Image Classification with Recurrent Attention Models

Stanislau Semeniuta
Universität zu Lübeck
Institut für Neuro- und Bioinformatik
Email: stas@inb.uni-luebeck.de

Erhardt Barth
Universität zu Lübeck
Institut für Neuro- und Bioinformatik
Email: barth@inb.uni-luebeck.de

Abstract—**In this work we apply a fully differentiable Recurrent Model of Visual Attention to unconstrained real-world images. We propose a deep recurrent attention model and show that it can successfully learn to jointly localize and classify objects. We evaluate our model on multiple digit images generated from MNIST data, Google Street View images, and a fine-grained recognition dataset of 200 bird species, and show that its performance is either comparable or superior to that of alternative models.**

I. Introduction

Convolutional Neural Networks (CNNs) have recently achieved excellent results in various of visual recognition tasks [1, 2, 3]. However, CNNs suffer from poor scalability with respect to the input image size. Due to high computational costs, despite the fact that images are almost always rescaled to a smaller size, most of the current state-of-the-art CNNs are trained either on multiple high-end GPUs or in a highly distributed environment [4]. One way to overcome this issue is to allow a network to dynamically attend to most informative regions of an image. The dynamic attention approach has received a lot of attention in recent years [5, 6, 7, 8, 9, 10] and prior to CNNs achieving state-of-the-art results [11, 12].

This work continues the line of research directed towards learning a Recurrent Attention Model (RAM). A RAM processes an image sequentially, gradually collecting information present in the image. The model achieves this by extracting a *glimpse* from an image, extracting features from the *glimpse* and updating its internal state. The most simple and straightforward form of *glimpse* extraction is cropping an area from an image. However, this procedure is not differentiable and thus prevents end-to-end learning with backpropagation. There are two major ways to overcome this issue. One is to use a different learning approach [9, 6, 10] and the other is to define a differentiable *glimpse* extraction procedure [13, 8]. Our work follows the second approach.

Our contribution is twofold. Firstly, we describe a deep differentiable Recurrent Attention Model, show that it can consistently learn to jointly locate and classify objects in an image, and demonstrate that it achieves performance superior to that of reinforcement learning based RAM. Secondly, we describe ways to increase the speed of learning, improve generalization, and prevent the learning of bad attention policies. We use these techniques to establish a new state-of-the-art on the task of transcribing tightly cropped unsegmented house numbers.

II. Related Work

The computational efficiency of recognition algorithms has always received a lot of attention. It is especially important for object detection algorithms, since they usually need to perform multiple inferences per image. The amount of computation is mostly affected by the number of regions to be examined. This number is the largest for the simple sliding window approach and researchers have made a number of attempts to find alternatives to the sliding window approach such as a model that internally performs object detection step.

The most straightforward way to integrate object detection is to enable a model to extract crops from an image and process them subsequently. However, this approach poses a problem to backpropagation based learning, since the cropping is not differentiable. A number of attempts to overcome this issue have been made. Mnih et al. [9] have shown that it is possible to use reinforcement learning to successfully train an attention model. Ba et al. [5] and Sermanet et al. [14] extended this work and showed that RAM trained with reinforcement learning can be effective in challenging real world applications. Their work differs from ours in the training method: the authors have used a Reinforcement Learning based algorithm, while our model can be trained end-to-end with backpropagation. Ranzato [10] have used a two phase algorithm that optimizes location estimation and target predictions independently. Gregor et al. [8] have proposed a generative model that creates an image in an iterative fashion. Among other contributions, the authors have presented a differentiable crop operation and thus completely removed the need of dedicated attention learning algorithms. They have demonstrated that their model outperforms the one by Mnih et al. [9] on a synthetic classification task. Our work extends the model and investigates the classification performance it can achieve. Jaderberg et al. [13] have proposed a trainable module called Spatial Transformer, that is very similar in nature to a recurrent attention model. The authors use an affine transformation matrix to parametrize grid sampling points and a differentiable interpolation kernel to produce a crop from an image. Sønderby et al. [15] combine the Spatial Transformer modules with a Recurrent Neural Network. This work is the closest to ours in terms of the methodology, however the direct comparison is difficult due to a limited amount of experimental results in Sønderby et al. [15].

III. Recurrent Attention Model

A Recurrent Attention Model is a model that processes information in a sequential manner. Its processing consists of three steps: choosing where to look, extracting features from the chosen location, and updating the RAM's internal state. These steps form one iteration, which is then repeated for as many times as required by the task at hand. This process is very different from one-shot approach of CNNs and has a number of potential advantages. It allows the model to ignore any kind of unnecessary information present in an image, collect data scattered over an image and naturally model object detection.

Figure 1 provides a graphical description of our model, which mostly follows the iterative scheme described before, with two notable differences. Firstly, we extract features from the whole frame before iterations start. These features encode a rough representation of an image and are used by the model to guide its attention. Secondly, we explicitly split the hidden state into two parts, one responsible for modeling attention and the other for aggregating features over multiple iterations. As shown in Figure 1, our model can be split into a number of relatively independent modules. Most of these modules are neural networks, each serving its distinct purpose. Now we will introduce these modules and comment on their structure and tasks they perform.

Context network. The context network receives the whole image as input and outputs a feature vector. The job of the context network is to provide clues on where to deploy the model's attention. We usually parametrize this network by a CNN with two or three layers. In addition, it is possible to downsample an image before processing it with the context network to save computations.

Recurrent network. The core of our model is a two-layer recurrent neural network that aggregates information over multiple glimpses and controls the locations of the glimpses. We initialize the states of both layers to zeros. In general, any kind of recurrent function can be used to implement this module. We use a Gated Recurrent Unit (GRU) [16] network for its good trade-off between stable learning and computational costs. There is a clear separation of tasks between the two recurrent layers: the first aggregates data over multiple iterations and is used as an input to the classification network, while the second one only affects attention. In contrast to the widely adopted bottom-up computation order, we compute activations of the second layer before activations of the first layer. The main reason for this top-down order of computation is that the context network is only connected to the second layer. Since we compute activations of the second layer prior to the first one, the context network features are able to affect the first glimpse. When the allowed number of iterations is small, i.e. 2 or 3, having the location of the first glimpse fixed can use a large portion of the allowed computations. We have experimented with a variant of our model with a single recurrent layer, but encountered severe with issues exploding gradients.

Glimpse extraction network. The extraction network maps the hidden state of the second recurrent layer to attention parameters and uses them to yield the current glimpse. We use a linear mapping ($\mathbf{y} = W\mathbf{x} + b$) from the hidden state of the second recurrent layer to the attention parameters. The module then extracts a NxN glimpse following the procedure described by Gregor et al. [8]. The authors use a NxN grid of Gaussian filters placed upon an image to create a glimpse. The grid is parametrized by the location of its center c_x and c_y, the distance between adjacent points δ, the variance of Gaussian filters σ and the sharpening coefficient γ. We make two alterations to the grid parametrization. First, we use rectangular attention and replace the δ parameter with a pair δ_x and δ_y. Second, we make sure that the predicted center of the sampling grid is always within the image by restricting c_x and c_y to be between -1 and 1.

Glimpse network. The glimpse network is a function that receives the current extracted glimpse and outputs a feature vector. Its responsibility is to extract a set of useful features that are later used by the recurrent network. In most of our experiments this function is implemented with a CNN.

Classification network. The classification network produces the prediction given a current state of the first recurrent layer. In general it outputs a prediction on every step. However the exact moments when to use supervision, i.e. only on last step, can vary. We have observed very little influence of this module on the final result, so we have used small networks, such as logistic regression or a fully connected network with one hidden layer.

Full model. Lastly, we put all the discussed modules together. Let $f_{context}$, f_{ext}, $f_{glimpse}$, f_{class} and f_{rec} denote context, extraction, glimpse, classification and recurrent networks respectively. Given input image I, our model can be expressed with the following equations:

$$\mathbf{h}_t^2 = f_{rec}(\left[\mathbf{h}_{t-1}^1, f_{context}(I)\right], \mathbf{h}_{t-1}^2) \tag{1}$$

$$g = f_{ext}(I, \mathbf{h}_t^2) \tag{2}$$

$$\mathbf{h}_t^1 = f_{rec}(f_{glimpse}(g), \mathbf{h}_{t-1}^1) \tag{3}$$

$$\mathbf{y}_t = f_{class}(\mathbf{h}_t^1) \tag{4}$$

where $\mathbf{h}_t^1$, $\mathbf{h}_t^2$ and $\mathbf{y}_t$ are the hidden state of first and second layers and the model's output on step t respectively. Note that the second recurrent layer uses the hidden state of the first layer from previous step as input. Since each module of our model is differentiable, the whole model can be trained end-to-end with backpropagation. The most important module in this regard is the glimpse extraction network.

IV. Experiments

First, our model is evaluated on synthetic tasks, derived from the MNIST dataset. We show that it achieves either superior or competitive results compared to a reinforcement learning based alternative [5]. We then train it to read house numbers from the publicly available Street View House Numbers (SVHN) dataset [17] and show that our model achieves state-of-the-art

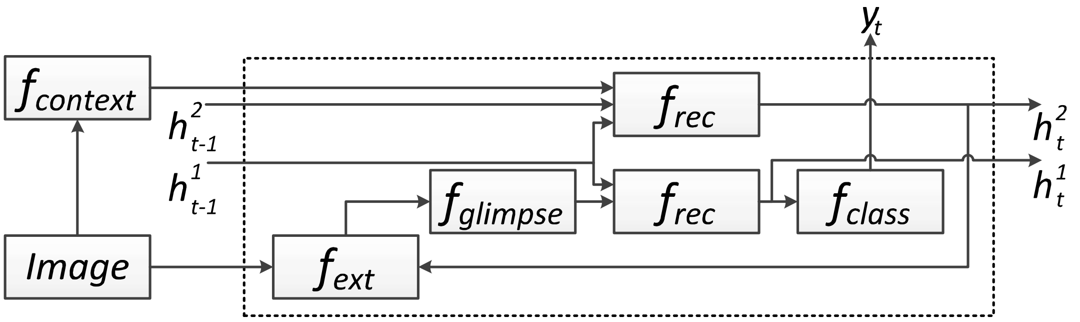

Fig. 1: Illustration of one processing step of the proposed model. $f_{context}$, f_{ext}, $f_{glimpse}$, f_{class} and f_{rec} denote context, extraction, glimpse, classification and recurrent networks respectively. See text for details on their structure and tasks. The blocks inside the dashed rectangle are iterated.

Model	Error
Mnih et al. [9], 4 glimpses	9.41%
Mnih et al. [9], 8 glimpses	8.11%
Gregor et al. [8], 4 glimpses	4.18%
Gregor et al. [8], 8 glimpses	3.36%
Ours, without context	3.29%
Ours, with context	1.80%

TABLE I: Test classification errors on the single digit recognition task

performance on the task of reading full house numbers from an image. Lastly, we demonstrate that the attention mechanism is capable of learning a decent object localizer when trained on the Birds-200-2011 dataset [18].

We have used Theano [19] and Caffe [20] frameworks to implement our experiments. Unless otherwise noted, the Adam [21] algorithm with an initial learning rate of 0.0001 was used to determine the step size of the gradient descent. β_1, β_2 and ϵ hyperparameters were set to 0.9, 0.999, and 1e-8 respectively.

A. Cluttered MNIST

Data. The dataset is derived from the popular MNIST benchmark by randomly placing one or two randomly selected digits in a 100x100 image. In addition, to simulate clutter present in natural images, 8x8 fragments of other digits are randomly placed in the same image as well. We define three subtasks based on this dataset: classifying an image with one digit into 10 classes, classifying an image with two digits into 55 classes, each corresponding to a particular pair of digits and classifying an image with two digits into 19 classes based on the sum of the digits. In the last task we follow [5] and use an empty background. Note that since digits are sampled randomly, the distribution of classes is not uniform.

Setup. We have chosen the glimpse and classification networks to be fully connected networks with one hidden layer with 256 units. In this case the model's performance directly depends on its ability to discover a good attention policy. In addition, it allows for a fair comparison with the work of [5].

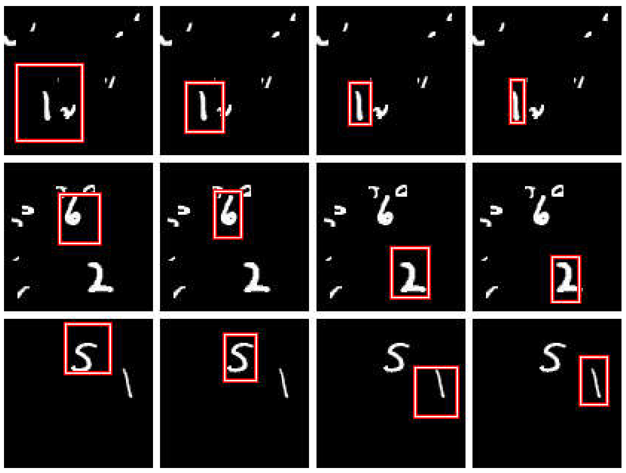

Fig. 2: Examples of learned attention on digit recognition (first row), pair recognition (second row) and digit summation (third row). Best viewed in color.

Model	Pair	Sum
Ba et al. [5]	5%	2.5%
Ours	4.3%	2.55%

TABLE II: Test classification errors on the pair recognition task

The number of glimpses was set to 4 in all experiments in this section. We have used a three layer CNN as context network.

We have found that it is difficult for our model to learn on weakly supervised images with two digits. There are two main problems that prevent it from learning a good attention policy. Firstly, the model tends to choose attention regions that include both digits instead of focusing on one and then switching on the other. This issue can be solved by imposing an upper limit on the size of the attention regions. However, it does not solve the problem completely, as in this case the network focuses on only one digit and never switches to the other. We have solved this issue by reparametrizing the reader's sampling grid and adding a regularization term to the cost function.

The extraction network, discussed in Section III, originally regresses absolute values of locations of grid centers. We have modified it to regress relative shifts $\triangle c_x$ and $\triangle c_y$ that are added to the previous center location. We then use Gaussian RBF depending on relative shifts as regularization term:

$$R = ae^{-(\triangle c_x^2 + \triangle c_y^2)/b} \tag{5}$$

where a and b are hyperparameters that we set to 0.1 in our experiments. This term promotes the model to choose locations that are far from the current one. We use this term only on the third processing step to force the model that after two steps it should deploy its attention elsewhere.

Results. The results of our model on single digit recognition task can be found in Table I. Our model achieves almost two times smaller error than the model of [8]. It should be noted that the authors did not strive to optimize the performance of their model on the digit recognition task. To highlight the importance of the context network, we note that errors in this task are caused by either incorrect digit localization or errors of classification network. When the context network is used the percentage of errors caused by mislocalization is less than 1% in contrast to about 10% without the context network.

Results of our model on pair classification and digit summation are given in Table II. Our RAM outperforms the model of [5] on the pair classification task. This is not the case for digit summation and our model performs slightly worse on this task. We find it somewhat surprising, since the images are very similar for these two tasks. In fact, it is possible to use a model trained on pair recognition to initialize weights of all components of a model for digit summation except for the last classification layer, and visa versa. Such model converges orders of magnitude faster than a randomly initialized one. Examples of learned attention mechanisms are presented in Figure 2.

B. Street View House Numbers

Data. In this section we apply our model to the medium sized dataset of house number images [17]. We follow Goodfellow et al. [22] and form a validation set by randomly choosing 5000 images from train and extra partitions. We also follow the same work in terms of data preprocessing and generate 64x64 tightly cropped images that contain all digits of a single house number. To do that, we use the ground truth bounding boxes, provided with the dataset, and compute a bounding box that includes all digits present in an image. We then increase the bounding box by 30% of its original size, crop the resulting area and rescale it to 64x64. We also follow Ba et al. [5] and generate 128x128 loosely cropped images by expanding the bounding boxes that encloses all digits in an image by 130%. In both cases we then convert the images to grayscale, divide by 128 and subtract the mean image.

Setup. This task is different from the ones discussed in Section IV-A since images contain variable numbers of objects. To address this, we train our model to read the image and output digits one by one. The model is trained to read an image from left to right. We add an extra class to encode that no more objects are present in an image and run our model until we receive the terminal label. The model is allowed to perform 3 glimpses per object. Since the dataset contains at most 5 objects in one image, the maximum possible amount of iterations is 18.

The SVHN dataset is more diverse than the one used in the previous section, so it is no longer sufficient to use a simple fully connected glimpse network. Following Ba et al. [5], we use a three layer CNN. However, we have observed that it leads to relatively slow convergence. We address this issue by training our model in two steps: first, we use a very simple glimpse network and train the model for a small number of epochs. After that we switch the simple glimpse network with the one that we actually would like to use. We have observed that it significantly speeds up training of the full model. Figure 4 shows learning curves of models trained from scratch and after 2 epochs of pretraining. Computational costs of pretraining are negligible with respect to the amount of time required to train the full model, so we use this approach in all experiments presented in this section. Note that such curriculum approach does not affect the final result, as the full model trained from scratch achieves similar results but requires more iterations to do so.

We regularize our model with 0.5 dropout in the outputs of the glimpse network and 0.5 dropout in recurrent connections [23]. Data augmentation does not provide satisfactory improvements of the final result, so we do not use it and always feed the network with the original 64x64 images. This can be explained by the presence of the attention mechanism that focuses on the same part of an image and effectively counteracts data augmentation. To address this, we add noise to the sampling grid parameters. We uniformly sample noise coefficient n from [-α, α] and add it to the location of grid center ($c_x = c_x + n$, $c_y = c_y + n$). This corresponds to small random shifts of attention regions. In addition, we multiply the distance between sampling points by $1+n$ ($\delta_x = \delta_x * (1+n)$, $\delta_y = \delta_y * (1+n)$), that results in either increase or decrease of corresponding dimensions. We have used $\alpha = 0.1$ in all our experiments and sample noise independently for every parameter. This modification can be viewed as the data augmentation applied to attention regions instead of the full image. There are two options of how to use noise during testing. One is to not use corrupted attention parameters. The other is to allow noise injection, run the model multiple times and average the results over all runs. The second approach is similar to MCMC averaging used by Ba et al. [5] and widely adopted oversampling [3]. It allows to achieve lower error rate at the cost of higher computational expenses when compared to the first option.

Results. The results of our model on transcribing house numbers are given in Table III. Similarly to Ba et al. [5], we have observed that our model tends to overestimate the number of digits in an image. To address this, we train an additional model that reads an image from right to left. We show an example of a single run of both models in Figure

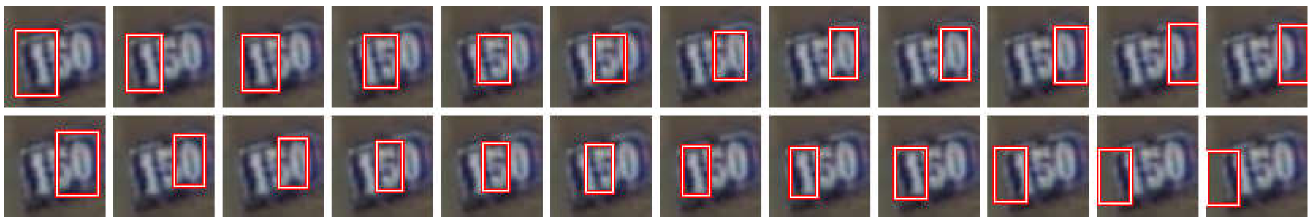

Fig. 3: Example of an image processed by forward (top row) and backward (bottom row) models. Computation proceeds from left to right, i.e. the first step corresponds to leftmost image in each row. Image was taken from the test partition. Best viewed in color.

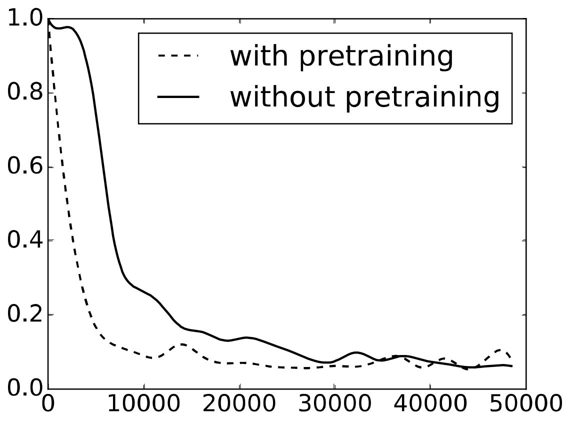

Fig. 4: Validation errors of full models with and without pretraining for first 25 epochs. We omit training errors since they follow the same pattern

TABLE III: Test classification errors on full sequence house number recognition task

Model	64x64	128x128
Goodfellow et al. [22]	3.96%	–
Ba et al. [5]	3.9%	4.46%
Jaderberg et al. [13]	3.6%	3.9%
forward, single pass	4.5%	5.5%
forward, average	4.3%	5.2%
both, single pass	3.9%	4.45%
both, average	3.65%	4.35%
5 layers, both, average	3.4%	4.34%

3. During testing we choose the smaller sequence length from two models and average predictions from them. This heuristic allows us to match the performance of the model with MCMC averaging from Ba et al. [5] with our model without averaging over multiple runs with sampling grid noise. When we allow injection of noise during testing and average over 10 runs, the performance of our model improves and almost achieves the performance of a Spatial Transformer based model [13]. All of the discussed models outperform a conventional CNN with eleven layers [22] while requiring much less computation. Lastly, we have found that our model with the three layer glimpse network slightly underfits training data due to regularization. This allowed us to add two more convolutional layers. The resulting five layer model yields a result better than that of Jaderberg et al. [13] and achieves state-of-the-art performance. However, it should be noted that the result of [13] is achieved with a single pass of a single model, while we use model averaging. In the case of loosely cropped house numbers our model achieves a better result than that of [5] by a similar margin. Interestingly, in this case the model with a 5 layer glimpse network achieves only marginally better result. However, the Spatial Transformer based model of Jaderberg et al. [13] is still superior to both RAMs on loosely cropped images.

TABLE IV: Test classification accuracies on the bird recognition task

Model	Accuracy
Jaderberg et al. [13]	84.1%
CNN, w/o localization	78.57%
CNN, GT localization	79.63%
CNN, RAM localization	79.77%

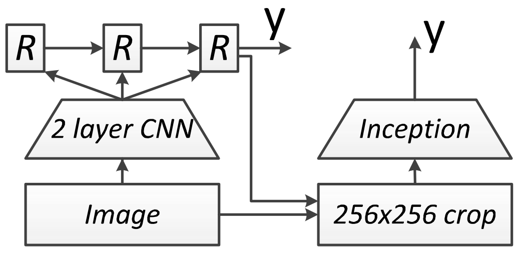

Fig. 5: Illustration of our approach to fine grained classification. *R* blocks denote one iteration of the model.

C. *Fine-Grained Bird Recognition*

Data. In this section we apply our model to the CUB-200-2011 dataset [18]. The dataset contains 12k images of various bird species split into train and test partitions of approximately the same size. We preprocess images by resizing to 256x256 and subtracting per-channel mean values.

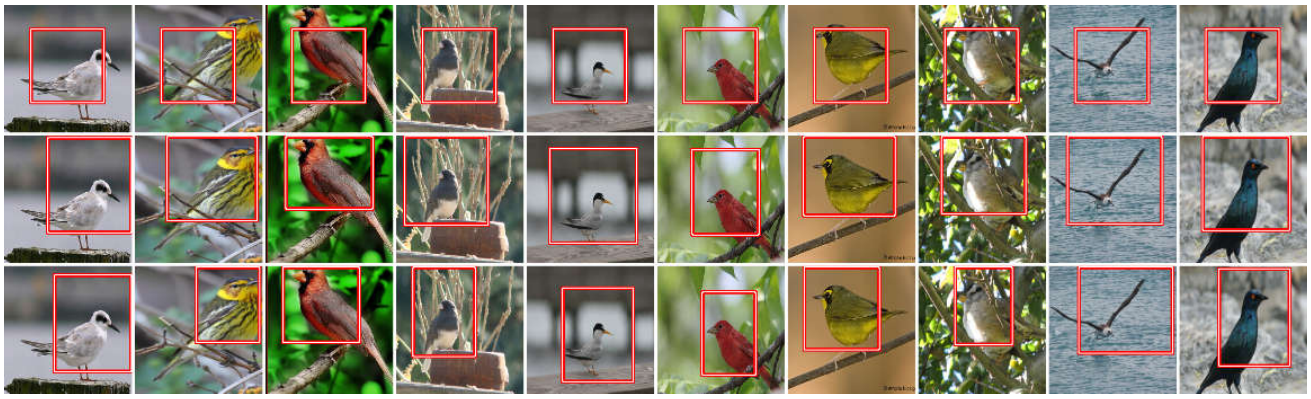

Fig. 6: Examples of learned attention policy. Computation proceeds from top to bottom, i.e. the first step corresponds to topmost image in each column. Since in this experiment our model acts on a convolutional image, shown bounding boxes are expanded to approximately cover the corresponding area of the raw image. Note that the attention region on the first step is always the same, since we do not use the context network. Images were taken from the test partition. Best viewed in color.

Setup. We consider an Inception architecture with Batch Normalization [24] as our baseline model. We train it on the ImageNet1000 dataset and finetune it on the CUB-200-2011 dataset. We use the standard data augmentation during training and randomly crop 224x224 areas of an image and mirror them with a 0.5 probability. We do not use multiscale augmentation. During testing we average outputs of the model applied to ten crops as described by [3]. The baseline Inception network achieves 78.57% accuracy.

In this case, the object of interest is relatively easy to locate and usually occupies a large portion of an image. Hence, we do not use the context network for these experiments. In contrast to our previous experiments, in this case we apply the attention mechanism to convolutional images. In particular, we first process an image with two sets of convolution-relu-pooling operations and only then use attention. Parameters of convolutional layers are pretrained on the ImageNet1000 dataset as part of a larger CNN with five convolutional layers and kept fixed throughout all experiments on this dataset. Our initial experiments have shown a substantial amount of overfitting. Therefore, we have changed the input of the classification network to be the output of the glimpse network instead of the hidden state of the first recurrent layer. In addition, we have hypothesized that the main source of overfitting is the classification network and that the full model does not have enough time to learn a good attention policy before the classification network overfits to the data. To address this issue, we have reset the weights of the last 200-way classification layer after every 10 epochs. We reset the weights of the classification network 10 times and then allow the model to converge. Indeed, this simple procedure reduces the validation error. However, while these modifications substantially reduce overfitting when compared to the original model, the baseline CNN still achieves a superior result. Thus, we use our model for an object localization step before baseline CNN.

To do that, we compute the attention region selected by the network on the last processing step and expand it to approximately cover the corresponding area of the raw image, extract this area from the image and resize it to 256x256 size. We process the whole dataset in this manner and train our baseline CNN on extracted images following exactly the same procedure as described before. This approach is schematically depicted in Figure 5. Note that the purpose of this experiment is to demonstrate that our model is capable of learning attention mechanisms on natural images. We do not achieve the current state-of-the-art results [13] in these experiments.

Results. Results of our experiments are presented in Table IV. The CNN that uses our model for localization achieves 1.2% improvement with respect to the baseline model trained on whole images. Examples of localizations are presented in Figure 6. To put performance of localization into context, we use the provided ground truth (GT) bounding boxes as a "perfect" localizer. Interestingly, CNN with RAM localization slightly outperforms the one with GT localization. The difference between these two is rather small, however.

V. Discussion

In our experiments the Recurrent Attention Model consistently learns good attention policies, although in some cases it requires some extra built-in knowledge. Reinforcement learning based RAM is superior to ours in this regard, since it can learn to switch between two digits based on the classification error only. This issue deserves further attention, since objects in natural scenes are very likely to appear at different locations of an image.

Ba et al. [5] have observed that their model does not benefit much from dropout regularization. In contrast, we have found that our model does not achieve the results of alternative models when we do not use regularization. We speculate that this is due to a certain amount of stochasticity built in into the model of Ba et al. [5], while ours is fully deterministic.

Our model has a number of further applications. While it shows reasonable results on image classification, CNN based models are already very well suited for this task and we do not expect RAM to significantly outperform them in terms of classification accuracy. RAM can rather yield an improvement for tasks that have some dependencies between predicted labels, such as optical character recognition or object detection. In addition, our model is easily extendable to temporal data. Some of our early experiments suggest that RAM trained on static images can learn to follow an object in a video with its attention region.

VI. Conclusions

We have presented a deep recurrent model based on recent advances in fully differentiable RAMs and have experimentally shown that it achieves competitive results on the MNIST based synthetic benchmark and state-of-the-art result on transcribing house numbers from an image. We have shown that it can operate on outputs of convolutional layers and learn object localizers from image labels only. We have demonstrated that learned attention can be used subsequently in a recognition pipeline to boost the final result. Lastly, we have shown a set of procedures that decrease the testing error of our model and improve convergence speed. The two most important techniques are pretraining with a simple glimpse network and injection of noise into attention parameters.

Acknowledgments

This project has received funding from the European Union's Framework Programme for Research and Innovation HORIZON 2020 (2014-2020) under the Marie Skłodowska-Curie Agreement No. 641805. Stanislau Semeniuta thanks the support from Pattern Recognition Company GmbH. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

References

[1] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

[2] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[5] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," *CoRR*, vol. abs/1412.7755, 2014.

[6] J. Ba, R. R. Salakhutdinov, R. B. Grosse, and B. J. Frey, "Learning wake-sleep recurrent attention models," in *NIPS*, 2015.

[7] S. M. A. Eslami, N. Heess, T. Weber, Y. Tassa, K. Kavukcuoglu, and G. E. Hinton, "Attend, infer, repeat: Fast scene understanding with generative models," *CoRR*, vol. abs/1603.08575, 2016.

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

[9] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," *CoRR*, vol. abs/1406.6247, 2014.

[10] M. Ranzato, "On learning where to look," *CoRR*, vol. abs/1405.5488, 2014.

[11] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE PAMI*, 1998.

[12] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order Boltzmann machine," in *NIPS*, 2010.

[13] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial transformer networks," in *NIPS*, 2015.

[14] P. Sermanet, A. Frome, and E. Real, "Attention for fine-grained categorization," *CoRR*, vol. abs/1412.7054, 2014.

[15] S. K. Sønderby, C. K. Sønderby, L. Maaløe, and O. Winther, "Recurrent spatial transformer networks," *CoRR*, vol. abs/1509.05329, 2015.

[16] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014.

[17] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[18] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Tech. Rep., 2011.

[19] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[22] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. D. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *CoRR*, vol. abs/1312.6082, 2013.

[23] S. Semeniuta, A. Severyn, and E. Barth, "Recurrent dropout without memory loss," *CoRR*, vol. abs/1603.05118, 2016.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.