RATM: Recurrent Attentive Tracking Model

Samira Ebrahimi Kahou[1], Vincent Michalski[2], Roland Memisevic[2], Christopher Pal[1], Pascal Vincent[2]
[1]École Polytechnique de Montréal, [2]Université de Montréal
{samira.ebrahimi-kahou,christopher.pal}@polymtl.ca, vincent.michalski@umontreal.ca, {memisevr,vincentp}@iro.umontreal.ca

Abstract

We present an attention-based modular neural framework for computer vision. The framework uses a soft attention mechanism allowing models to be trained with gradient descent. It consists of three modules: a recurrent attention module controlling where *to look in an image or video frame, a feature-extraction module providing a representation of* what *is seen, and an objective module formalizing* why *the model learns its attentive behavior. The attention module allows the model to focus computation on task-related information in the input. We apply the framework to several object tracking tasks and explore various design choices. We experiment with three data sets, bouncing ball, moving digits and the real-world KTH data set. The proposed Recurrent Attentional Tracking Model (RATM) performs well on all three tasks and can generalize to related but previously unseen sequences from a challenging tracking data set.*

1. Introduction

Attention mechanisms are one of the biggest trends in deep-learning research and have been successfully applied in a variety of neural-network architectures across different tasks. In computer vision, for instance, attention mechanisms have been used for image generation [13] and image captioning [40]. In natural language processing they have been used for machine translation [1] and sentence summarization [27]. And in computational biology attention was used for subcellular protein localization [32].

In these kinds of applications usually not all information contained in the input data is relevant for the given task. Attention mechanisms allow the neural network to focus on the relevant parts of the input, while ignoring other, potentially distracting, information. Besides enabling models to ignore distracting information, attention mechanisms can be helpful in streaming data scenarios, where the amount of data per frame can be prohibitively large for full processing. In addition, some studies suggest that there is a representational advantage of sequential processing of image parts over a single pass over the whole image (see for example [22, 18, 13, 7, 26, 29]).

Recently, [13] introduced the Deep Recurrent Attentive Writer (DRAW), which involves a Recurrent Neural Network (RNN) that controls a read and a write mechanism based on attention. The read mechanism extracts a parametrized window from the static input image. Similarly, the write mechanism is used to write into a window on an output canvas. This model is trained to sequentially produce a reconstruction of the input image on the canvas. Interestingly, one of the experiments on handwritten digits showed that the read mechanism learns to trace digit contours and the write mechanism generates digits in a continuous motion. This observation hints at the potential of such mechanisms in visual object tracking applications, where the primary goal is to trace the spatio-temporal "contours" of an object as it moves in a video.

Previous work on the application of attention mechanisms for tracking includes [7] and references therein. In contrast to that line of work, we propose a model based on a fully-integrated neural framework, that can be trained end-to-end using back-propagation. The framework consists of three modules: a recurrent differentiable attention module controlling *where* to look in an image, a feature-extraction module providing a representation of *what* is seen, and an objective module formalizing *why* the model learns its attentive behavior. As we shall show, a suitable surrogate cost in the objective module can provide a supervised learning signal, that allows us to train the network end-to-end, and to learn attentional strategies using simple supervised back-prop without resorting to reinforcement learning or sampling methods.

According to a recent survey of tracking methods [30], many approaches to visual tracking involve a search over multiple window candidates based on a similarity measure in a feature space. Successful methods involving deep learning, such as [24], perform tracking-by-detection, e.g. by using a Convolutional Neural Network (CNN) for foreground-background classification of region proposals. As in most approaches, the method in [24] at each time step samples a number of region proposals (256) from a

Gaussian distribution centered on the region of the previous frame. Such methods do not benefit from useful correlations between the target location and the object's past trajectory. There are deep-learning approaches that consider trajectories by employing particle filters such as [38], which still involves ranking of region proposals ($1,000$ particles). More recently, Siamese networks [3] have been employed to compute the matching of proposal windows [36, 2]. The latter work achieves beyond real-time performance by using a fully-convolutional Siamese architecture to compute a map of support in a single evaluation per frame.

In our RATM, an RNN predicts the position of an object at time t, given a real-valued hidden state vector. The state vector can summarize the history of observations and predictions of previous time steps. We rely on a *single* prediction per time step instead of using the predicted location as basis for a search over multiple region proposals. This allows for easy integration of our framework's components and training with simple gradient-based methods. Since the initial version of this work, [21] have developed a similar approach to action recognition, the main difference being the use of a convolutional Long Short-Term Memory (LSTM) for better modeling of spatial structure and guidance of the attention by optical flow features.

The main contribution of our work is the introduction of a modular neural framework, that can be trained end-to-end with gradient-based learning methods. Using object tracking as an example application, we explore different settings and provide insights into model design and training. While the proposed framework is targeted primarily at videos, it can also be applied to sequential processing of still images.

2. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are powerful machine learning models that are used for learning in sequential processing tasks. Advances in understanding the learning dynamics of RNNs enabled their successful application in a wide range of tasks (for example [15, 25, 12, 35, 5, 33]).

In each time step t, the network computes a new hidden state $\mathbf{h}_t$ based on the previous state $\mathbf{h}_{t-1}$ and the input $\mathbf{x}_t$:

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

where σ is a non-linear activation function, $\mathbf{W}_{in}$ is the matrix containing the input-to-hidden weights and $\mathbf{W}_{rec}$ is the recurrent weight matrix from the hidden layer to itself. At each time step the RNN also generates an output

$$\mathbf{y}_t = \mathbf{W}_{out}\mathbf{h}_t + \mathbf{b}_y, \quad (2)$$

where $\mathbf{W}_{out}$ is the matrix with weights from the hidden to the output layer.

Although the application of recurrent networks with sophisticated hidden units, such as LSTM [15] or Gated Recurrent Unit (GRU) [5], has become common in recent years (for example [1, 35, 33]), we rely on the simple IRNN proposed by [19], and show that it works well in the context of visual attention. The IRNN corresponds to a standard RNN, where recurrent weights $\mathbf{W}_{rec}$ are initialized with a scaled version of the identity matrix and the hidden activation function $\sigma(.)$ is the element-wise Rectified Linear Unit (ReLU) function [23]. The initial hidden state $\mathbf{h}_0$ is initialized as the zero vector. Our experiments are based on the Theano [37] implementation of the IRNN shown to work well for video in [8].

3. Neural Attention Mechanisms

Our attention mechanism is a modification of the read mechanism introduced in [13]. It extracts *glimpses* from the input image by applying a grid of two-dimensional Gaussian window filters. Each of the filter responses corresponds to one pixel of the glimpse. An example of the glimpse extraction is shown in Figure 1.

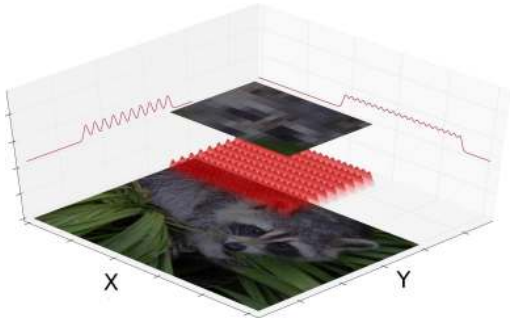

Figure 1: A 20×10 glimpse is extracted from the full image by applying a grid of 20×10 two-dimensional Gaussian window filters. The separability of the multi-dimensional Gaussian window allows for efficient computation of the extracted glimpse.

Given an image $\mathbf{x}$ with A columns and B rows, the attention mechanism separately applies a set of M column filters $\mathbf{F}_X \in \mathbb{R}^{M \times A}$ and a set of N row filters $\mathbf{F}_Y \in \mathbb{R}^{N \times B}$, extracting an $M \times N$ glimpse $\mathbf{p} = \mathbf{F}_Y \mathbf{x} \mathbf{F}_X^{\mathrm{T}}$. This implicitly computes $M \times N$ two-dimensional filter responses due to the separability of two-dimensional Gaussian filters. For multi-channel images the same filters are applied to each channel separately. The sets of one-dimensional row ($\mathbf{F}_Y$) and column ($\mathbf{F}_X$) filters have three parameters each[1]: the grid center coordinates g_X, g_Y, the standard deviation for each axis σ_X, σ_Y and the stride between grid points on each axis δ_X, δ_Y. These parameters are dynamically computed as an affine transformation of a vector of activations $\mathbf{h}$ from

[1]The original read mechanism in [13] also adds a scalar intensity parameter γ, that is multiplied to filter responses.

a neural network layer:

$$(\tilde{g}_X, \tilde{g}_Y, \tilde{\sigma_X}, \tilde{\sigma_Y}, \tilde{\delta_X}, \tilde{\delta}_Y) = \mathbf{W}\mathbf{h} + \mathbf{b}, \tag{3}$$

where $\mathbf{W}$ is the transformation matrix and $\mathbf{b}$ is the bias. This is followed by normalization of the parameters:

$$g_X = \frac{\tilde{g}_X + 1}{2}, \qquad g_Y = \frac{\tilde{g}_Y + 1}{2}, \tag{4}$$

$$\delta_X = \frac{A-1}{M-1} \cdot |\tilde{\delta_X}|, \qquad \delta_Y = \frac{B-1}{N-1} \cdot |\tilde{\delta_Y}|, \tag{5}$$

$$\sigma_X = |\tilde{\sigma_X}|, \qquad \sigma_Y = |\tilde{\sigma_Y}|. \tag{6}$$

The mean coordinates μ_X^i, μ_Y^j of the Gaussian filter at column i, row j in the attention grid are computed as follows:

$$\mu_X^i = g_X + (i - \frac{M}{2} - 0.5) \cdot \delta_X, \tag{7}$$

$$\mu_Y^j = g_Y + (j - \frac{N}{2} - 0.5) \cdot \delta_Y \tag{8}$$

Finally, the filter banks $\mathbf{F}_X$ and $\mathbf{F}_Y$ are defined by:

$$\mathbf{F}_X[i, a] = \exp\left(-\frac{(a - \mu_X^i)^2}{2\sigma_X^2}\right), \tag{9}$$

$$\mathbf{F}_Y[j, b] = \exp\left(-\frac{(b - \mu_Y^j)^2}{2\sigma_Y^2}\right) \tag{10}$$

The filters (rows of $\mathbf{F}_X$ and $\mathbf{F}_Y$) are later normalized to sum to one.

Our read mechanism makes the following modifications to the DRAW read mechanism [13]:

- We allow rectangular (not only square) attention grids and use separate strides and standard deviations for X and Y-axis. This allows the model to stretch and smooth the glimpse content to correct for distortions introduced by ignoring the original aspect ratio of an input image.
- We use $|x|$ instead of $\exp(x)$ to ensure positivity of strides and standard deviations (see Equations 5 and 6). The motivation for this modification is that in our experiments we observed stride and standard deviation parameters to often saturate at low values, causing the attention window to zoom in on a single pixel. This effectively inhibits gradient flow through neighboring pixels of the attention filters. Piecewise linear activation functions have been shown to benefit optimization [23] and the absolute value function is a convenient trade-off between the harsh zeroing of all negative inputs of the ReLU and the extreme saturation for highly negative inputs of the exponential function.
- We drop the additional scalar intensity parameter γ, because we did not observe it to influence the performance in our experiments.

4. A Modular Framework for Vision

The proposed modular framework for an attention-based approach to computer vision consists of three components: an attention module (controlling *where* to look), a feature-extraction module (providing a representation of *what* is seen) and an objective module (formalizing *why* the model is learning its attentive behavior). An example architecture for tracking using these modules is described in Section 5.

4.1. Feature-extraction module

The feature-extraction module computes a representation of a given input glimpse. This representation can be as simple as the identity transformation, i.e. raw pixels, or a more sophisticated feature extractor, e.g. an CNN. The extracted features are used by other modules to reason about the visual input. Given a hierarchy of features, such as the activations of layers in an CNN, different features can be passed to the attention and objective modules.

We found that it can be useful to pre-train the feature-extraction module on a large data set, before starting to train the full architecture. After pre-training, the feature extractor's parameters can either be continued to be updated during end-to-end training, or kept fixed. Figure 2 shows the symbol used in the following sections to represent a feature-extraction module.

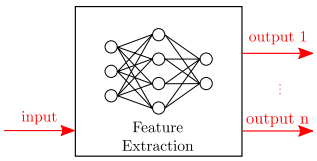

Figure 2: The symbol for the feature-extraction module. It can have multiple outputs (e.g. activations from different layers of an CNN).

4.2. Attention Module

The attention module is composed of an RNN (see Section 2) and a read mechanism (see Section 3). At each time step, a glimpse is extracted from the current input frame using the attention parameters the RNN predicted in the previous time step (see Section 3). Note that in this context, Equation 3 of the read mechanism corresponds to Equation 2 of the RNN. After the glimpse extraction, the RNN updates its hidden state using the feature representation of the glimpse as input (see Equation 1). Figure 3 shows the symbolic representation used in the following sections to represent the recurrent attention module.

4.3. Objective Module

An objective module guides the model to learn an attentional policy to solve a given task. It outputs a scalar

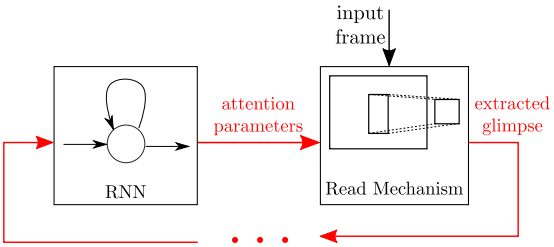

Figure 3: The symbolic representation of a recurrent attention module, which is composed of an RNN and a read mechanism that extracts a glimpse from the input frame. The extracted glimpse is fed back to the RNN. The dots indicate, that the feed-back connection can involve intermediate processing steps, such as feature extraction.

cost, that is computed as function of its target and prediction inputs. There can be multiple objective modules for a single task. A learning algorithm, such as Stochastic Gradient Descent (SGD), uses the sum of cost terms from all objective modules to adapt the parameters of the other modules. Objective modules can receive their input from different parts of the network. For example, if we want to define a penalty between window coordinates, the module would receive predicted attention parameters from the attention module and target parameters from the trainer.

In all our objective modules we use the Mean Squared Error (MSE) for training:

$$\mathcal{L}_{MSE} = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{y}_{target} - \mathbf{y}_{pred}||_2^2, \tag{11}$$

where n is the number of training samples, $\mathbf{y}_{pred}$ is the model's prediction, $\mathbf{y}_{target}$ is the target value and $||.||_2^2$ is the squared Euclidean norm. We use the MSE even for classification, as this makes the combination of multiple objectives simpler and worked well. Figure 4 shows the symbol we use to represent an objective module.

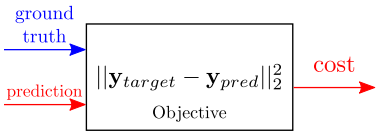

Figure 4: The symbol for the objective module.

5. Building a Recurrent Attentive Tracking Model

The task of tracking involves mapping a sequence of input images $\mathbf{x}_1, \ldots, \mathbf{x}_T$ to a sequence of object locations $\mathbf{y}_1, \ldots, \mathbf{y}_T$. For the prediction $\hat{\mathbf{y}}_t$ of an object's location at time t, the trajectory $(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1})$ usually contains relevant contextual information, and an RNN has the capacity to represent this trajectory in its hidden state.

5.1. Architecture

At each time step, the recurrent attention module outputs a glimpse from the current input frame using the attention parameters predicted at the previous time step. Optionally, a feature-extraction module extracts a representation of the glimpse and feeds it back to the attention module, which updates its hidden state. The tracking behavior can be learned in various ways:

- One can penalize the difference between the glimpse content and a ground truth image. For simple data sets, this is done on the raw pixel representation. This loss is defined as

$$\mathcal{L}_{pixel} = ||\hat{\mathbf{p}} - \mathbf{p}||_2^2, \tag{12}$$

  where $\hat{\mathbf{p}}$ is the glimpse extracted by the attention mechanism and $\mathbf{p}$ is the ground truth image. Objects with more variance in appearance, require a more robust distance measure, e.g. defined via a feature mapping $f(.)$ (implemented by a feature-extraction module):

$$\mathcal{L}_{feat} = ||f(\hat{\mathbf{p}}) - f(\mathbf{p})||_2^2, \tag{13}$$

- Alternatively, a penalty term can also be defined directly on the attention parameters. For instance, the distance between the center of the ground truth bounding box and the attention mechanism's $\hat{\mathbf{g}} = (g_x, g_y)$ parameters can be used as a localization loss

$$\mathcal{L}_{loc} = ||\hat{\mathbf{g}} - \mathbf{g}||_2^2, \tag{14}$$

We explore several variations of this architecture in Section 6.

5.2. Evaluation of Tracking Performance

Tracking models can be evaluated quantitatively on test data using the average Intersection-over-Union (IoU) [10]

$$IoU = \frac{|B_{gt} \cap B_{pred}|}{|B_{gt} \cup B_{pred}|}, \tag{15}$$

where B_{gt} and B_{pred} are the ground truth and predicted bounding boxes. A predicted bounding box for RATM 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 glimpse is a weighted sum of pixels around the grid points and the boxes are smaller than the region seen by the attention module. While this might affect the performance under the average IoU metric, the average IoU still serves as a reasonable metric for the soft attention mechanism's performance in tracking.

6. Experimental Results

For an initial study, we use generated data, as described in Sections 6.1 and 6.2, to explore design choices without limitations by the number of available training sequences. In Section 6.3, we show how one can apply the RATM in a real-world context.

6.1. Bouncing Balls

For our initial experiment, we generated videos of a bouncing ball using the script released with [34]. The videos have 32 frames of resolution 20×20. We used $100,000$ videos for training and $10,000$ for testing. The **attention module** has 64 hiddens in its RNN and its read mechanism extracts glimpses of size 5×5. The attention parameters are initialized to a random glimpse in the first frame. The input to the RNN are raw pixels of the glimpse, i.e. the **feature-extraction module** here is the identity. The **objective module** computes the MSE between the glimpse at the last time step and a target patch, which is simply a cropped ball image, since shape and color of the object are constant across the whole data set.

For learning, we use SGD with a mini-batch size of 16, a learning rate of 0.01 and gradient clipping [25] with a threshold of 1 for 200 epochs. RATM is able to learn the correct tracking behaviour only using the penalty on the last frame. We also trained a version with the objective module computing the average MSE between glimpses of all time steps and the target patch. The first two rows of Table 1 show the test performance of the model trained with only penalizing the last frame during training. The first row shows the average IoU of the last frame and the second shows the average IoU over all 32 frames of test sequences. The third row shows the average IoU over all frames of the model trained with the penalty on all frames.

The model trained with the penalty at every time step is able to track a bouncing ball for sequences that are much longer than the training sequences. We generated videos that are almost ten times longer (300 frames) and RATM reliably tracks the ball until the last frame.

The dynamics in this data-set are rather limited, but as a proof-of-concept they show that the model is able to learn tracking behavior end-to-end. We describe more challenging tasks in the following sections.

6.2. MNIST

To increase the difficulty of the tracking task, we move to more challenging data sets, containing more than a single type of object (ten digits), each with variation. We generate videos from 28×28 MNIST images of handwritten digits [20] by placing randomly-drawn digits in a larger 100×100 canvas with black background and moving the digits from one frame to the next. We respected the training and testing split of the original MNIST data-set for the generation of videos. Figure 5 shows the schematic of RATM for the

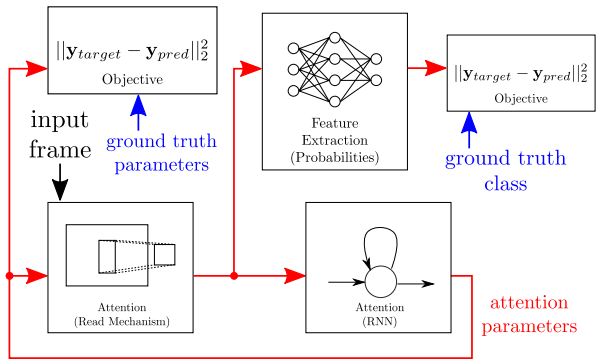

Figure 5: The architecture used for MNIST experiments.

MNIST experiments. The **attention module** is similar to the one used in Section 6.1, except that its RNN has 100 hidden units and the size of the glimpse is 28×28 (the size of the MNIST images and the CNN input layer).

In the bouncing balls experiment we were able to generate a reliable training signal using pixel-based similarity. However, the variation in the MNIST data set requires a representation that is robust against small variations to guide training. For this reason, our **feature-extraction module** consists of a (relatively shallow) CNN, that is pre-trained on classification of MNIST digits. Note, that the CNN is only used during training. The CNN structure has two convolutional layers with filter bank sizes of $32 \times 5 \times 5$, each followed by a 2×2 maxpooling layer, 0.25 dropout [14], and ReLU activation function. These layers are followed by a 10-unit softmax layer for classification. The CNN was trained using SGD with a mini-batch size of 128, a learning rate 0.01, momentum of 0.9 and gradient clipping with a threshold of 5.0 to reach a validation accuracy of 99%.

This CNN is used to extract class probabilities for each glimpse and its parameters remain fixed after pre-training. One **objective module** computes the loss using these probabilities and the target class. Since training did not converge to a useful solution using only this loss, we first introduced an additional objective module penalizing distances between the upper-left and lower-right bounding-box corners and the corresponding target coordinates. While this also led to unsatisfactory results, we found that replacing the bounding box objective module with one that penalizes only grid center coordinates worked well. One possible explanation is, that this does not constrain the stride and the zoom can be adjusted after locating the object. The two penalties on misclassification and on grid center distance, helped the model to reliably find and track the digit. 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

to maximize classification accuracy. For learning we use SGD with mini-batch size of 32, a learning rate of 0.001, momentum of 0.9 and gradient clipping with a threshold of 1 for $32,000$ gradient descent steps.

Single-Digit: In the first MNIST experiment, we generate videos, each with a single digit moving in a random walk with momentum. The data set consists of $100,000$ training sequences and $10,000$ test sequences. The initial glimpse roughly covers the whole frame. 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. The fourth row of Table 1 shows the average IoU over all frames of the test sequences.

Multi-Digit: It it interesting to investigate how robust RATM is in presence of another moving digit in the background. To this end, we generated new sequences by modifying the bouncing balls script released with [34]. The balls were replaced by randomly drawn MNIST digits. We also added a random walk with momentum to the motion vectors. We generated $100,000$ sequences for training and $5,000$ for testing. Here, the bias for attention parameters is not a learn-able parameter. For each video, the bias is set such that the initial glimpse is centered on the digit to be tracked. Width and height are set to about 80% of the frame size. The model was also trained on 10-frame sequences and was able to track digits for at least 15 frames on test data. Figure 6 shows tracking results on a test sequence. The fifth row of Table 1 shows the average IoU of all test sequences over 30 frames.

6.3. Tracking humans in video

To evaluate the performance on a real-world data set, we train RATM to track humans in the KTH action recognition data set [28], which has a reasonably large number of sequences[2]. We selected the three activity categories, which show considerable motion: walking, running and jogging. We used bounding boxes provided by [17], which were not hand-labeled and contain noise, such as bounding boxes around the shadow instead of the subject itself.

For the **feature-extraction module** in this experiment, we trained a CNN on binary – human vs. background – classification of 28×28 grayscale patches. The data consisted of $21,134$ positive patches cropped from the ETH pedestrian [9] and INRIA person [6] data sets and $29,923$ negative patches cropped from the KITTI detection benchmark [11]. $20,000$ samples of each class were used for training of the CNN. The architecture of the CNN is as follows: two convolutional layers with filter bank sizes $128 \times 5 \times 5$ and $64 \times 3 \times 3$, each followed by 2×2 max-pooling and ReLU activation. After the convolutional layers, we added one fully-connected ReLU-layer with 256 hiddens and the output softmax-layer of size 2. For **pre-training**, we used SGD with mini-batch size of 64, a learning rate of 0.01, momentum of 0.9 and gradient clipping with threshold 1. We performed early stopping with a held-out validation set sampled randomly from the combined data set.

As this real-world data set has more variation than the previous data sets, the **attention module**'s RNN can also benefit from a richer feature representation. Therefore, the ReLU activations of the second convolutional layer of the feature-extraction module are used as input to the attention module. The RNN has 32 hidden units. This low number of hidden units was selected to avoid overfitting, as the number of sequences ($1,200$ short sequences) in this data set is much lower than in the synthetic data sets. We initialize the attention parameters for the first time step with the first frame's target window. The initial and target bounding boxes are scaled up by a factor of 1.5 and the predicted bounding boxes are scaled back down with factor $\frac{1}{1.5}$ for testing. This was necessary, because the training data for the feature-extraction module had significantly larger bounding box annotations.

The inputs to the **objective module** are the ReLU activations of the fully-connected layer, extracted from the predicted window and from the target window. The computed cost is the MSE between the two feature vectors. We also tried using the cosine distance between two feature vectors, but did not observe any improvement in performance. The target window is extracted using the same read mechanism as in the attention module. Simply cropping the target bounding boxes would have yielded local image statistics that are too different from windows extracted using the read mechanism. Figure 7 shows the schematic of the architecture used in this experiment.

For learning, we used SGD with a mini-batch size of 16, a learning rate of 0.001 and gradient clipping with a threshold of 1.0. In this experiment we also added a weight-decay regularization term to the cost function that penalizes the sum of the squared Frobenius norms of the RNN weight matrices from the input to the hidden layer and from the hidden layer to the attention parameters. This regularization term improved the stability during learning. As another stabilization measure, we started training with short five-frame sequences and increased the length of sequences by one frame every 160 gradient descent steps.

For evaluation, we performed a leave-one-subject-out experiment. For each of the 25 subjects in KTH, we used the remaining 24 for training and validation. A validation subject was selected randomly and used for early stopping. The reported number in the sixth row of Table 1 is the IoU on full-length videos of the test subject averaged over frames of each left-out subject and then averaged over subjects.

Figure 8 shows an example of test sequences for the class

[2]Code for this experiment is available at `https://github.com/saebrahimi/RATM`

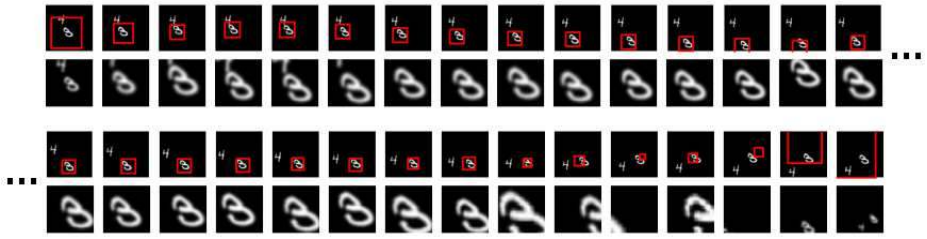

Figure 6: Tracking one of two digits. The first and second row show the sequence and corresponding extracted glimpses, respectively. The red rectangle indicates the location of the glimpse in the frame. The third and fourth row are the continuation. Prediction works well for sequences twice as long as the training sequences with 10 frames.

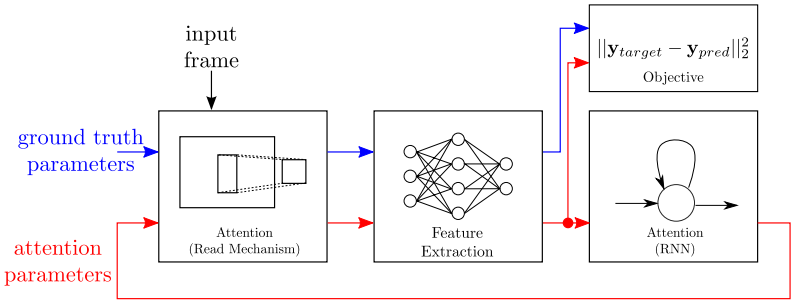

Figure 7: The architecture used for KTH experiments.

walking. Note, that the region captured by the glimpses is larger than the bounding boxes, because the model internally scales the width and height by factor 1.5 and the Gaussian sampling kernels of the attention mechanism extend beyond the bounding box. An interesting observation is that RATM scales up the noisy initial bounding box in Figure 8 (bottom example), which covers only a small part of the subject. This likely results from pre-training the feature-extraction module on full images of persons. Although the evaluation assumes accurate target bounding boxes, RATM is able to recover from such noise.

To show how the model generalizes to unseen videos containing humans, we let it predict sequences of the TB-100 tracking benchmark [39]. For this experiment, we picked one of the 25 KTH model, that had a reasonably stable learning curve (IoU over epochs). Figure 9 shows every tenth predicted frame of the sequences *Skater2* and *BlurBody*. For the first example, *Skater2*, RATM tracks the subject reliably through the whole length of the sequence. This is interesting, as the tracking model was trained only on sequences of up to 30 frames length and the variation in this data is quite different from KTH. The BlurBody sequence is more challenging, including extreme camera motion, causing the model to fail on parts of the sequence. Interestingly, in some cases it seems to recover.

In general, the model shows the tendency to grow the window, when it loses a subject. This might be explained by instability of RNN dynamics and blurry glimpses due to flat Gaussians in the attention mechanism.

7. Discussion

We propose a novel neural framework including a soft attention mechanism for vision, and demonstrate its application to several tracking tasks. Contrary to most existing similar approaches, RATM only processes a small window of each frame. The selection of this window is controlled by a learned attentive behavior. Our experiments explore several design decisions that help overcome challenges associated with adapting the model to new data sets. Several observation in the real-world scenario in Section 6.3, are important for applications of attention mechanisms in computer vision in general:

- The model can be trained on noisy bounding box annotation of videos and at test time recover from noisy initialization. This might be related to pre-training of the feature-extraction module. The information about the appearance of humans is transferred to the attention module, which learns to adapt the horizontal and vertical strides among other parameters of the glimpse to match this appearance.

- The trained human tracker seems to generalize to related but more challenging data.

The modular architecture is fully differentiable, allowing end-to-end training. End-to-end training allows the dis-

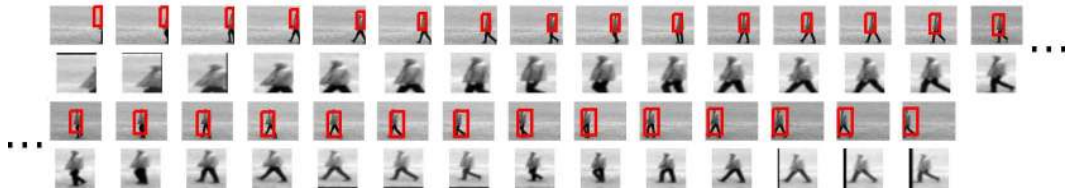

Figure 8: An example of tracking on the KTH data set. The layout is as follows: the first row shows 15 frames of one test sequence with a red rectangle indicating the location of the glimpse. The second row contains the extracted glimpses. The third and fourth row show the continuation of the sequence. We only show every second frame.

Figure 9: Predictions of a KTH model on sequences from the TB-100 benchmark. From top to bottom we show the sequences *Skater2* and *BlurBody*. To save space, we only show every tenth frame. The layout for each sequence is as follows: The first row shows 15 frames of one test sequence with a red rectangle indicating the location of the predicted glimpse. The second row contains the extracted glimpses. The third and fourth row show the continuation of the sequence.

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)
KTH (average leave-one-subject-out)	55.03 (full length of test sequences)

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

covery of spatio-temporal patterns, which would be hard to learn with separate training of feature extraction and attention modules. In future work we plan to selectively combine multiple data sets from different tasks, e.g. activity recognition, tracking and detection. This might allow to benefit from synergies between tasks [4], and can help overcome data set limitations. We also intend to explore alternatives for the chosen modules, e.g. using *spatial transformers* [16] as read mechanism, that can align glimpses using various types of transformations. Spatial transformers in an RNN applied to digit recognition have been explored in [31].

Acknowledgments

We thank the developers of Theano [37], Kishore Konda, Jörg Bornschein and Pierre-Luc St-Charles. This work was supported by an NSERC Discovery Award, CIFAR, FQRNT and the German BMBF, project 01GQ0841.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision*, pages 850–865. Springer, 2016.

[3] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

[4] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In C. Schmid, S. Soatto, and C. Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[7] M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas. Learning where to attend with deep architectures for image tracking. *CoRR*, abs/1109.3737, 2011.

[8] S. Ebrahimi Kahou, V. Michalski, K. Konda, R. Memisevic, and C. Pal. Recurrent neural networks for emotion recognition in video. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, ICMI '15, pages 467–474, New York, NY, USA, 2015. ACM.

[9] A. Ess, B. Leibe, K. Schindler, , and L. van Gool. A mobile vision system for robust multi-person tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. IEEE Press, June 2008.

[10] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.

[13] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623, 2015.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2008–2016, 2015.

[17] Z. Jiang, Z. Lin, and L. S. Davis. Recognizing human actions by learning and matching shape-motion prototype trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):533–547, 2012.

[18] H. Larochelle and G. E. Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Advances in neural information processing systems*, pages 1243–1251, 2010.

[19] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Z. Li, E. Gavves, M. Jain, and C. G. Snoek. Videolstm convolves, attends and flows for action recognition. *arXiv preprint arXiv:1607.01794*, 2016.

[22] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014.

[23] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[24] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. *CoRR*, abs/1510.07945, 2015.

[25] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

[26] M. Ranzato. On learning where to look. *arXiv preprint arXiv:1405.5488*, 2014.

[27] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[28] C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 32–36. IEEE, 2004.

[29] P. Sermanet, A. Frome, and E. Real. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054*, 2014.

[30] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(7):1442–1468, 2014.

[31] S. K. Sønderby, C. K. Sønderby, L. Maaløe, and O. Winther. Recurrent spatial transformer networks. *arXiv preprint arXiv:1509.05329*, 2015.

[32] S. K. Sønderby, C. K. Sønderby, H. Nielsen, and O. Winther. Convolutional lstm networks for subcellular localization of proteins. In *Algorithms for computational biology*, pages 68–80. Springer, 2015.

[33] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*, 2015.

[34] I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted boltzmann machine. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2009.

[35] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[36] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1420–1429, 2016.

[37] T. T. D. Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[38] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *Advances in neural information processing systems*, pages 809–817, 2013.

[39] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(9):1834–1848, 2015.

[40] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.