Training Recurrent Attention Models on Handwritten Word Images

Name: Yadnyesh Patil, Roll No. : 193050067
Name: Rohit Saluja, Roll No. : 144054003
Name: Gaurav Baluni, Roll No : 19V980006
Name: Shivam Dixit, Roll No : 193050012

Abstract

In past Recurrent Attention Models have been used for reading MNIST digit images. The partial information is made available by applying different bandwidth limited glimpse sensors (for multiple resolutions focussed at the same location) on the image. In this approach we modify the RAM model by introducing fisheye based approach and decaying reward policy. We have applied our new improved model to EMNIST dataset and results are shown to improve performances gradually. Finally we have applied our model to IAM handwriting dataset and tried to mitigate certain challenges, as discussed, that we have faced.

Introduction

Handwritten Character recognition deals with interpreting intelligible handwritten input from sources like scanned paper documents, photographs, PDAs and other similar kinds of devices. There are generally two methods of sensing; the one being "offline sensing" where the previously written text is inputted to the machine for interpretation and other being "online sensing" where the strokes of pen are recorded for interpretation. Offline HTR becomes necessary for interpreting historical documents or in digitisation of hand-filled forms. Offline approach is quite difficult task as different people have different writing styles.

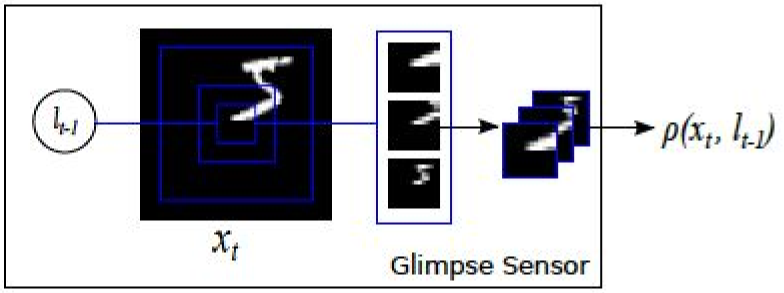
Generally convolutional based networks CNNs are employed to train on handwritten datasets. CNN based approaches have higher computational demands which scales linearly with the number of image pixels. Recently RAM (Recurrent Attention Model) have been employed for handwritten digit recognition which employs bandwidth limited glimpse sensors and makes only partial information available to the network[1]. In RAM model based approach the number of parameters and the computation it performs can be controlled independently of the size of the input image thus outperfroming CNNs. In this paper we first discuss original RAM model, and then our modified version of RAM model. The paper then discusses the experiments performed on EMNIST and IAM datasets.

Recurrent Attention Model

Recurrent Attention Model (RAM) the idea is based on how a human eye focuses on a particular region of interest in a widely captured view. The attention model uses this idea to let the network focus on only the informative part of image as the network learns while paying less attention elsewhere. This approach has two advantages: first the model pays more attention to the relevant part of image and secondly the image processing burden lessens as relatively shorter glimpse sizes are fed into the network.

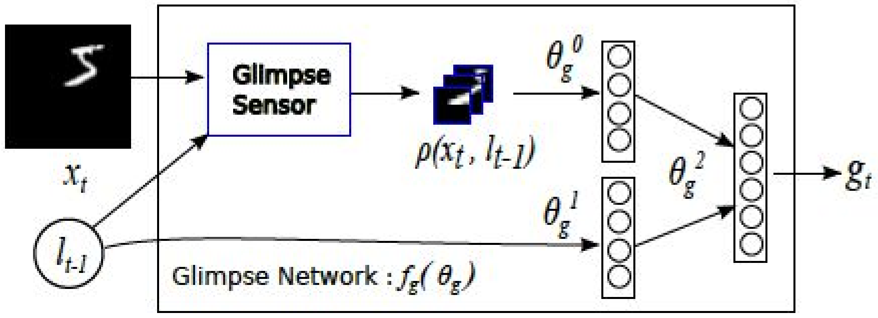
The Network Architecture

Glimpse Sensor



Glimpse Sensor works like our Retina of eye. Network "takes a look" at the image around a given location, called a glimpse, then extracts and resize it into various scales of image using the same resolution. Thus making, the smallest scale of crop in the centre the most detailed, whereas the largest crop in the outer ring is most blurred. The Glimpse network combines the glimpses and previous location tot output g_t.

Glimpse Network



Glimpse Network uses glimpse sensor, flattens it, and then combine the extracted retina representation with the glimpse location using hidden layers and ReLU, emitting a single vector

g. This vector contains the information of both "what" (glimpse) and "where" (the focused location within the image).

Recurrent Network

Recurrent Network takes feature vector input from Glimpse Network, and keeps the useful information via it's hidden states.

Location Network

Location Network inputs the hidden states from Recurrent Network, and outputs the next location to look/focus at. The next location then becomes input to the Glimpse Network in the next time step in the recurrent network. The Location Network uses policy gradient approach for predicting the next location of glimpse. The RAM model introduces a stochastic policy approach (i.e. gaussian distribution) to generate next location, and use reinforcement learning techniques to learn. It is also referred to as "hard" attention, since this stochastic process is non-differentiable (compared to "soft" attention).

REINFORCE with baseline policy gradient algorithm is used to solve this problem which relies on an estimated return by Monte-Carlo methods using episode samples to update the policy parameter θ. REINFORCE works because the expectation of the sample gradient is equal to the actual gradient:

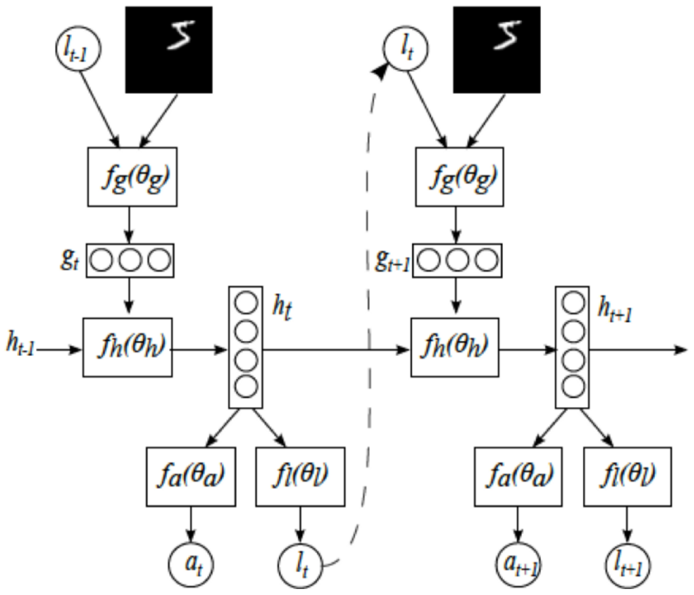
$$\nabla_{\theta} E_x [f(x)] = E_x [f(x) \nabla_{\theta} log p(x)]$$

The RAM model uses baseline 'b' to be the expected reward E[R]. The mean square error between R and b is minimized by backpropagation.

Classification Network

Classification Network takes hidden states from Recurrent Network as input, and predicts the digit. In this process for correct classification, reward point is generated, which is then used to train the Location Network.

The combined network architecture looks like this:



Fisheye Approach

The present work on RAM [2] exploits multiple glimpses each consisting of 'k' different patches (of different resolutions) around the common center (shown in different colors in left of Figure). The larger patches are then resized to the size of smallest to save compute (Refer top right of Figure). This includes redundancy because the central part of the glimpse is present in all the patches with different resolutions. In our implementation we have made it to a single variable resolution patch where resolution decreases as we move away from the center of the glimpse. The overall result would be a fisheye (like) glimpse for multiple-resolutions as shown in Figure .

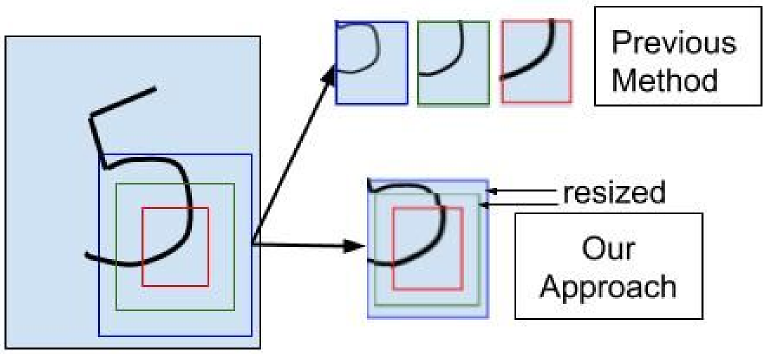


Figure 5: Our Approach: Single fisheye (like) glimpse (bottom right) for multiple-resolutions in original image (left). Method used in RAM [2] (top right).

Experiment 1: Experiments on EMNIST Dataset

EMNIST Dataset:

EMNIST-Letters consists of 26 classes (capital and small letters) containing 145600 samples. We trained on 112320 samples and tested on 12480. Sample images can be observed in Fig 1.1a. The label for a capital letter and its small version is the same, thus making the recognition task more complex (as compared to MNIST or any character recognition task) as there are many (2) to one mapping. Each character image is 1 channel grayscale coded with size 28X28.

Experiment 1.1: Exploring the Number of Glimpses on EMNIST

First we used different number of glimpses and ran experiments on EMNIST dataset for 100 epochs to find out the best number of glimpses. We use default patch_size of 8X8 and avoid multiple patches because the number of pixels in each image is 784 (28X28, refer section EMNIST Dataset), and 6 glimpses (best for MNIST) would had mean 768 (6X8X8X2) pixels read through two patches.

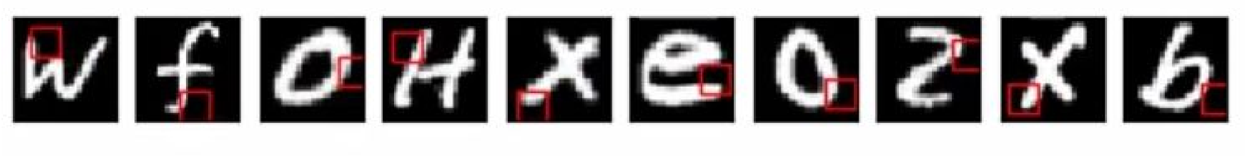
Baseline: These results can be compared with CNN Based approach (87.05 % accuracy on test set) https://www.simonwenkel.com/2019/07/16/exploring-EMNIST.html

For validation and testing, we pass each image 10 times through our model(s) and find the mean output to extract predicted characters. Following are the results with different number of glimpses over each EMNIST image.

Number of Glimpses	Test accuracy	Best Epoch No.
8	85.80	60
9	85.30	19
10	83.24	92

We ran our experiments for 100 epochs each. As observed, the increased number of glimpses result in lower accuracy. One of the probable reasons for this is that the extra glimpses wander to creates confusion (as seen in visualization of 9 glimpses below for l and m).

Screen_shot from 8 glimpse visualization:-



Screen_shot from 9 glimpse visualization:-

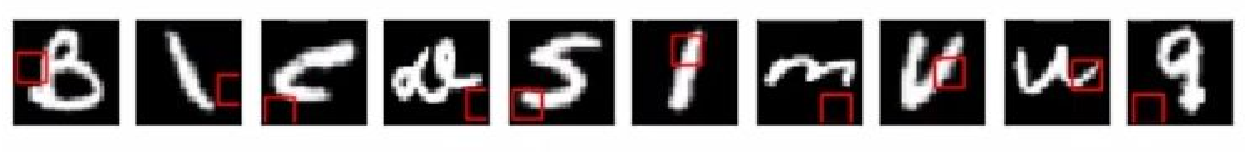


Fig 1.1a Comparison of 8 glimpses with 9 glimpses

Experiment 1.2: Exploring the effect of Fish Eye on EMNIST

For comparing our results with single patch fisheye with first scale of size 8X8 and second scale of size 16X16 resized to 10X10, we use the baseline of MNIST with single patch of size 10X10 and skyline of 2 patches of size 8X8 and 16X16 (resized to 8X8). To compensate with extra coverage, we consider 6 glimpses else total pixels seen by our model would had exceeded no of pixels in the image for 8 glimpses (i.e. 8X10X10 > 784 i.e. 28X28, in our case it is 6X10X10 < 784).

Following are the results obtained after running the experiments for 50 epochs.

Approach	Valid accuracy	Best Epoch No.
Baseline	86.37	11
FishEye	87.89	44
Skyline	91.13	92

Thus, we also observe here that we outperform CNN baseline (87.05 %) on EMNIST data as discussed in Section Experiment 1.1

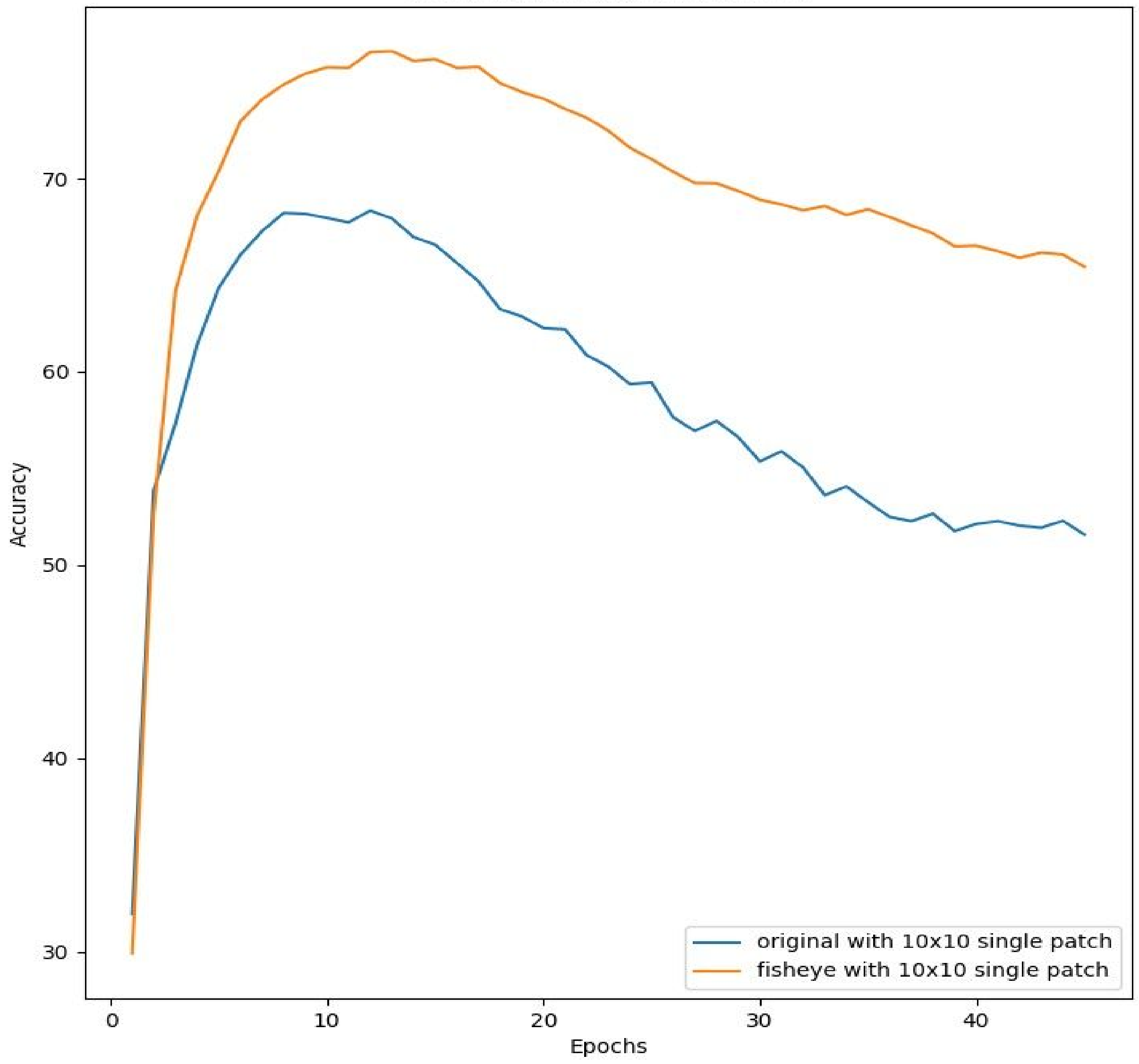


Fig 1.1b Comparison of training accuracies of each epoch of fisheye approach with effective single 10x10 patch.

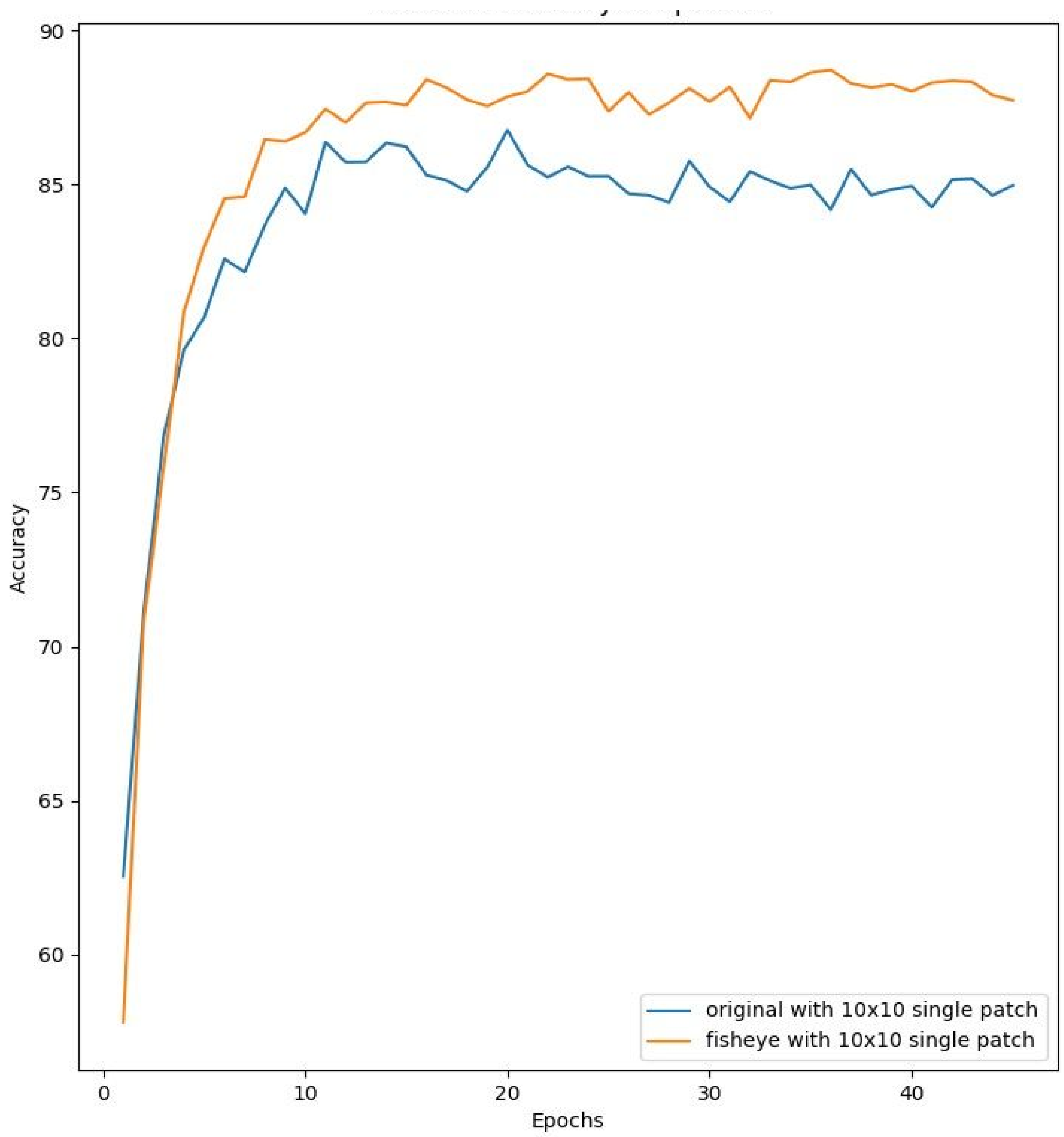


Fig 1.1c Comparison of test accuracy of each epoch of fisheye approach with effective single 10x10 patch.

From the above figures, improvement can be clearly observed by fisheye approach as opposed to usual (mentioned in paper) taking high resolution images simply. Size of the both patches are same i.e 10x10, so both approaches require equal computations during processing. Hence, fisheye approach seems more efficient. Fisheye approach delivers better accuracy in same computations.

Experiment 2: Experiments on IAM Dataset

IAM dataset consists of 115'320 isolated and labeled words[7]. Sample images are shown in Figures 2.1a,b&c.

We faced various challenges while working on IAM Dataset as follows:

1. Starting Location: The default starting location was random and hence reading from left to right was not trivial.
2. Initialization: The location network initializes the first location to the centre of image due to hidden layer being initialized to zeros, and location being extracted from the hidden layer.
3. Wandering of Masks across characters.

Following are the results obtained after running the experiments for 50 epochs.

Approach	Test accuracy	Best Epoch No.
Left Middle Starting Location (3 patches, 8 glimpses)	39.93	5 (not best)
Better Initialization of Hidden Layers (3 patches, 8 glimpses)	44.22	37
Better Initialization of Hidden Layers (1 patch, 4 glimpses per char)	49.00	97
Better Initialization of Hidden Layers With Fish Eye (1 patch, 4 glimpses per char)	46.32	87 (not best)

Wandering Correction Via negative Reward (1 patch, 4 glimpses per char)	14.986	19 (not best)

Table 2: Results on IAM Datasets

Following experiments discuss the ways we tried to mitigate these problems.

Experiment 2.1: Changing the Starting Location

Our experiments would not converge since the starting location was choosen randomly, whereas there is a reading order in English from left to right which we can exploit. Hence we initialized the starting location to left most middle part of the image.

With this, we obtained the accuracy of 39.9% as shown in the first row of Table 2.

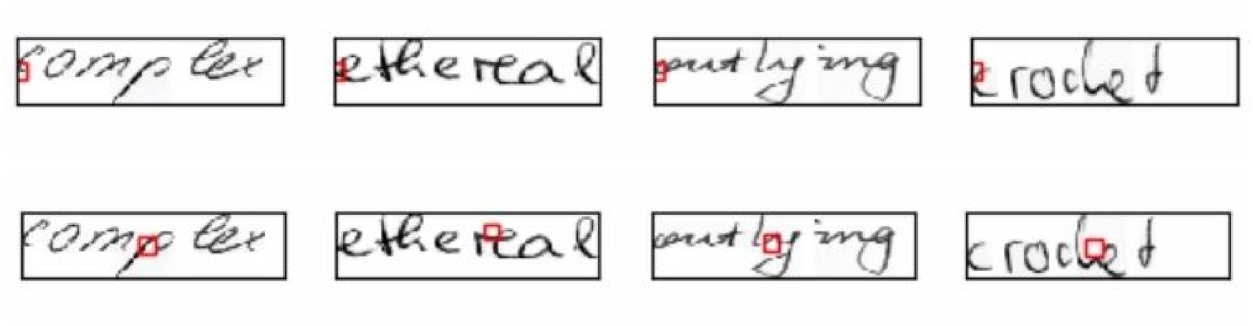


Fig2.1a: First two glimpses (1st on top, 2nd on bottom) after training for 1 epoch, for the network when initialized with proposed starting location. For rest of epochs the mask could not explore the left middle part of image.

Experiment 2.2: Better Initialization of hidden states

Original approach initialized hidden layers with zeros, due to which the location network would output the second location to centre of image (referred to as [0,0] in location space) as discussed in previous section. We initialized half of the hidden layers with -1 to start with.

This solved the problem discussed in previous section after a few epochs of training as shown below. The results obtained are shown in 2nd row of Table 2.

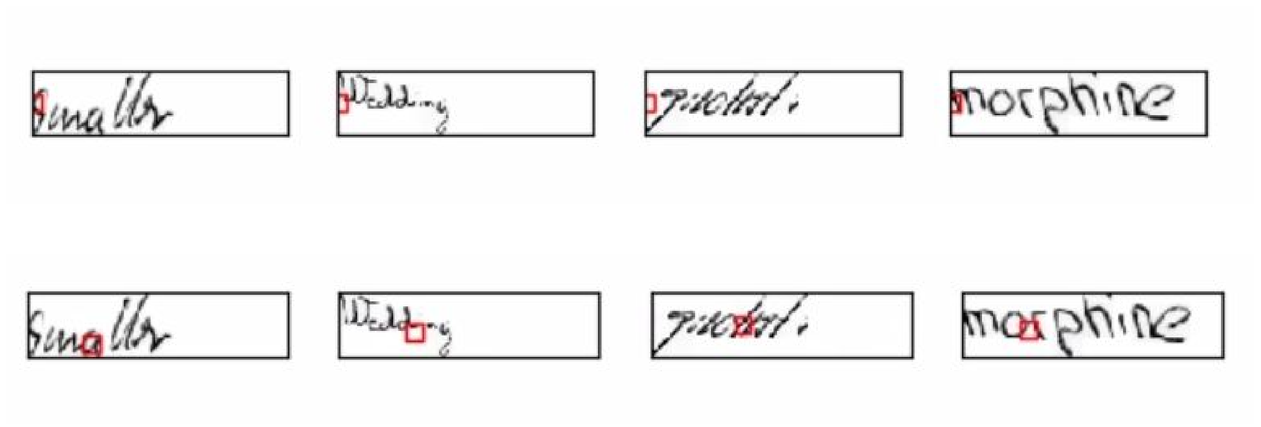


Fig2.1b: First two glimpses (1st on top, 2nd on bottom) after training for 37 epochs (or best model).

Since experiments started converging at this point, we used lesser glimpses (4) for each character and single patch as shown in brackets of 1st column of Table2.

The experiments were conducted with fisheye approach as well with same size as in EMNIST experiments. The results obtained are shown in 4th row of Table 2. We obtain better results as compared to 2nd row (multiple scales may create more wandering problem).

Experiment 2.3: Wandering Correction

It can still be observed that Mask Wanders unexpectedly, as we train it with supervision of characters sequentially, via action network. with for 8 glimpses over each character. We try to solve this problem by adding negative rewards proportional to distance between consecutive glimpse locations. Unexpectedly we observe lesser accuracy but the above problem of wandering is resolved. The probable reason for such behaviour can be attributed to the high penalization of our reward scheme. In future work we would like to fine tune the proportionality constant to mitigate this problem.

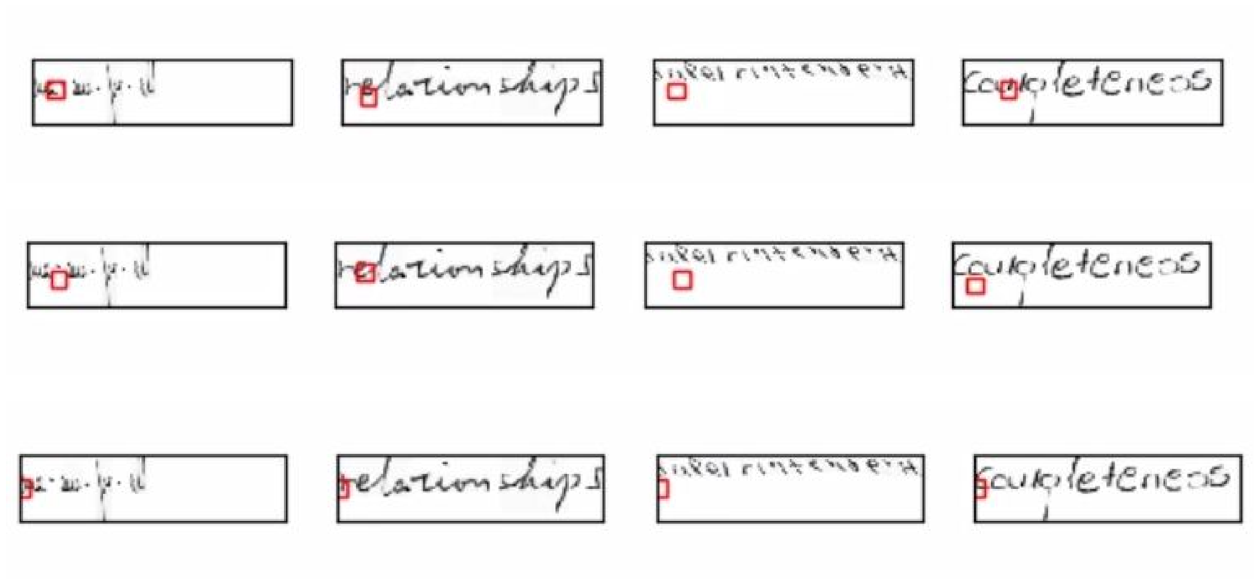


Fig2.1c: First two glimpses (1st on top, 2nd in middle) and last glimpse (bottom) after training for 19 epochs (or best model).

Conclusion

For the experiments we conclude that:
1. No. of glimpses affect the performance and extra glimpses may lead glimpses to wander in unnecessary regions.
2. Fisheye approach works better than single patch with same architecture parameters.
3. On word images, problems of improper starting location, initialization and wandering of glimpses can be corrected by setting the starting location to leftmost middle part of image, initializing the half of hidden layers of LSTM with -1, and giving negative rewards proportional to distance between consecutive glimpse locations. In future work we would like to fine tune the proportionality constant to mitigate this problem.

IMP: All our codes are available in attached folder. In each sub folder go to plots/ram_* to visualise videos. Logs are also available in file names log*

Acknowledgement

Kevnzakka code
https://github.com/kevinzakka/recurrent-visual-attention

References:

[1] Denil, M., Bazzani, L., Larochelle, H. and de Freitas, N., 2012. Learning where to attend with deep architectures for image tracking. *Neural computation*, *24*(8), pp.2151-2184.
[2] Mnih, V., Heess, N. and Graves, A., 2014. Recurrent models of visual attention. In *Advances in neural information processing systems* (pp. 2204-2212).
[3] Sutton, R.S., McAllester, D.A., Singh, S.P. and Mansour, Y., 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems* (pp. 1057-1063).
[4] Cohen, G., Afshar, S., Tapson, J. and van Schaik, A., 2017. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*.
[5] Jimmy Ba, Volodymyr Mnih, Koray Kavukcuoglu. Multiple Object Recognition with Visual Attention
[6] IAM Dataset http://www.fki.inf.unibe.ch/databases/iam-handwriting-database