of camera self-motion and moving objects, with a relatively low frame rate, leading to very large (non-realistic) displacements between consecutive frames. Note also that the self-motion makes the entire scene to *move* in terms of optical flow, making the prediction very hard and, in the same time, not useful for the purposes of our work: if all the points in the scene move, it is difficult to delineate objects.

Figure 3: Warping performed by the GG and S modules, given a video frame and the ground truth optical flow map. First line: input frame and ground truth optical flow map displayed using the colour code shown in the top-right corner of the image; the colour encodes the direction of movement and the colour intensity reflects the magnitude; darker shades correspond to higher magnitudes (Baker et al., 2011). Second line: ground truth next frame and sampled next frame; note the artifacts close to the boundaries.

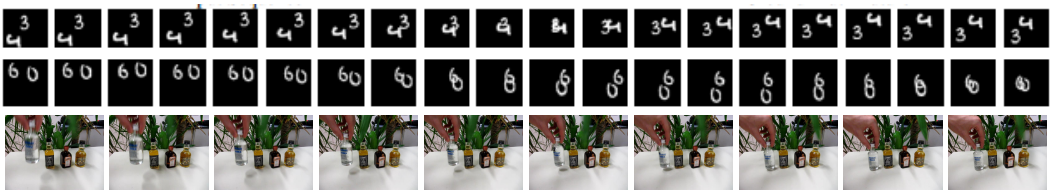

Figure 4: Samples of images from moving MNIST and PROST datasets.

SYNTHETIC DATASET

Moving MNIST dataset (Srivastava et al., 2015) consists of sequences of 20 frames each, obtained by moving (translating) MNIST digit images inside a square of size 64×64, using uniform random sampling to obtain direction and velocity; the sequences can contain several overlapping digits in one frame (see Figure 4, first two lines, for sample images). We generated 10k sequences for training and 3k sequences for validation. We ran experiments on the original dataset and on a modified version of it, in which we superimposed a pair of lines, one vertical and one horizontal, at random positions, to act as fixed reference in understanding the digits' movement.

Figure 5 shows representative qualitative results obtained on moving MNIST dataset and on the modified version. The slightly cleaner optical flow in the experiment with lines could suggest that it is important to have static image elements when trying to understand moving elements. These qualitative results show that the network captures the movement and generates accurate predictions for the next frame in the sequence, even when overlaps or bouncing against the limits of the squares occur. The lines in the modified moving MNIST can help the reader to see clearly that the objects are translated in the correct positions in the predicted next frame. Equally, these results show that the optical flow maps produced by the network correspond to a basic form of image segmentation into objects identified by their movement. Due to the built-in feedback loop, no labelled data is needed. Hence, this setup could potentially be more useful to pre-train a network for supervised video segmentation than a classic video autoencoder.

To highlight the advantages of using the proposed convolutional LSTM units when dealing with video sequences, we ran quantitative experiments using different architectures, summarised in Table 2. For example, *conv-1enc-1LSTM* corresponds to the vanilla version of our architecture, with one

Figure 5: Qualitative results on moving MNIST dataset (left) and moving MNIST with lines (right). In both cases, the first five columns show the last frames (out of 10 in total) fed into the network (from $t-4$ to t), the $t+1$ column shows the ground truth next frame, and the last two columns show the predicted next frame and the predicted optical flow.

convolutional-pooling-downsampling (CPD) module in the spatial encoder/decoder and one convolutional LSTM layer in the temporal encoder; *conv-2enc-2LSTM* corresponds to a deeper version of the architecture with 2 CPD modules in the spatial encoder/decoder and two convolutional LSTM layers in the temporal encoder; *fc-2enc-1LSTM* corresponds to a version of the architecture in which we replaced the convolutional LSTM layer with a fully-connected LSTM layer. We did not run experiments with only one CPD module in the spatial encoder/decoder with fully-connected LSTM, since flattening the spatial feature maps to feed in as input to the fully-connected LSTM would lead to a huge number of parameters, overpassing the hardware limits. Table 2 summarises the number of parameters for each architecture. Note the significant difference in the number of parameters between the convolutional and the fully-connected versions, making the former more appealing (at least) in terms of hardware constraints.

The loss function for this experiment was the binary cross-entropy (Srivastava et al., 2015), and the sequences are fed in as binary sequences. Figure 6 shows the evolution of the training errors for the above-mentioned architectures, and Table 2 presents the test errors. The versions of the architecture with convolutional LSTMs perform better than the versions using fully-connected LSTMs, due to their better ability to preserve the spatial local information, crucial for motion understanding. Equally, the results show that deeper architectures can be trained with our unsupervised setup, yield-

Architecture	Number of trainable parameters	Cross-entropy test error
conv-1enc-1LSTM	703,651	0.12827
conv-1enc-2LSTM	6,083,563	0.09116
conv-2enc-1LSTM	1,285,995	0.11810
conv-2enc-2LSTM	6,132,203	**0.08346**
fc-2enc-1LSTM	50,404,347	0.15648
fc-2enc-2LSTM	83,973,115	0.14484

Table 2: Number of parameters for different versions of the architecture (see text) and their next-frame prediction error on moving MNIST.

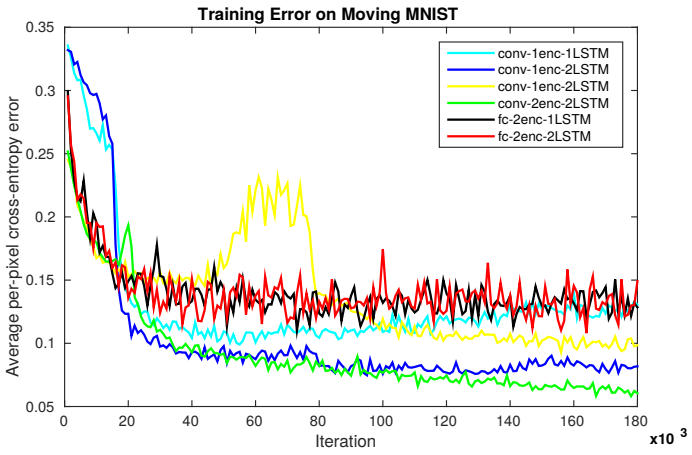

Figure 6: Average per-pixel training error on moving MNIST for different versions of the architecture.

ing better results than the vanilla architecture. For the fully-connected version, adding more layers does not seem to help, showing that it is not a suitable choice for our framework. Note that for all the architectures, the evolution of the training error is quite noisy, possibly because we feed in one sequence at a time (due to GPU memory constraints), hence one training sample, which makes the weight updates more noisy than in a mini-batch training regime. Another interesting observation is that for both models with small number of parameters (*conv-1enc-1LSTM* and *conv-2enc-1LSTM*), the error shows unexpected jumps towards the beginning of the training, whereas for the larger models the decrease is more steady. This might illustrate the fact that larger models are better at avoiding local minima.

We ran an additional experiment using only *conv-1enc-2LSTM*, to analyse the accuracy obtained when computing the loss using the difference-of-intensities function, the difference-of-gradients function, and a sum of both. In this experiment, the moving MNIST sequences were fed in as float sequences (not binary), and the loss function was the sum of squared differences. The L2 test errors are shown in Table 3. The difference-of-gradients performs slightly better than the difference-of-intensities; considering both terms leads to a lower error than considering any of the terms individually.

Architecture/Loss function	Difference-of-intensities	Difference-of-gradients	Sum of both
conv-1enc-2LSTM	0.03186	0.03095	0.02964

Table 3: Test L2-squared error on (float) moving MNIST for different loss functions.

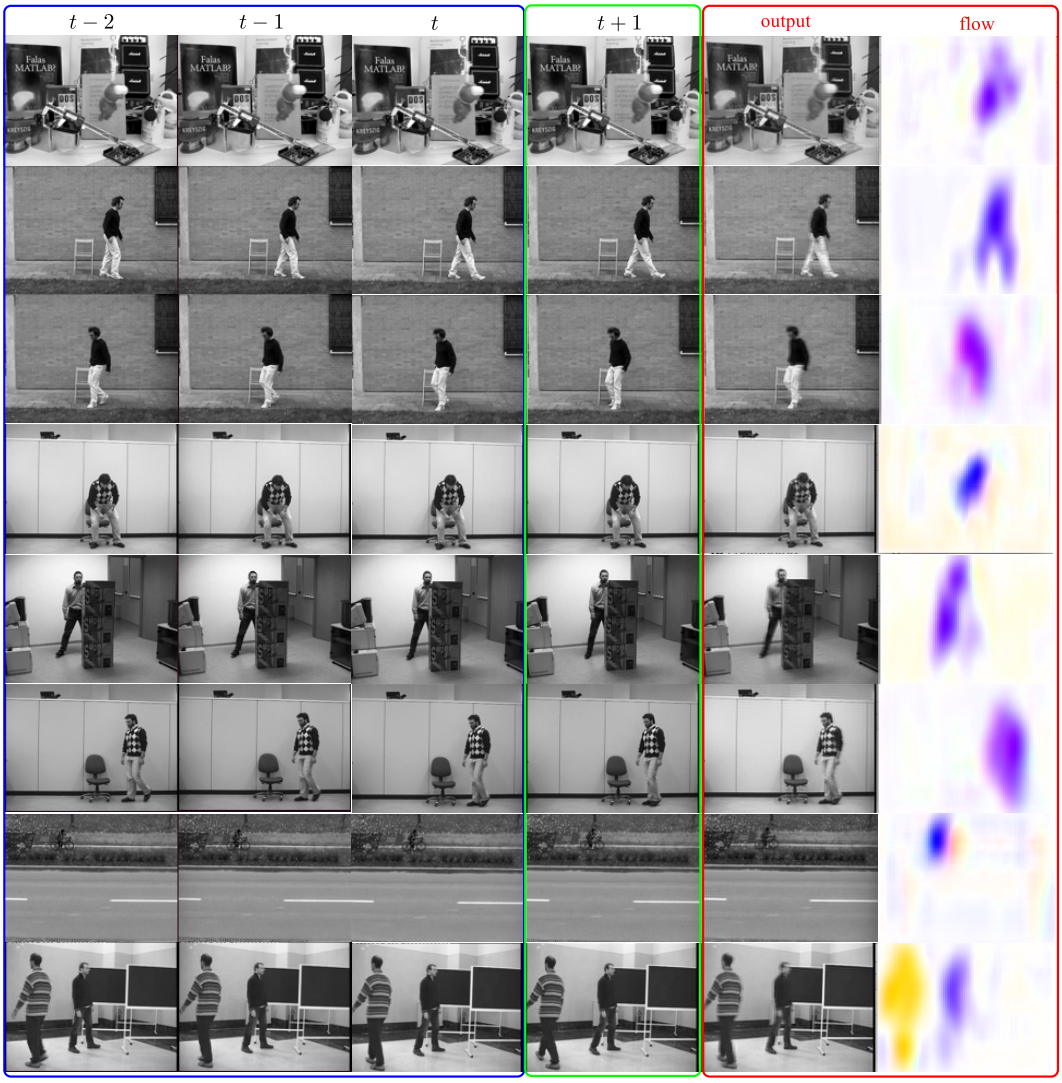

Figure 7: Qualitative results on real videos. The first three columns show the last frames (out of 12 in total) fed into the network (from $t-2$ to t), the $t+1$ column shows the ground truth next frame, and the last two columns show the predicted next frame and the predicted optical flow.

REAL DATASET

The real dataset used for training the architecture is formed by videos extracted from HMDB-51 (Kuehne et al., 2011) dataset, with about 107k frames in total (1.1 hours). Figure 7 shows qualitative results on different sequences extracted from PROST (Santner et al., 2010) and ViSOR (Vezzani & Cucchiara, 2010) datasets (see also Figure 4, third row, for sample frames). Videos showing the optical flow for entire test sequences are available at (Con). Note that since these sequences belong to different datasets than the training set, they can contain very different objects and motion models compared to the training set. The optical flow videos indicate that the moving elements of the scene are identified by the network. However, a flickering effect can be observed, pointing to the need of a recurrent module for flow regression to improve the temporal smoothness. Equally, an additional sparsity penalty on the flow map might help to clean up the hallucinated flow on the uniform regions in the background.

6 CONCLUSION AND FUTURE WORK

We proposed an original spatio-termporal video autoencoder based on an end-to-end differentiable architecture that allows unsupervised training for motion prediction. The core of the architecture is a module implementing a convolutional version of long short-term memory (LSTM) cells to be used as a form of artificial visual short-term memory. We showed that the proposed convolutional LSTM module performs better than existing fully-connected implementations, while having a reduced number of parameters.

We believe that our work can open up the path to a number of exciting directions. Due to the built-in feedback loop, various experiments can be carried out effortlessly, to develop further the basic memory module that we proposed. In particular, investigating on the size of the memory and its resolution/downsampling could shed some light into the causes of some geometric optical illusions; *e.g.* when storing in memory an overly-downsampled version of the visual input, this could affect the capacity of correctly perceiving the visual stimuli, in accordance with Shannon-Nyquist sampling theorem (Shannon, 1949). Equally, we hypothesise that our convolutional version of LSTM cells can lead to ambiguous memory activations, *i.e.* the same activation would be produced when presenting a temporally moving boundary or a spatially repeated boundary. This could imitate the illusory motion experienced by biological VSTMs, *i.e.* static repeated patterns that produce a false perception of movement (Conway et al., 2005).

A necessary future development is the integration of the memory module with an attention mechanism and a form of long-term memory module, to enable a complete memory system, indispensable for supervised tasks.

Last but not least, the proposed architecture composed of memory module and built-in feedback loop could be applied for static images as a compression mechanism, similar to the inspirational work on jigsaw video compression (Kannan et al., 2007). The memory addressing in that case could use a content-based mechanism similar to NTM (Graves et al., 2014).

ACKNOWLEDGMENTS

We are greatly indebted to the *Torch* community for their efforts to maintain this great library, and we are grateful to CSIC–University of Cambridge for funding this work (grant number EP/L010917/1).

REFERENCES

Convlstm. `https://github.com/viorik/ConvLSTM`.

Pulkit Agrawal, João Carreira, and Jitendra Malik. Learning to see by moving. *CoRR*, abs/1505.01596, 2015. URL `http://arxiv.org/abs/1505.01596`.

B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11), 2012. doi: 10.3758/BF03203943.

Reneé Baillargeon. Infants' physical world. *American Psychological Society*, 13(3), 2004.

Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *Int. J. Comput. Vision*, 92(1):1–31, 2011.

K. Boahen. Neuromorphic microchips. *Scientific American*, 292(5), 2005.

Thomas Brox, Andrs Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision - ECCV 2004*, volume 3024 of *Lecture Notes in Computer Science*, pp. 25–36. 2004.

D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conf. on Computer Vision (ECCV)*, pp. 611–625, 2012.

Ronan Collobert, Koray Kavukcuoglu, and Clment Farabet. Torch7: A matlab-like environment for machine learning.

Bevil R. Conway, Akiyoshi Kitaoka, Arash Yazdanbakhsh, Christopher C. Pack, and Margaret S. Livingstone. Neural basis for a powerful static motion illusion. *The Journal of Neuroscience*, 25 (23):5651–5656, 2005.

Douglas Eck and Jürgen Schmidhuber. A first look at music composition using lstm recurrent neural networks. Technical Report IDSIA-07-02, IDSIA / USI-SUPSI.

David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015. URL http://arxiv.org/abs/1504.06852.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Multi-dimensional recurrent neural networks. In *Artificial Neural Networks*, volume 4668, pp. 549–558. 2007.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Sepp Hochreiter, Fakultat F. Informatik, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. In *Field Guide to Dynamical Recurrent Networks*. 2000.

A. Hollingworth. Constructing visual representations of natural scenes: the roles of short- and long-term visual memory. *J Exp Psychol Hum Percept Perform.*, 30(3), 2004.

Berthold K. P. Horn and B. G. Schunck. Artificial intelligence in perspective. chapter Determining Optical Flow: A Retrospective, pp. 81–87. 1994.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015. URL http://arxiv.org/abs/1506.02025.

Anitha Kannan, John Winn, and Carsten Rother. Clustering appearance and shape by learning jigsaws. In *Advances in Neural Information Processing Systems*, pp. 657–664. 2007.

Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoï Ieng, and Andrew J. Davison. Simultaneous mosaicing and tracking with an event camera. In *British Machine Vision Conference (BMVC)*. BMVC 2014, 2014.

H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

Nicholas Léonard, Sagar Waghmare, Yang Wang, and Jin-Hwa Kim. rnn : Recurrent library for torch. *CoRR*, abs/1511.07889, 2015.

Svein Magnussen. Low-level memory processes in vision. *Trends in Neurosciences*, 23(6):247–251, 2000. ISSN 0166-2236. doi: http://dx.doi.org/10.1016/S0166-2236(00)01569-1. URL http://www.sciencedirect.com/science/article/pii/S0166223600015691.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning*, volume 6791, pp. 52–59. 2011.

Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3061–3070, 2015.

W.A. Phillips. On the distinction between sensory storage and short-term visual memory. *Perception & Psychophysics*, 16(2):283–290, 1974. ISSN 0031-5117. doi: 10.3758/BF03203943. URL http://dx.doi.org/10.3758/BF03203943.

Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128, 2014. URL http://arxiv.org/abs/1402.1128.

Jakob Santner, Christian Leistner, Amir Saffari, Thomas Pock, and Horst Bischof. PROST Parallel Robust Online Simple Tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA, 2010.

C.E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, Jan 1949. ISSN 0096-8390. doi: 10.1109/JRPROC.1949.232969.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, abs/1506.04214, 2015.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *ICML*, 2015.

Marijn Stollenga, Wonmin Byeon, Marcus Liwicki, and Jürgen Schmidhuber. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. *CoRR*, abs/1506.07452, 2015. URL http://arxiv.org/abs/1506.07452.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *NIPS*, 2014.

Roberto Vezzani and Rita Cucchiara. Video surveillance online repository (visor): an integrated framework. *Multimedia Tools and Applications*, 50(2):359–380, 2010.

P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pp. 1385–1392, Dec 2013. doi: 10.1109/ICCV.2013.175.

Manuel Werlberger, Werner Trobin, Thomas Pock, Andreas Wedel, Daniel Cremers, and Horst Bischof. Anisotropic Huber-L1 optical flow. In *Proceedings of the British Machine Vision Conference (BMVC)*, London, UK, September 2009.

Ronald J. Williams and David Zipser. Backpropagation. chapter Gradient-based Learning Algorithms for Recurrent Networks and Their Computational Complexity, pp. 433–486. 1995.