Glimpse Attention Models

Victor Ng
victorng@stanford.edu

Abstract

This study investigates the application of transformer architectures to recurrent attention models (RAMs) for image classification tasks. Traditional RAMs utilize recurrent neural networks to sequentially process glimpses of an image, but recent advances in transformer architectures suggest potential improvements in modeling global dependencies between glimpses. A transformer-based attention mechanism that replaces the RNN components in RAMs with an encoder-decoder transformer architecture, where glimpse vectors are processed through an encoder and the resulting representations are used as memory for an autoregressive decoder. Experiments on the Street View House Numbers (SVHN) dataset demonstrate that while transformer-based RAMs achieve competitive accuracy (94.81% with optimal hyperparameters), they require significantly more computational resources than their RNN counterparts. Comprehensive ablation studies on patch size, number of glimpses, learning rates, and hidden dimensions, reveal that 12 pixel patches and 3 or 6 glimpses provide optimal performance.

1. Introduction

Convolutional neural networks and vision transformers have both been very successful in image classification and object recognition. However, one of the primary drawbacks of these networks is their linear or quadratic scaling with image resolution, requiring less than ideal solutions such as down-sampling images or running on more powerful hardware.

Semi-sparse image recognition models like Recurrent Attention Models [1, 2] have taken inspiration from the way humans perform image recognition tasks. RAMs are sequence models that at each step process a local view of the image, called a glimpse, which it then uses to update an internal hidden state and output the next glimpse location. The process continues until the model decides that there are no more objects to process.

The benefit of RAMs is that they scale well with image resolution. While an initial low-resolution feature map is required to provide initial context, the full-resolution image is only processed with sparse glimpses which do not scale with resolution.

This work aims to improve upon the performance of RAMs by replacing the RNN with a transformer encoder-decoder architecture [3]. A key limitation of RNNs is that they compress glimpse information into fixed-size hidden states, potentially losing important spatial relationships between glimpses. Instead of each glimpse being used to update the hidden state of an RNN, each glimpse will be processed as a token and passed into a transformer decoder block. The output token will then be used to predict the next glimpse location.

2. Related Works

First, [1] introduced RAMs which use a recurrent neural network to process sequentially sampled glimpses from an image. Their model demonstrated that attention-based processing can be more efficient than convolutional approaches for large images, as computation remains constant regardless of image size. The RAM architecture consists of a glimpse network, a recurrent core network, a location network, and an action network, trained end-to-end using reinforcement learning.

Second, [2] extended the RAM framework to address multiple object recognition tasks, showing how attention mechanisms can effectively transcribe multi-digit sequences from real-world images. Their approach highlighted the ability of recurrent attention models to outperform convolutional networks on cluttered images while using fewer parameters and less computation.

Third, [3] presented the Transformer architecture that relies entirely on attention mechanisms without fixed-size hidden states. Their work showed that self-attention mechanisms allow modeling of dependencies without regard to distance in the input sequence, enabling more effective capture of global relationships.

3. Method

The proposed method, which is called Glimpse Attention Models (GAM), replaces the RNN in the original RAM with a transformer decoder architecture. This fundamental change aims to better capture the relationships between glimpses through global attention mechanisms rather than compressed hidden states.

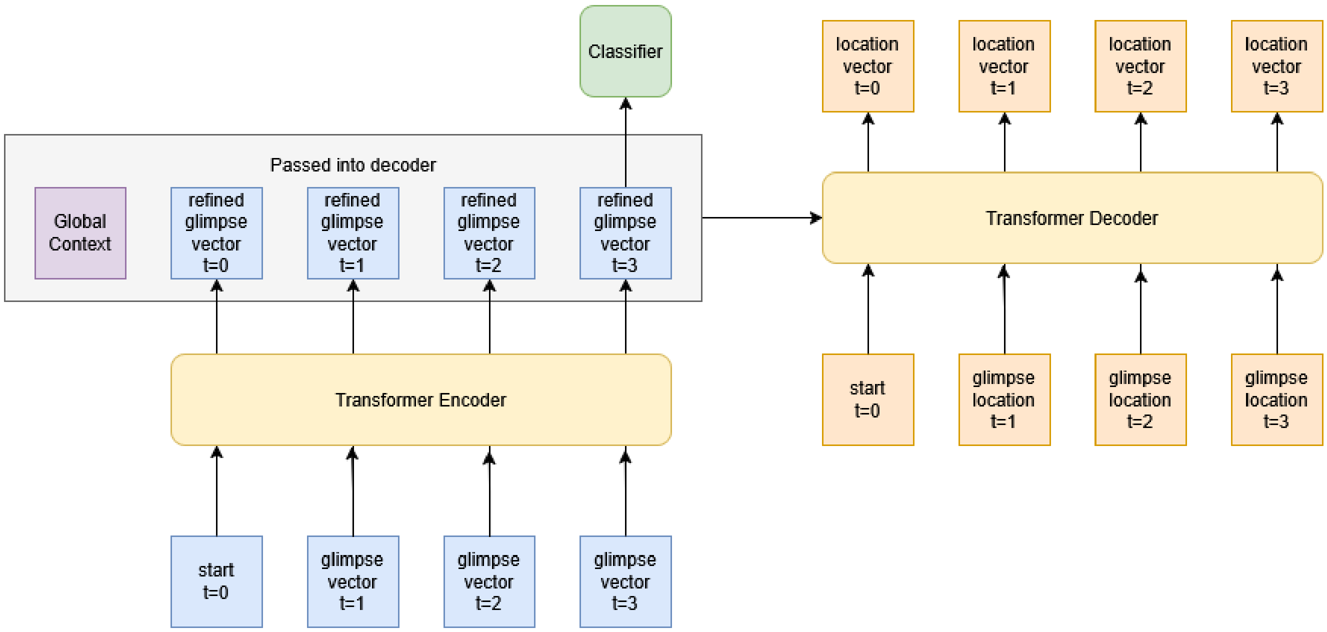

Figure 1: Structure of Transformer encoder-decoder network

3.1. Architecture

GAM follows the general structure of RAM, with similar sub-networks.

Glimpse Network

The glimpse network extracts useful features from the current glimpse x_n at location $l_n = (x_n, y_n)$ in the input image. For each glimpse, k square patches centered at location l_n are extracted, with each patch having progressively lower resolution as they move outward.

Following [2], the glimpse network consists of three convolutional layers followed by a fully connected layer. Additionally, the location tuple is transformed by an MLP into the same dimensionality as the processed glimpse. Then both the “what” and “where” information is combined via element-wise multiplication.

$$g_n = G_{image}(x_n) \odot G_{loc}(l_n) \quad (1)$$

Additionally, glimpses are refined by passing them into a transformer encoder layer, to let glimpses attend to each other. This allows both glimpses from different parts of the image to attend to each other.

Context Network

The context network provides features of the entire input image and is used to determine where to take the first glimpse. The network takes a down-sampled version of the entire input image and outputs a fixed length vector C_n. The goal of the context network is to provide reasonable ideas of where interesting parts of the network are. Following [2], it consists of three convolutional layers.

Transformer Network

Instead of using an RNN to process glimpse representations sequentially, an autoregressive transformer encoder-decoder architecture is used (Figure 1).

For any given glimpse, all glimpses taken so far are passed into a transformer encoder, where each glimpse vector can attend to each other. Glimpses of different parts of the image can communicate with each other, enriching their representations. Critically, the global context from the context network is not passed into the encoder for reasons which will be explained later. Only after the encoder is the global context combined with the enriched glimpse vectors to be passed as memory.

On the decoder side, all predicted (x,y) locations from the location network are passed in as queries. The self-attention layers of the decoder allow the different glimpse locations to attend to each other, allowing the prediction of the next location to be conditioned on what areas have already been visited. There is also cross-attention with the enriched glimpse vectors and global context, which allows the network to, from a semantic point of view, determine where to look next.

Unlike RNNs, which compress all previous glimpse information into a fixed-size hidden state, the transformer's self-attention mechanism allows each new glimpse to directly access and attend to all previous glimpses, potentially capturing more complex relationships.

Location Network

The location network predicts the next glimpse location based on the most recent location vector from the transformer decoder. It consists of an MLP that maps the most recent location vector to the parameters of a normal distribution. During training, the next location coordinate tuple l_{n+1} is sampled from this distribution, while during inference the mean of the distribution is used directly.

Classification Network

The classification network outputs a class label for the image based on the glimpses of the image and consists of an MLP. The MLP takes as input the last glimpse token from the transformer encoder and outputs the class label.

The output of the transformer decoder is specifically not used for classification as its outputs have been attended to by the initial context vector C_n. The network could learn to rely on this context vector instead of combining information from the glimpses. This is undesirable behavior as the network should learn to depend on its glimpses for high-quality image information rather than the low-resolution C_n.

3.2. Forward Pass

For a given image (see Figure 2), the image is first downsampled to a fixed-resolution and passed into the Context Network to generate the global context. Next using the global context, a learned start token for the glimpse vectors, and a learned start token for the locations, the Transformer model outputs a location vector. This location vector is then passed into the location network to predict the x,y location of the next glimpse. The glimpse is sampled from the image and passed into the Glimpse Network, which outputs a glimpse vector. The global context, accumulated glimpse vectors, and accumulated locations are then passed into the Transformer model once again. This cycle repeats until the target number of glimpses occurs. The number of glimpses is a hyperparameter.

3.3. Training Algorithm

The model is trained via reinforcement learning due to the non-differentiable nature of location selection. Referencing [2], REINFORCE [7] is used to train the model.

For each object in the image, the model will be rewarded with a reward of 1 if the object was properly classified and a reward of 0 otherwise. When rewarded, the model will be updated to maximize the probability of each glimpse being chosen.

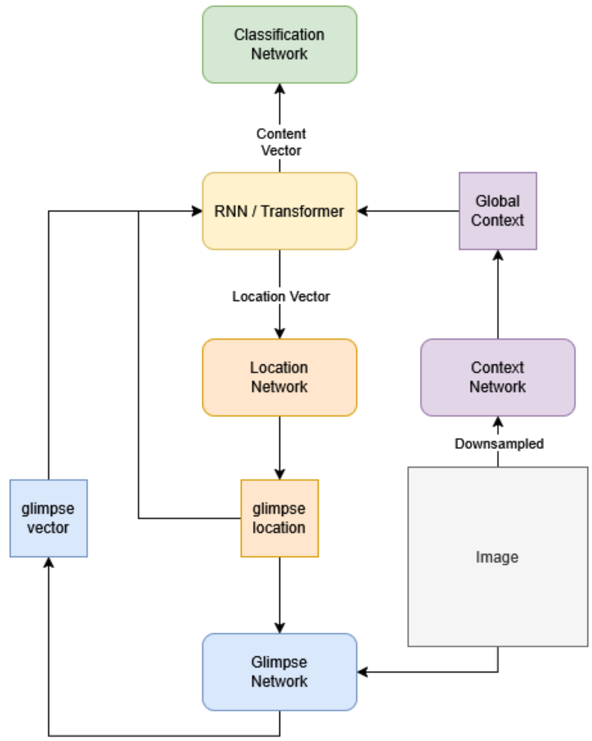

Figure 2: Logical flow of RAM

4. Dataset

The publicly available single-digit street view house number (SVHN) dataset [5] consists of 32x32 images of digits taken from pictures of house numbers. The train set consists of both the "train" and "extra" datasets, for a total of 604,388 images. Of these images 10% are set aside as validation images to select hyperparameters. Testing is done on the "test" split of 26,032 images.

Figure 3: Example image from SVHN

5. Experiments

5.1. Experiment Setup

Both architectures are implemented from scratch in PyTorch to ensure fair comparison and precise control over architectural details. The baseline RNN model follows the original RAM architecture [2] using LSTM cells, while the transformer model implements the encoder-decoder structure described in the methods section.

All experiments use identical context network architectures consisting of three convolutional layers with 64, 64, and 128 output filters and kernel sizes of 5, 3, 3 respectively, followed by fully connected layers that produce 1024-dimensional glimpse representations. The location network generates 2D coordinates through a two-layer MLP with tanh activations, while the action network uses a standard classifier head with softmax output. The REINFORCE reward baseline is implemented as a separate network that predicts expected rewards given glimpse sequences.

Training employs the Adam optimizer with gradient clipping to ensure stable convergence. A batch size of 128 is used across all experiments and the final models are trained for 100,000 iterations.

Performance is measured using top-1 classification accuracy, measured as the number of images with a correct prediction divided by the number of images.

5.2. Hyperparameter Optimization

Hyperparameter optimization was conducted across learning rates, hidden dimensions, glimpse sizes, and number of glimpses. Hyperparameter optimization is run for 10,000 steps and the hyperparameter with the best validation accuracy is used in the final model.

5.3. Learning Rate

Various learning rates are searched over, revealing 1e-4 as the optimal choice. Learning rates adjacent to 1e-4 show promise but either converge too slowly or have a learning rate that is too high also leading to slower convergence.

Table 1: Learning Rate vs. Validation Accuracy

Learning Rate	Validation Accuracy
1e-2	19.59%
1e-3	71.30%
1e-4	83.54%
1e-5	75.53%

5.4. Hidden Dimension

Hidden dimension analysis demonstrates that 64-dimensional hidden states provide optimal performance for our task setup. Interestingly, performance degrades when the hidden dimension exceeds 256. One hypothesis is that the model overfits well enough that it memorizes the training data and can successfully classify training data even with degenerate glimpse sequences, leading to poor generalization on the test set.

Table 2: Hidden dimension vs validation accuracy

Hidden Dimension	Validation Accuracy
64	84.43%
128	63.8%
256	17.2%
104	17.1 %

5.5. Number of Glimpses

One of the key advantages of attention models is their ability to trade off compute for performance by changing the number of glimpses. Both the number of glimpses and glimpse size are ablated to evaluate this tradeoff.

Patch sizes of 8x8, 12x12, and 16x16 pixels are evaluated in combination with 3 and 6 glimpses per image. Increasing patch size and number of glimpses increases the amount of computation required per image but theoretically allows the model to see more of the image.

In practice, increasing patch size and number of glimpses does increase performance. The performance boost via increase in patch size is most noticeable when moving from 8x8 to 12x12 patches and saturates after. Increasing number of glimpses is most noticeable with a smaller patch size as well, with the largest difference of 2.51% occurring at patch size 8x8.

This is likely because with larger patch sizes, the model can already view enough of the image to make an accurate classification. As the patch size decreases, the model sees less of the image at a time and benefits more from the increased number of glimpses.

Table 3: Patch size and number of glimpses vs. Validation Accuracy

	# Glimpses	
Patch Size	3	6
8x8	92.24%	94.75%
12x12	96.68%	97.02%
16x16	96.85%	96.08%

5.6. Speed

One tradeoff of Transformers is their more intensive compute. To compare the throughput of the RNN based model and the Transformer based model, representative models with similar validation accuracies are benchmarked. The RNN is the baseline model mentioned in the experiment setup and the Transformer model is

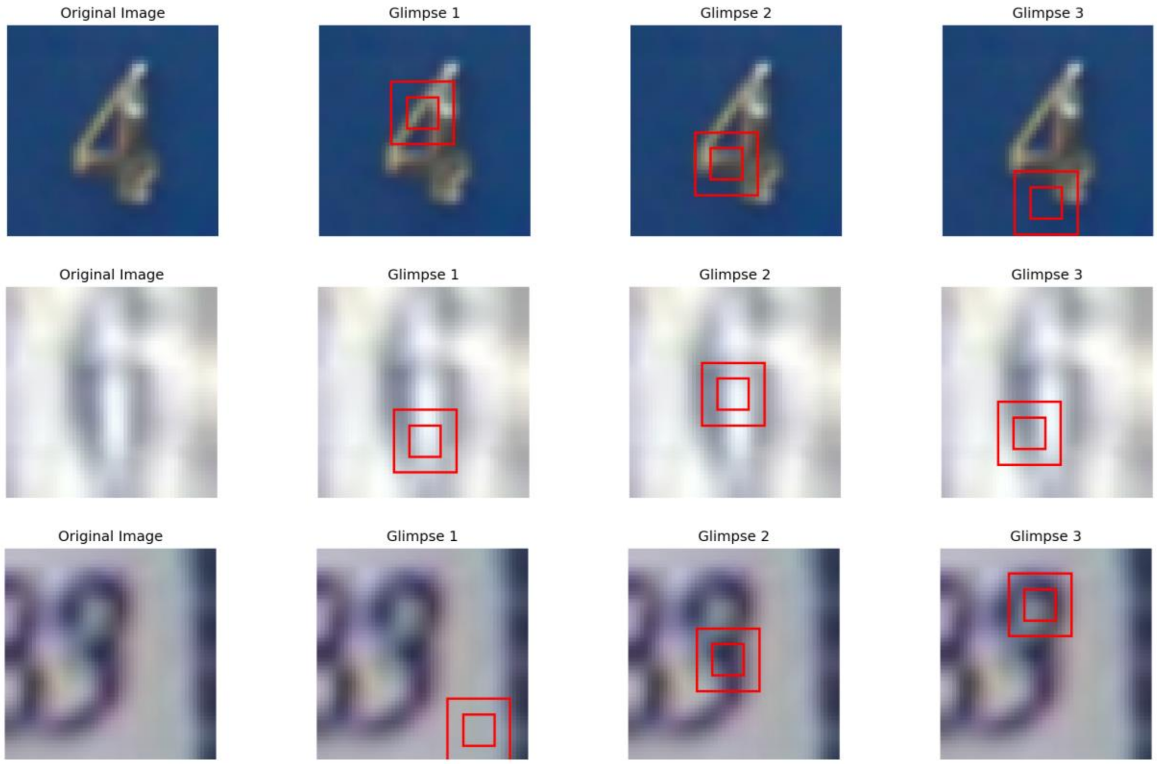

Figure 4: Each row represents the glimpse sequence for a 3-glimpse Transformer based RAM. The red squares represent the sampled patches for that glimpse. **Row 1)** A correct classification. The model takes a top-down approach to its glimpses. **Row 2)** An incorrect classification. The model classified the 6 incorrectly as a 0. The model only noticed the bottom half of the 6, leading to a misclassification. **Row 3)** A correct classification. The model is mistakenly drawn to the bottom-right of the image, likely due to the strong edge on the right side. It corrects its mistake in the following glimpses and successfully classifies the image as a 3.

using a hidden dimension of 64 with one encoder layer and one decoder layer.

Table 4: Throughput of RAM models on an RTX 3080

Model	Images per second
RNN (3 glimpses 8x8 patch)	9088
RNN (6 glimpses 8x8 patch)	6656
Transformer (3 glimpses 8x8 patch)	5248
Transformer (6 glimpses 8x8 patch)	3328

Even with a much lighter model, the Transformer based RAM is 1.7x slower than the RNN architecture, due to the quadratic nature of transformer attention.

5.7. Model Comparison and Performance Analysis

The final transformer model is trained with a hidden dimension of 64, 6 glimpses, and a patch size of 12x12, representing the best performing hyperparameters in the search.

The Transformer based RAM achieves a similar performance to the RNN, with a test accuracy that is within measurement noise of the RNN baseline.

Both methods successfully learn a policy that explores the image via glimpses.

One hypothesis is that Transformer's advantage of RNNs is their ability to model long sequences without compression and the ability to model more complex relationships. The single-digit SVHN task may be too simple a task for Transformer networks to see a benefit over RNNs. Tasks like optical character recognition may be a better target for Transformer RAMs due to their potentially very long sequences.

Table 5: Test accuracy of RAMs

Model	Test Accuracy
RNN Baseline (6 glimpses 12x12 patch)	94.72%
Transformer RAM (6 glimpse 12x12 patch)	94.81%

5.8. Qualitative Analysis of Glimpses

The model generally takes either a top-down or bottom-up flow when choosing its glimpses. The first row of Figure 4 shows a typical glimpse trajectory, starting from the top of the image and incrementally moving downward to view the points of interest. Since single-digit SVHN contains centered digits, most movement of the glimpse occurs in the y direction, as digits are generally taller than they are wide.

The second row of Figure 4 shows a failure case where the model misclassifies a 6 as a 0. The glimpse provides some intuition as to why, as no glimpse ever views the top portion of the 6, and the bottom half of a 6 is very similar to a 0. This either indicates that the context network did not represent the top half of the 6 well or that the Transformer network did not identify the top-half to be semantically relevant.

The third row of Figure 4 shows a recovered classification, where the network mistakenly chooses the bottom-right of the image as one of the glimpses before recovering and correctly predicting the image as a 3. The model was likely drawn to the strong edge on the right side of the image. It then managed to recover by returning the remaining glimpses to the actual object in the image. This demonstrates a robustness of RAM models, where having multiple glimpses provides redundancy, allowing the model to make some mistakes while still being able to classify the image correctly.

6. Conclusion

This work presents a systematic investigation of transformer architectures for recurrent attention models in image classification. The encoder-decoder transformer architecture is competitive with traditional RNN architectures while maintaining the sequential glimpse-based processing paradigm that makes attention models computationally efficient for large images.

The experimental results demonstrate that transformer-based attention models can achieve competitive performance on the SVHN dataset, with optimal configurations reaching 94.81% accuracy using 6 glimpses of 12-pixel patches. However, this comes at significant computational cost, with inference times 1.7x higher than equivalent RNN models due to the quadratic scaling of attention mechanisms.

Several key insights emerge from the evaluation. First, glimpse count optimization reveals diminishing returns beyond 3 glimpses, suggesting that effective attention strategies can be learned with relatively few sequential observations. Second, patch size selection noticeably impacts performance, with 12-pixel patches providing optimal information density for digit recognition tasks.

Future work could explore several promising directions. Investigating more complex visual reasoning tasks that better exploit transformer capabilities could reveal scenarios where the computational overhead is justified by performance gains. Additionally, investigating efficient attention mechanisms that reduce the quadratic scaling while preserving global modeling capabilities remains an important research challenge.

7. Contributions & Acknowledgements

All code, models, experiments, and reports were created, trained, and run by me. I had no collaborators, and this project was done solely for CS231n.

The baseline RAM model and training was reimplemented from scratch and based on the code from [6]. Additionally, code for positional encoding and Transformer data flow was referenced from CS231n Assignment 3.

References

[1] V. Mnih, N. Heess, A. Graves, K. Kavukcuoglu, and G. Deepmind, “Recurrent Models of Visual Attention.”, 2014.

[2] J. Ba, V. Mnih, and K. Kavukcuoglu, “Multiple Object Recognition with Visual Attention”, 2015.

[3] A. Vaswani et al., “Attention Is All You Need,” 2017.

[4] A. Paszke et al., “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” CoRR, vol. abs/1912.01703, 2019

[5] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng, Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*

[6] T. Tianyu, Visual Attention Model [Source code] 2017 https://github.com/tianyu-tristan/Visual-Attention-Model/tree/master?tab=readme-ov-file

[7] R. J. Williams, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning,” Machine Learning, vol. 8, pp. 229–256, 2004