From Vision to NLP: A Merge

Alisha Rege
Computer Science
Stanford University
amr6114@stanford.edu

Payal Bajaj
Computer Science
Stanford University
pabajaj@stanford.edu

Abstract

The study of artificial intelligence can be simplified into one goal: trying to mimic/enhance human senses. This paper attempts to combine computer vision and natural language processing to create a question answer system. This system takes a question and an image as input and outputs a response to the answer based on how the RCNN understands the question asked. The system correlates the question with the image by leveraging attention and memory mechanisms.

Mentor: Arun Chaganty

1 Introduction

With the recent progress of machine learning and deep learning, technology has progressed to the state that we can, not only, make the computer understand words and their grammatical state, but also, make the computer understand how to correlate questions to images and form a logical response. We propose an architecture that takes as input natural language queries about an image and returns as output the answers to those queries. The input output relationship is presented in Figure 1.

The purpose of the project is to understand the impact of introducing attention and episodic memory mechanisms to solve the Visual QA problem. In addition to this, we will experiment with multiple hyper-parameters and demonstrate improvements compared to the hyper-parameters used a current state of the art paper [4].

2 Related Work

Dynamic Memory Networks: This approach uses a form of a recurrent neural network to learn features and associate them with answers. It was introduced as a general architecture for question answering (QA) in [4] where the authors employ different aspects such as input representations or memory components to be analyzed and improved independently. The work was extended to Visual QA in [5] where an input module was added for the task.

Visual Genome: Using crowd sourcing they developed a system that uses WordNet to train the question answering system. This paper[1] is mostly about crowd sourcing annotations and does not propose a robust question answering system. It hopes to encompass all possible questions by having a big dataset.

We have combined both these papers for the purpose of this project.

Figure 1: Example for Visual QA

3 Dataset Description

We worked with a dataset developed by Facebook called Visual Genome[6]. This research paper created their own dataset to deal with question answering systems. It is comprised of 108,077 Images, 5.4 Million Region Descriptions, 1.7 Million Visual Question Answers, and 2.8 Million Attributes. For the purpose of this project, we work with a subset of this dataset consisting of 2000 images and their corresponding questions (approximately 5000 questions). Fig 2 provides an example of a data point:the objects are tagged in the image, the relationships between objects are established using a network, and many labels are derived.

4 Methodology

The input to the system is a question in natural language and an image on which the question is based. We focus on one-word answers for the purpose of this project. The task of the system is to predict the one-word answer of the question. The question Q is a sequence of words $(w_1, w_2, ..., w_{L_Q})$

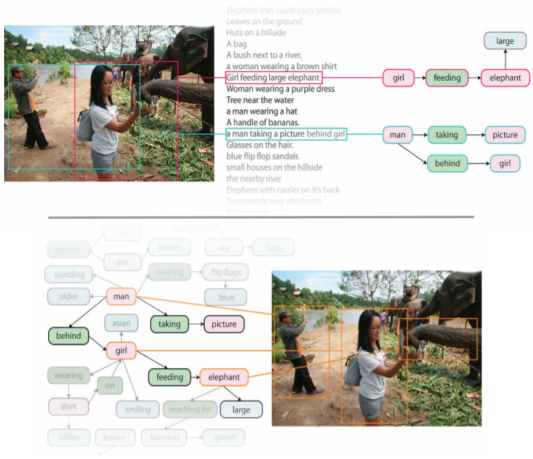

Figure 2: Visual Genome Data Example

where L_Q is the length of the question or the number of words in the question. Each question is processed using a question input module which is as follows:

Question Input Module The question is processed using a RNN(GRU) using the following equations:

$$q_t = GRU(L[w_t], q_{t-1}) \tag{1}$$

where L represents the embedding matrix for the words, w_t represents the word at t^{th} index in the question. q_0 is initialized to a zero vector and the final question representation is the last vector generated using this process:

$$q = q_{L_Q} \tag{2}$$

4.1 Baseline

The baseline approach is summarized in 3. For the question, we use the question input module to obtain the vector q. For images, we use the pre-trained VGG-19 model and tested it based on VGG-19 architecture [1]. The visual embedding matrix is derived from the last pooling layer which a $14 \times 14 \times 512$. The mean of this embedding matrix is taken as a dimensional vector denoted by i. Both q and i are mapped to vocabulary size by linear transformations using $W^{(q)}$ and $W_{(i)}$ respectively, where the W matrices form the parameters of the model and are learned using training data. The result of this transformation is passed through a Softmax layer to calculate the predicted answer.

4.2 DMN+

Dynamic Memory Networks were initially introduced as Textual Question-Answering systems[4]. This work was extended to improve performance and extend the system for Visual Question-Answering using DMN+[5]. The work-flow of the system is summarized in fig 4.
The question input module is same as before. The image input module takes in an image and produces a set of facts for each image. These facts represent the embeddings of the image and are learned while training the model.

1. **Image Input Module**
   This module is summarized in 5 and has the following components:
   Visual Feature Extraction: The input is passed through a pre-trained CNN model based upon the $VGG-19$ model[1]. All the images are rescaled to 448 $\times$ 448 and and the output from the last pooling layer is taken. This has dimensionality $d = 14 \times 14 \times 512$. The dimensions are such because the pooling layer divides the image into a grid of 14×14, resulting in 196 local regional vectors of d = 512.

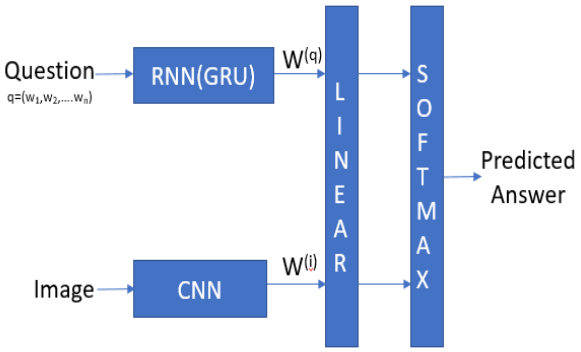

Figure 3: Work-flow for Baseline

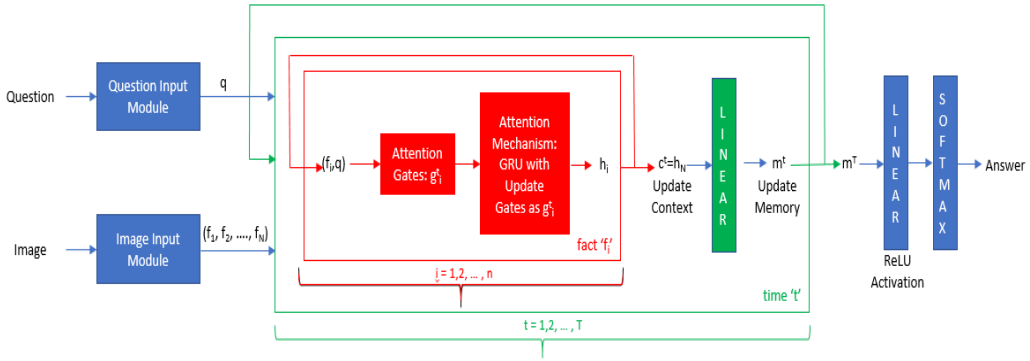

Figure 4: Work-flow for DMN+

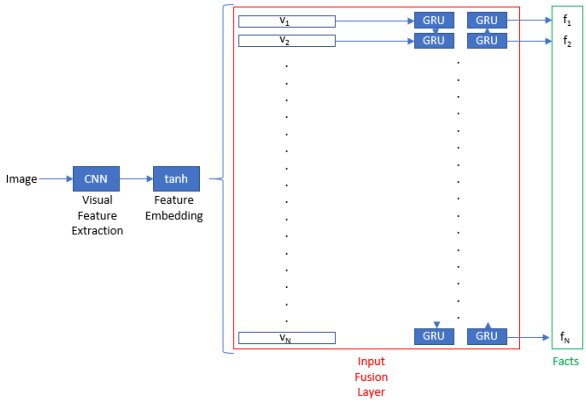

Figure 5: Image Input Module

Feature Embedding: A linear layer with $tanh$ activation is added to project the local regional vectors to the textual feature space used by the question vector q.
Input Fusion Layer: A bi-directional GRU is used to learn globally aware facts $F = (f_1, f_2, ..., f_N)$ from the local regional vectors. This allows to capture spatial information from neighboring image patches.

The question vector q and the set of facts F are processed iteratively using the attention and memory mechanism to get the context vector and episodic memory at each time-step. We study the impact of number of time-steps in the results section.

2. **Attention Gates:** This attention by associating a single scalar value, the attention gate g_i^t with each fact f_i during pass t. This is computed by allowing interactions between the fact and both the question representation and the episode memory state. '.' denotes element-wise multiplication.

$$z_i^t = [f_i.q; f_i.m^{t1}; |f_i - q|; |f_i - m^{t-1}|] \tag{3}$$

$$Z_i^t = W^{(2)} tanh(W^{(1)} z_i^t + b^{(1)}) + b^{(2)} \tag{4}$$

$$g_i^t = \frac{exp(Z_i^t)}{\sum_{k=1}^{M_i} exp(Z_k^t)} \tag{5}$$

3. **Attention Mechanism: Attention based GRU** To produce the contextual vector c^t used for updating the episodic memory state m^t, we use the final hidden state of the attention based GRU.

$$h_i = g_i^t.h_i' + (1 - g_i^t).h_{i-1} \tag{6}$$

4. **Memory Updates:** The contextual vector c^t, the episodic memory at previous time m_{t-1} and the vector representation of the question q are used to update the episodic memory m^t at time t using a linear transformation followed by rectified linear unit activation.

$$m^t = ReLU(W^t[m^{t1}; c^t; q] + b) \tag{7}$$

5. **Answer Module:** The final state of the memory is passed through a linear transformation followed by softmax to predict the answer.

$$y^t = softmax(W^{(a)}, a^t) \tag{8}$$

5 Experiments

We have used $GLoVE$ embeddings[2] to initialize vector representation of words in the question. Since all the questions have different lengths, or number of words, we use a padding mechanism to process a batch of questions: we pad the questions with a special token word UNK to symbolize the unknown word till the maximum length of question in that batch. These embeddings are treated as variables and learned in the classification process. Tensorflow's $dynamic\_rnn$ function provides and option to learn based on varying sequence length and thus we use that to learn embeddings for words in questions of different lengths. For VGG, we have used [8] as a template and improved it for our system.

The training data is split into 70% training, 20% development and 10% test data. We work with a batch size of 128 training examples. The best model we achieve takes about 48 hours end-to-end.
Metrics:

1. Accuracy: This metric calculates the fraction of times the predicted answer was correct.
2. WUPS Score: The WUPS calculates the similarity between two words based on their longest common subsequence in the taxonomy tree. If the similarity between two words is less than a threshold then a score of zero will be given to the candidate answer. We have used the Wordnet[7] database to measure the similarity between the words.

Summary: Table 1 summarizes the results for the baseline and DMN+ models. Using the baseline we achieve an accuracy of 26.2% on development set and 25% on test set. DMN+ + Paper refers to the the DMN+ paper, and we achieve an accuracy of 17.2% using the parameters specified in the paper. We achieve 27.5% and 26.5% on development and test sets using hyper-parameter tuning.

Fig 6 represents training and development accuracy for the three models: Baseline, DMN+ with parameters same as paper(marked as DMN+ Paper) and DMN+ we implemented with hyper-parameter tuning. Our model outperforms the other two and converges faster. Also, the development set accuracy is consistently higher for our model than the other two models. Fig 7 demonstrates WUPS scores achieved by all models. As discussed later in the error analysis, we predict synonyms of actual answers for multiple questions and that is why we get consistently high WUPS score values.

Models	Training Accuracy	Dev Accuracy	Test Accuracy	WUPS
Baseline on Visual Genome Data	98	26.1	25.0	0.960
DMN+ on Visual Genome Data	97	27.6	26.5	0.964
Dropout Value: p=0.0				
DMN+ + Paper(Parameter Tuning)	96.2	18.5	17.2	0.956
Dropout Value: p=0.1				
DMN+ + Paper(Parameter Tuning)	98.6	22.9	22.3	0.97
Dropout Value: p=0.2				
DMN+ + Paper(Parameter Tuning)	98.5	23.1	22	0.971
Dropout Value: p=0.3				
DMN+ + Paper(Parameter Tuning)	98	22.8	25.2	0.97
GLoVE Embedding Size: 50				
DMN+ + Paper(Parameter Tuning)	97.4	22.1	17.2	0.517
GLoVE Embedding Size: 100				
DMN+ + Paper(Parameter Tuning)	97.3	25.5	25	0.964
GLoVE Embedding Size: 200				
DMN+ + Paper(Parameter Tuning)	98.5	27.6	26	0.969
Data Size: 500				
DMN+ + Paper(Parameter Tuning)	93.5	22.5	19.9	0.947
Data Size: 1000				
DMN+ + Paper(Parameter Tuning)	97.4	22.1	22.3	0.963
Data Size: 1500				
DMN+ + Paper(Parameter Tuning)	98	22	22.8	0.97
Data Size: 2000				
DMN+ + Paper(Parameter Tuning)	97.5	23.4	22.8	0.966

Table 1: Test Accuracy for Different Dropout Probabilities

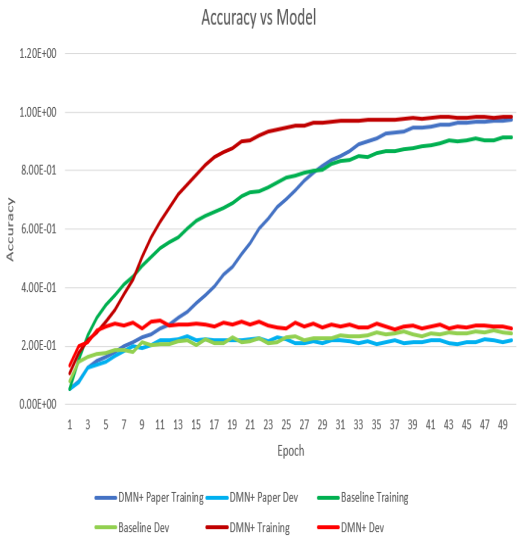

Figure 6: Accuracy of multiple models

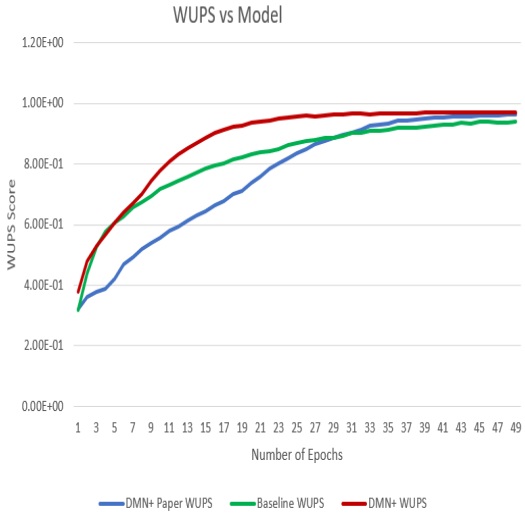

Figure 7: WUPS of multiple models

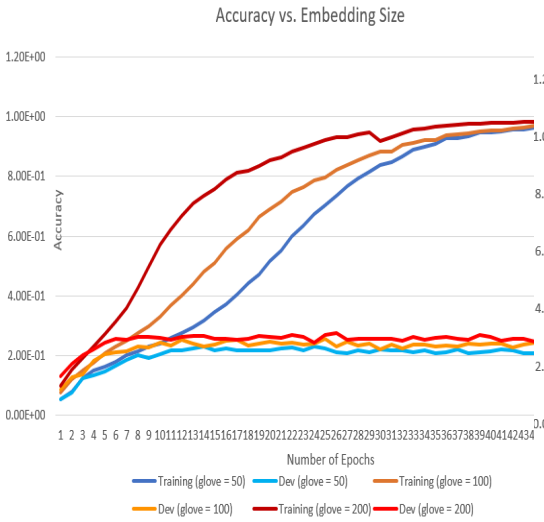

Figure 8: Accuracy for Different Embedding Sizes

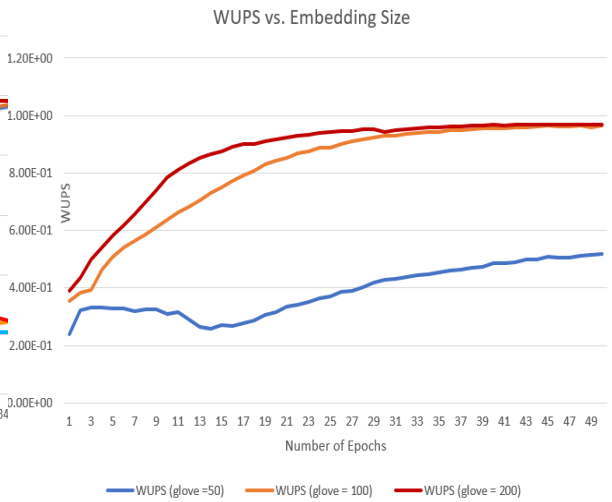

Figure 9: WUPS for Different Embedding Sizes

5.1 Effect of Glove Embedding Size

We experimented with different lengths of the GLoVE vectors for word in the questions and the results are summarized in fig 8 and 9. The plot demonstrates that WUPS is better for larger embedding sizes and also the model converges faster with increase in embedding size. This is expected because the larger the embedding size, more the number of parameters the system can learn and therefore it adapts to the training data faster leading to higher accuracy faster.

5.2 Effect of Dropout Probability

The dropout is applied to the input layer only. No dropout leads to poor performance and slow convergence as can be seen with the line associated with $p = 0.0$. The performance for $p = 0.1$ performs the best among all values we experimented with. For higher values of p, we ran it for lower epochs but we can see the convergence is slower compared to 0.1, this is because eliminating more connections slows the capability of the network to train efficiently.

5.3 Effect of Amount of Data

The variation of accuracy and WUPS with training data size has been shown in fig 12 and 13. Bigger data samples lead to faster convergence when the number of images considered are 1000 or more, whereas fewer than 1000 images lead to lower accuracy and WUPS scores.

5.4 Effect of Parameter T

The parameter T denoted the number of times the context end episodic memory is updated in the training process for every batch. We repeated the training process for multiple values of T and the results are summarized in fig 14. We noticed that the results don't vary much as value of T varies. This may be because not many more associations need to be learned after the first iteration.

6 Error Analysis

Most of the errors of the system were caused by prediction of synonyms instead of the exact word as the answer. For instance, the system predicts $eating$ instead of the specific verb $grazing$.

A lot of errors were caused due to the system's inability to distinguish between visual items in the image. This implies that the system often predicts an answer incorrect options from the image such as $Pizza$ instead of $Fork$ and $Lettuce$ instead of $Beans$.

There were very few bizarre numerical errors such as incorrect counts. For instance, the system predicts 24 as the answer when the correct count was 3. One possible reason of this could be the image features unable to distinguish between the numbers in the image.

These issues can be handled by using better image representations and using more corresponding to such images where the system makes such error so that oversampling under-represented data can help the system to learn to distinguish intricacies of such images. Further, these issues can be addressed by using more images and question-answers so that the system learns the parameters in a better way so as to be able to distinguish between answers.

We observed high deviations in the development set accuracy ranging from 27% to 29% so we expected our test accuracy to be in that range as well but the best test accuracy we achieved was 26.5%. We believe that the test accuracy could be higher, had more experiments been run; however since each iteration of training takes 48 hours, we chose to report after one run.

7 Conclusion and Future Work

In this paper, we use Dynamic Memory Networks for a Visual QA system. These networks use attention and episodic memory mechanisms on representations of questions and images to predict answers. We tuned multiple hyper parameters and demonstrate comparable performance to the papers and better performance than the baseline.

We attempted to run our model for 2000 images and maximum GLoVE embeddings with higher iterations of episodic memory module (T=10) but the experiment couldn't finish due to RAM constraints.

One possible direction for extending this work is to run the model with more global descriptors of images, that is more facts per image, and over bigger datasets to achieve better performance.

Overall, this architecture is sufficient to prove that Visual QA solutions are progressing at a rapid rate; although technology needs to improve more to allow more hyper-parameter training, we achieve an accuracy that is parallel to some state-of-the art papers and this work builds upon one such paper.

References

[1] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[2] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." EMNLP. Vol. 14. 2014.

[3] Malinowski, Mateusz, and Mario Fritz. "Towards a visual turing challenge." arXiv preprint arXiv:1410.8027 (2014).

[4] Kumar, Ankit, et al. "Ask me anything: Dynamic memory networks for natural language processing." CoRR, abs/1506.07285 (2015).

[5] Xiong, Caiming, Stephen Merity, and Richard Socher. "Dynamic memory networks for visual and textual question answering." arXiv 1603 (2016).

[6] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016.

[7] Miller, George A. "WordNet: a lexical database for English." Communications of the ACM 38.11 (1995): 39-41.

[8] https://github.com/machrisaa/tensorflow-vgg?files=1

Appendices

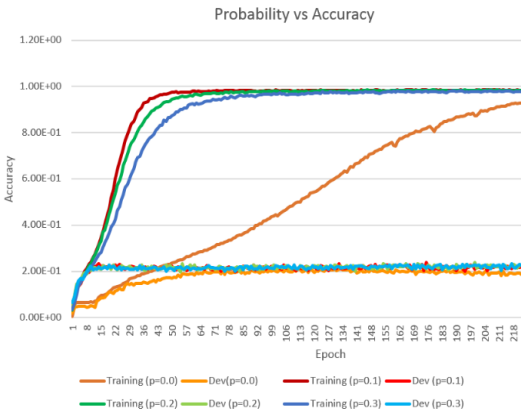

Figure 10: Accuracy for Different Dropout Probabilities

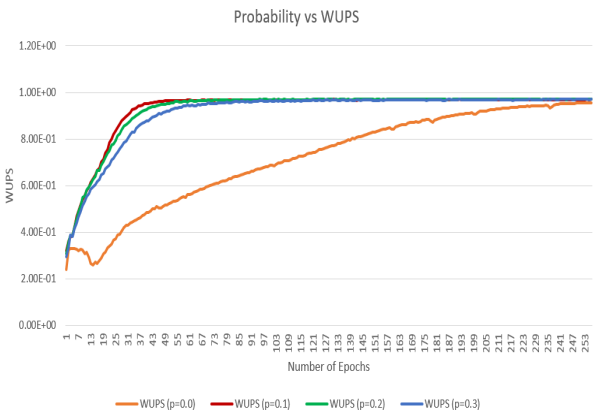

Figure 11: WUPS for Different Dropout Probabilities

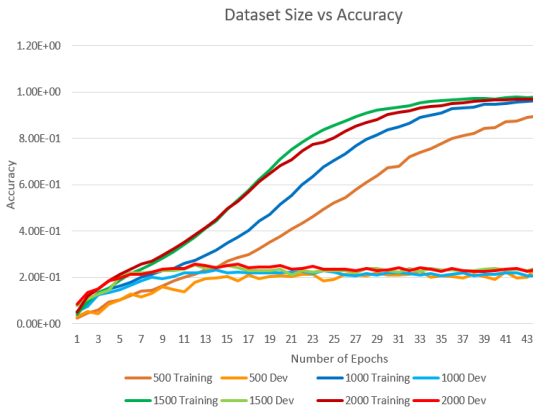

Figure 12: Accuracy for Different Training Data Sizes

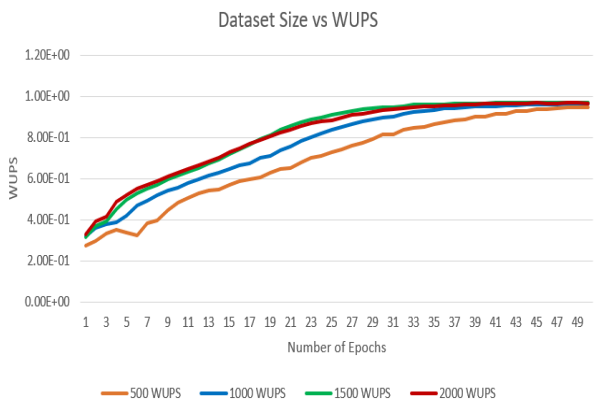

Figure 13: WUPS for Different Training Data Sizes

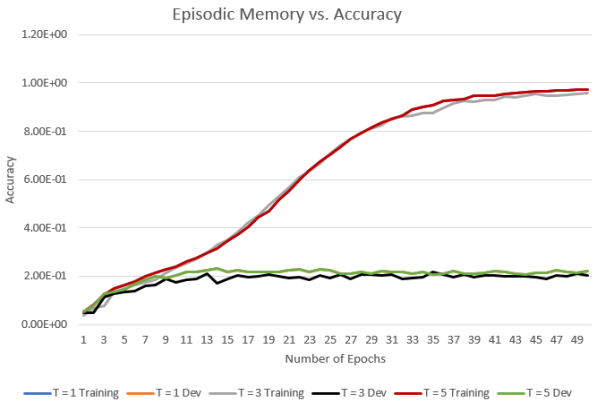

Figure 14: Accuracy for Different Values of T