

RATM: RECURRENT ATTENTIVE TRACKING MODEL

Samira Ebrahimi Kahou

École Polytechnique de Montréal, Canada

samira.ebrahimi-kahou@polymtl.ca

Vincent Michalski

Université de Montréal, Canada

vincent.michalski@umontreal.ca

Roland Memisevic

Université de Montréal, Canada

roland.memisevic@umontreal.ca

ABSTRACT

This work presents an attention based approach to tracking objects in video. A recurrent neural network is trained to predict the position of an object in the video at time $t+1$ given a series of selective glimpses at times 1 to t . Glimpses are selected based on a differentiable (soft-)attention mechanism, which makes it possible to train the model end-to-end using standard stochastic gradient descent. Experiments on artificial data-sets show the importance of various design choices and that the model is able to perform simple tracking tasks.

1 INTRODUCTION

Object tracking is a classic computer vision task in which the goal is to follow the position of an object in a video. Traditional work on object tracking typically uses sampling methods to select and evaluate multiple window candidates based on similarity in feature space (Smeulders et al., 2014). In this work we introduce a tracking method that predicts the position of an object at time t efficiently without searching within the frame at time t .

The proposed recurrent attentive tracking model (RATM) employs a modified version of the fully differentiable attention mechanisms proposed recently by Gregor et al. (2015). In that work, the attention mechanism can selectively read and write in images in order to generate images sequentially. Interestingly, the experiments in Gregor et al. (2015) on handwritten digits show that the read mechanism learns to trace contours and the write mechanism to generate digits using the continuous motion of the “write” window. This hints at the potential of this method in tracking applications, where the primary goal is to trace out the spatio-temporal “contours” that an object follows as it moves in a video.

Previous work that investigates the application of attention mechanisms for tracking with neural networks includes (Denil et al., 2011) and references therein. In contrast to that line of work, we develop a fully-integrated neural framework, that can be trained end-to-end using back-propagation. Most importantly, we show that combining a differentiable attention mechanism with suitable surrogate costs to provide a supervised learning signal makes it possible to train attention policies using simple supervised back-prop and without resorting to reinforcement learning or sampling methods.

2 METHOD

The proposed model mainly consists of two components: an attention mechanism that extracts patches from the input images and a recurrent neural network (RNN), which predicts attention parameters, i.e. it predicts where to look in the next frame. Sections 2.1 and 2.2 describe the attention mechanism and the RNN, respectively. The proposed architecture is described in Section 2.3.

2.1 ATTENTION

Attention mechanisms allow neural networks to focus on task-relevant information. They have been successfully applied to a variety of vision and natural language processing problems (Mnih et al., 2014; Xu et al., 2015; Rush et al., 2015). Such mechanisms can be useful in tracking applications by helping to narrow down the region-of-interest given the past trajectory of an object. The attention mechanism introduced in Gregor et al. (2015) extracts glimpses from the input data by applying a grid of 2D Gaussian window filters. Each filter response corresponds to one pixel of the glimpse. Exploiting the separability of 2D Gaussian windows, this attention mechanism constructs an $N \times N$ grid of two-dimensional Gaussian filters by sequentially applying a set of row filters and a set of column filters. The grid has 4 parameters¹, the grid center g_X, g_Y , the isotropic standard deviation σ and the stride between grid points δ . For the filter at row i , column j in the attention patch, the mean locations μ_X^i, μ_Y^j are computed as follows:

$$\mu_X^i = g_X + (i - \frac{N}{2} - 0.5) \cdot \delta, \quad \mu_Y^j = g_Y + (j - \frac{N}{2} - 0.5) \cdot \delta \quad (1)$$

The parameters are dynamically generated as a linear transformation of a vector \mathbf{h} :

$$(\tilde{g}_X, \tilde{g}_Y, \tilde{\sigma}, \tilde{\delta}) = \mathbf{W}\mathbf{h}, \quad (2)$$

followed by normalization of the parameters:

$$g_X = \frac{\tilde{g}_X + 1}{2}, \quad g_Y = \frac{\tilde{g}_Y + 1}{2}, \quad \delta = \frac{\max(A, B) - 1}{N - 1} \cdot |\tilde{\delta}|, \quad \sigma = |\tilde{\sigma}|, \quad (3)$$

where A and B are the width and height of the input image.

In contrast to the original implementation in Gregor et al. (2015), we use the absolute value function $\text{abs}(x) = |x|$ to ensure positivity of δ and σ . The intuition behind this choice is the following: In our experiments we observed that the stride and standard deviation parameters tend to saturate at low values, causing the attention window to zoom in into a single pixel. This effectively inhibited gradient flow through neighboring pixels of the attention filters. By replacing the exponential function with the absolute value, we introduce a discontinuity at $x = 0$. Piecewise linear activation functions have been shown to benefit optimization (Nair & Hinton, 2010) and the absolute value function is a convenient trade-off between the harsh zeroing of all negative inputs of the Rectified Linear Unit (ReLU) and the extreme saturation for highly negative inputs of the exponential function.

The filters which are used to extract a patch \mathbf{p} from an image \mathbf{x} are computed as follows:

$$\mathbf{F}_X[i, a] = \frac{1}{Z_X} \exp\left(-\frac{(a - \mu_X^i)^2}{2\sigma^2}\right), \quad \mathbf{F}_Y[j, b] = \frac{1}{Z_Y} \exp\left(-\frac{(b - \mu_Y^j)^2}{2\sigma^2}\right) \quad (4)$$

In Equation 4, a and b correspond to column and row indices in the input image. The $N \times N$ patch \mathbf{p} is then extracted from the image \mathbf{x} by separate application of both filters:

$$\mathbf{p} = \mathbf{F}_Y \mathbf{x} \mathbf{F}_X^T. \quad (5)$$

An example of the patch extraction is shown in Figure 1 on the right side.

2.2 RECURRENT NEURAL NETWORK

Advances in the optimization of RNNs enable their successful application in a wide range of tasks (Sutskever et al., 2014; Cho et al., 2014; Hochreiter & Schmidhuber, 1997). The most simple RNN consists of one input, one hidden and one output layer. In time-step t , the network, based on the input frame \mathbf{x}_t and the previous hidden state \mathbf{h}_{t-1} , computes the new hidden state

$$\mathbf{h}_t = \sigma(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{W}_{rec}\mathbf{h}_{t-1}), \quad (6)$$

where σ is a non-linear activation function, \mathbf{W}_{in} is the input and \mathbf{W}_{rec} is the recurrent weight matrix. While the application of the more sophisticated gated recurrent units (Cho et al., 2014)

¹The original read mechanism from Gregor et al. (2015) also adds a scalar intensity parameter, that is multiplied to filter responses.

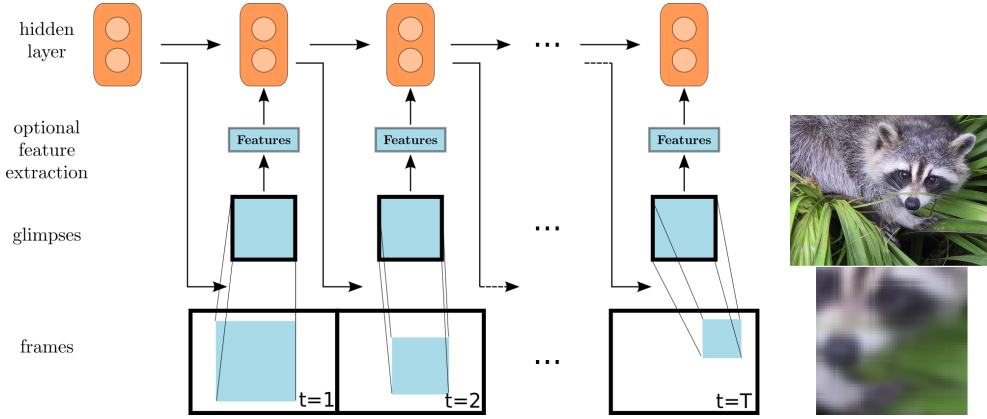


Figure 1: Left: Layout of the recurrent attentive tracking model unrolled in time. Hidden units at time t are mapped to parameters of the attention mechanism, yielding a glimpse into the input frame at time $t + 1$. Right: A patch is extracted from the full image on top.

and long-short-term memory (Hochreiter & Schmidhuber, 1997) is very common in recent work, we rely on a simpler variant of recurrent networks, called IRNN (Le et al., 2015), in which \mathbf{W}_{rec} is initialized with a scaled version of the identity matrix and σ in Equation 6 corresponds to the element-wise ReLU activation function (Nair & Hinton, 2010)

$$\sigma(x) = \max(0, x). \quad (7)$$

In our experiments we use the Theano-based (Bastien et al., 2012; Bergstra et al., 2010) implementation of the IRNN released with Ebrahimi Kahou et al. (2015).

2.3 RECURRENT ATTENTIVE TRACKING MODEL

The task of tracking involves the mapping of a sequence of inputs $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T)$ to a sequence of locations $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_T)$. Given that for the prediction $\hat{\mathbf{y}}_t$ of an object’s location at time t , the trajectory $(\mathbf{y}_1, \dots, \mathbf{y}_{t-1})$ usually contains highly relevant contextual information, it is suitable to choose a hidden state model which has the capacity of building a representation of this trajectory. The proposed RATM is a fully differentiable neural network, consisting of the described attention mechanism and an RNN. In each time step the RNN’s hidden state vector \mathbf{h}_t is mapped to window parameters according to Equations 2 and 3. In the next time step the resulting image patch returned by the attention mechanism is passed as input \mathbf{x}_{t+1} to the RNN, which computes \mathbf{h}_{t+1} according to Equation 6. A sketch of the model is shown in Figure 1 on the left.

Instead of passing the raw input patch to the RNN, a pre-trained model can be used to extract image features, e.g. a convolutional neural network (CNN, Fukushima (1980); LeCun et al. (1998)). Figure 2 shows how a feature extractor can be integrated into RATM. Our experiment on MNIST videos can be seen as a special case of this hybrid architecture, in which the selected feature layer corresponds to the input layer of the CNN. The CNN can also be used to generate a target prediction for classification. Note that the size of this feature model affects memory consumption and/or computation speed of the whole model, especially during training, because gradients have to be back-propagated through its instances in all time-steps. In Section 3, we explore various ways of generating a training signal for gradient descent training of RATM.

3 EXPERIMENTS

In this section we present some of the experiments with RATM and comment on qualitative results. All experiments use training sets of size 100,000 and test sets with 10,000, gradient clipping of 1.0 and a mini-batch size of 32 or 16.

Our main focus was on designing an appropriate cost function for each experiment. The choice of hyper-parameters, including weight initialization, was made to stabilize training.

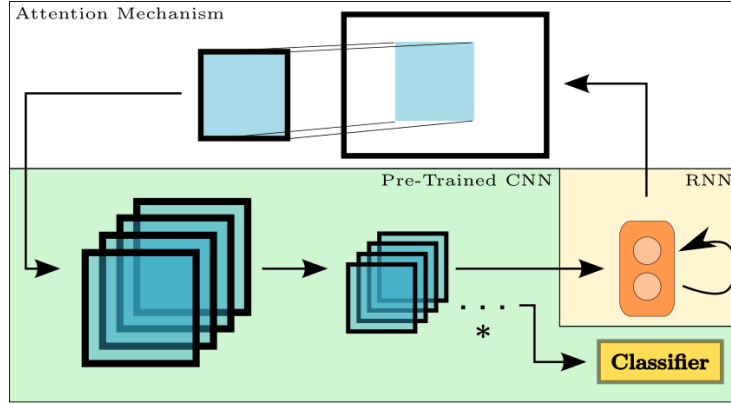


Figure 2: Layout of RATM with a pre-trained CNN for feature extraction and classification. The star indicates intermediate layers between the selected feature layer and the output layer.

The code for these experiments, including data generation, will be made available online².

3.1 BOUNCING BALLS

For our initial experiment, we used the bouncing balls data-set (Sutskever et al., 2009) with 32 frames of resolution 20×20 . We train RATM with 64 hidden units in the RNN and feed the raw 5×5 attention patch as input to the RNN. For learning we use stochastic gradient descent (SGD) with a learning rate of 0.01, without momentum for 200 epochs. The initial parameters of the attention mechanism are set such that the resulting first patch roughly covers the whole image. In the objective function, we minimized only the mean-squared error between the selected patch in the last of the 32 frames and a target patch which is simply a cropped white ball image as the shape and color of the object is constant across the whole data-set. We also trained a version with this penalty applied to every frame. An example sequence is shown in the supplementary material. The dynamics in this data-set are very limited and even though the tracking looks accurate, it can be explained by the RNNs potential for memorizing the set of possible trajectories. For this reason, we experimented with more challenging data-sets. Figure 3 shows the results of tracking a ball in a test sequence.

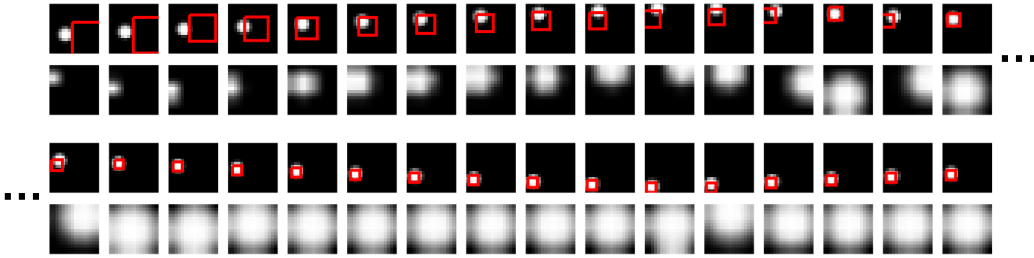


Figure 3: An example of tracking on the bouncing balls data-set. The first row shows the first 16 frames of the sequence with a red rectangle indicating the location of the attention patch. The second row contains the extracted patches. The third and fourth row show the continuation of the sequence.

3.2 MNIST

To increase the difficulty of the tracking task, we move to more challenging data-sets, containing more than a single type of object (10 digits) and less predictable dynamics. We generate videos from 28×28 MNIST digit images (LeCun et al., 1998) by placing them in a larger 100×100 canvas with black background and translating them from frame-to-frame. In both of these experiments we train

²<https://github.com/saebrahimi/RATM>

RATM with 100 hidden units in the RNN using SGD with a learning rate of 0.001 and momentum of 0.9 for 500 epochs.

3.2.1 PRE-TRAINED FEATURE EXTRACTOR AND CLASSIFIER

We were able to generate a reliable training signal using pixel-based similarity in the bouncing balls experiment. However, the variation in the MNIST data-set requires a representation that is robust against small variations to guide training. As an objective function we used the average classification error provided by a pre-trained shallow CNN, that classifies the patches of each time-step. This CNN is trained on MNIST digits and its parameters remained fixed during RATM training. The CNN structure has two convolutional layers with $32\ 5\times 5$ filters, each followed by a 2×2 maxpooling layer, 0.25 dropout (Hinton et al., 2012), and the ReLU activation function. These layers are followed by a 10-unit softmax layer for classification.

3.2.2 SINGLE-DIGIT

To better understand the learning ability of RATM, we generated a new data-set by moving one digit randomly drawn from MNIST. We respected the same data split for training and testing as in the original MNIST data-set, i.e. digits were drawn from the training split to generate training sequences and testing split for testing. The generated videos have frame size 100×100 and show a digit moving in a random walk with momentum against a black background. As in the bouncing balls experiment in Section 3.1, the initial patch roughly covers the whole image. This MNIST experiment is different from bouncing balls tracking since digits are from different classes which the model has to recognize. The model failed to learn only based on the classification term. To amend this, we initially tried to add a penalty on the distance between corners of the ground truth bounding boxes and the corners of the attention grid. This again led to unsatisfactory results and we changed the localization cost to the mean-squared error between ground truth center coordinates and patch center. This modification helped the model to reliably detect and track the digits. The localization term helped in the early stages of training to guide RATM to track the digits, whereas the classification term encourages the model to properly zoom into the image and maximize classification accuracy. Training is done on sequences with only 10 frames. The classification and localization penalties were applied at every time-step. At test time, the CNN is switched off and we let the model track test sequences of 30 frames. Figure 4 shows one test sample.

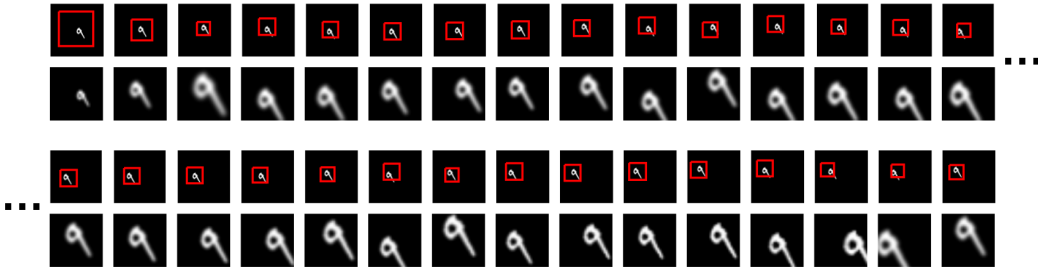


Figure 4: Tracking one digit. The first and second row show the sequence and corresponding extracted patches, respectively. The red rectangle indicates the location of the patch in the frame. The third and fourth row are the continuation. Prediction works well far beyond the training horizon of 10 frames.

3.2.3 MULTI-DIGIT

It was interesting to see how robust RATM is in presence of another moving digit in the background. Thus, we generated new sequences by modifying the bouncing balls script released with Sutskever et al. (2009). The balls were replaced by randomly drawn MNIST digits. We also added a random walk with momentum to the motion vector. The cost function and settings are the same as in the single digit experiment, except that we set the bias of the attention mechanism such that the initial window is centered on the digit to be tracked. The model was trained on 10 frames and was able to

Table 1: Average Intersection-over-Union scores on test data.

Experiment	Average IoU (over # frames)
Bouncing Balls (training penalty only on last frame)	69.15 (1, only last frame)
Bouncing Balls (training penalty only on last frame)	54.65 (32)
Bouncing Balls (training penalty on all frames)	66.86 (32)
MNIST (single-digit)	63.53 (30)
MNIST (multi-digit)	51.62 (30)

focus on digits at least for 15 frames on test data. Figure 5 shows tracking results on a test sequence.

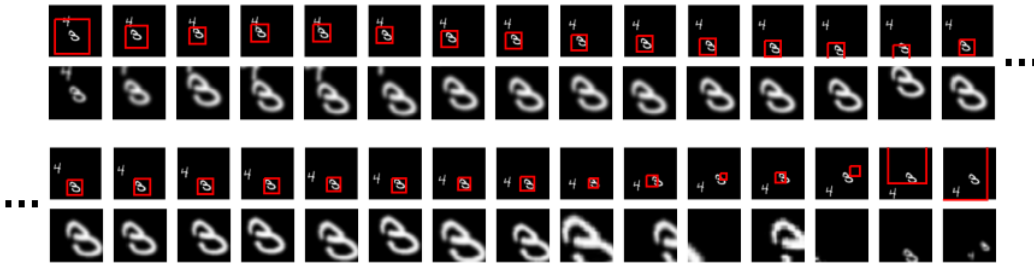


Figure 5: Tracking one of two digits. The first and second row show the sequence and corresponding extracted patches, respectively. The red rectangle indicates the location of the patch in the frame. The third and fourth row are the continuation. Prediction works well for sequences twice as long as the training sequences with 10 frames.

4 EVALUATION

For our experiments on artificial data, bounding box size information was not given to the model. However, our cost term based on visual similarity, average pixel distance in the case of bouncing balls and a classification error for MNIST, encourages the attention mechanism to find proper parameters for patch selection.

As a quantitative measure, we evaluate the tracking results on test data using the average Intersection-over-Union (IoU) (Everingham et al., 2010)

$$IoU = \frac{|B_{gt} \cap B_{pred}|}{|B_{gt} \cup B_{pred}|}, \quad (8)$$

where B_{gt} and B_{pred} are the ground truth and predicted bounding boxes. A predicted bounding box is defined as the rectangle between the corner points of the attention grid. This definition of predicted bounding boxes ignores the fact that each point in the attention patch is a weighted sum of pixels around the grid points and the boxes are smaller than the region selected by the attention mechanism. This affects the performance under the average IoU metric. However, it still serves as a reasonable metric for the soft attention mechanism. We report the average per-frame IoU for a test set. We did not exclude first frames from this evaluation, although their bounding boxes are fixed and were not learned.

The average test IoU measures for different experiments are listed in Table 1. In the supplementary material we also provide a plot of average IoUs as a function of sequence length for the MNIST experiments.

5 CONCLUSION AND FUTURE WORK

In this paper we introduced a recurrent neural network augmented with an attention mechanism for tracking. We evaluated our proposed model on artificial data-sets and the qualitative and quantitative results show that the model can learn to adapt to these tasks. We highlighted some challenges, for instance the design of an appropriate cost function. In the experiment on bouncing balls, RATM learned the trajectory between the first and last frame, based only on a penalty applied in the 32nd frame. For the MNIST experiment, we decided on a classification term as a measure of object similarity that is more flexible than raw pixel values. That alone did not work and we had to add a localization cost, based on the mean-squared error between the attention patch center and the ground truth object centers.

We are working on extending this work towards real-world data-sets. We experimented with RATM on the Caltech Pedestrian Dataset (Dollár et al., 2009; 2012), but the number of long annotated training sequences which contain the same subject is quite low for deep learning standards. Moreover, due to the high frame rate motion is very limited, so the model learned to predict almost a static window. The results were inconclusive and are not presented here.

As a long-term project goal, we plan to explore multi-object tracking and to investigate the representation learned by the network. In particular, we will focus on the model’s ability to deal with occlusions and to compress object identity and motion trajectory in its hidden state.

ACKNOWLEDGMENTS

The authors would like to thank the developers of Theano (Bastien et al., 2012; Bergstra et al., 2010). We thank Jörg Bornschein and Pierre-Luc St-Charles for helpful discussions. This work was supported by an NSERC Discovery Award and the German BMBF, project 01GQ0841. We also thank the Canadian Foundation for Innovation (CFI) for support under the Leaders program.

REFERENCES

- Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Bergstra, James, Goodfellow, Ian, Bergeron, Arnaud, Bouchard, Nicolas, Warde-Farley, David, and Bengio, Yoshua. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.
- Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, pp. 3. Austin, TX, 2010.
- Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- Denil, Misha, Bazzani, Loris, Larochelle, Hugo, and de Freitas, Nando. Learning where to attend with deep architectures for image tracking. *CoRR*, abs/1109.3737, 2011. URL <http://arxiv.org/abs/1109.3737>.
- Dollár, Piotr, Wojek, Christian, Schiele, Bernt, and Perona, Pietro. Pedestrian detection: A benchmark. In *CVPR*, June 2009.
- Dollár, Piotr, Wojek, Christian, Schiele, Bernt, and Perona, Pietro. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 34, 2012.
- Ebrahimi Kahou, Samira, Michalski, Vincent, Konda, Kishore, Memisevic, Roland, and Pal, Christopher. Recurrent neural networks for emotion recognition in video. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, ICMI ’15*, pp. 467–474, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3912-4. doi: 10.1145/2818346.2830596. URL <http://doi.acm.org/10.1145/2818346.2830596>.
- Everingham, Mark, Van Gool, Luc, Williams, Christopher KI, Winn, John, and Zisserman, Andrew. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

- Fukushima, Kunihiro. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.
- Gregor, Karol, Danihelka, Ivo, Graves, Alex, and Wierstra, Daan. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623, 2015. URL <http://arxiv.org/abs/1502.04623>.
- Hinton, Geoffrey E, Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.
- Le, Quoc V, Jaitly, Navdeep, and Hinton, Geoffrey E. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.
- LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Mnih, Volodymyr, Heess, Nicolas, Graves, Alex, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pp. 2204–2212, 2014.
- Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.
- Rush, Alexander M, Chopra, Sumit, and Weston, Jason. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.
- Smeulders, Arnold W M, Chu, Dung M, Cucchiara, Rita, Calderara, Simone, Dehghan, Afshin, and Shah, Mubarak. Visual tracking: An experimental survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(7):1442–1468, July 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.230.
- Sutskever, Ilya, Hinton, Geoffrey E, and Taylor, Graham W. The recurrent temporal restricted boltzmann machine. In *Advances in Neural Information Processing Systems*, pp. 1601–1608, 2009.
- Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.
- Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Courville, Aaron, Salakhutdinov, Ruslan, Zemel, Richard, and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

6 SUPPLEMENTARY MATERIAL

6.1 BOUNCING BALLS

The bouncing balls experiment was also performed with the mean-squared error penalty applied in every time-step. The attention mechanism learned to find the ball early in the sequence and tracks it reliably as shown in Figure 6.

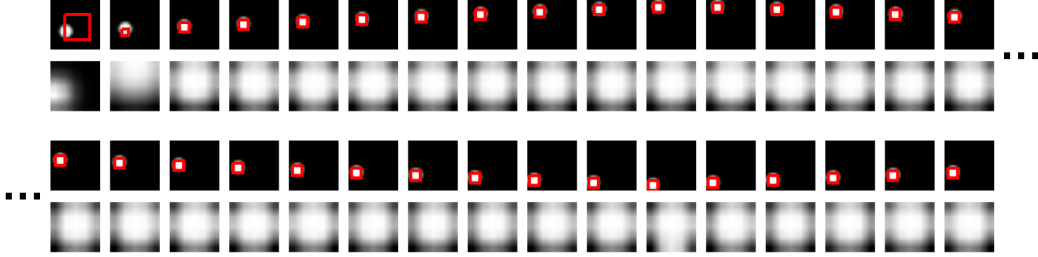


Figure 6: Tracking result on a test sequence from a model trained with a mean-squared error penalty for every time-step. The first row shows the first 16 frames of the sequence with a red rectangle indicating the location of the attention patch. The second row contains the extracted patches. The third and fourth row show the continuation of the sequence.

6.2 MNIST TRACKING

To show the performance of our MNIST tracking models, we plot the test IoU measure as a function of sequence length in Figure 7. These plots begin with a very low value due to the fixed sub-optimal initialization in the first frame. The multi-digit tracking is more difficult and the performance drops for longer test sequences.

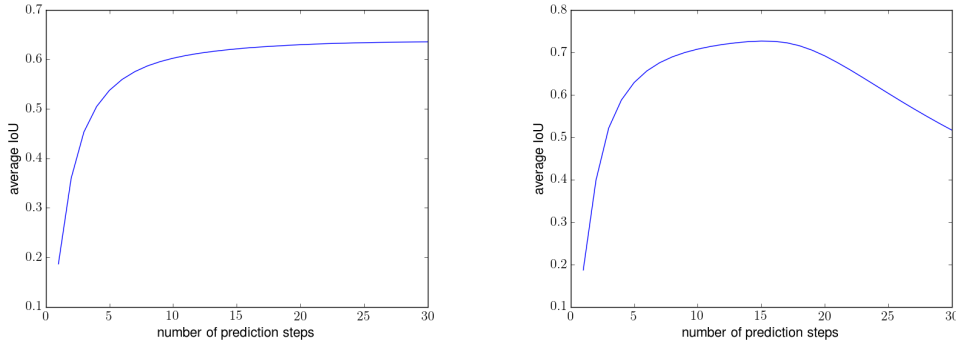


Figure 7: Average test IoU as a function of sequence length. Left: Single-digit data-set. Right: Multi-digit data-set.