10703 Course Project Final Report: Observe, Attend and Act: Attention Mechanisms in DQN

Yilun Chen
yilunc1
Robotics Institute
`chenyilun16@cmu.edu`

Rui Zhu
rz1
Robotics Institute
`rz1@andrew.cmu.edu`

Hanyue Liang
hanyuel
Robotics Institttue
`hanyuel@andrew.cmu.edu`

Abstract

In this project, we explore by introducing attention mechanism into classic Deep Q-network(DQN) architecture. The attention mechanism includes both temporal and spatial attention models. We implement Recurrent DQN (DRQN) as the baseline and inject attention mechanism into this model. Temporal attention model collects the LSTM outputs at previous T time steps and weighs these outputs according to an optimized weight vector. The combined outputs are used for computing the Q values. Spatial attention model instead weighs the low-level CNN features according to their spatial locations in the input image and then combines them into a context vector before feeding into the recurrent module. Our experiments on Seaquest show great performance boost over DQN and DRQN with our new method. Also both temporal and spatial attention models add intuitive interpretability to the model regarding where and when the agent's attention is focusing on while making decisions.

1 Introduction

1.1 Problem

In this project, our goal is to improve the basic DQN algorithm in Atari game playing by considering history sequence and adding attention mechanism. We started with Double Deep Q-network(DDQN) with dueling network and then Deep Recurrent Q-network(DRQN). We explored further by introducing attention mechanism from recent vision advances [14] into classic Deep Q-network(DQN) architecture. The attention mechanism includes both temporal and spatial attention models. We implemented Recurrent DQN (DRQN) as the baseline and inject attention mechanism into this model.

1.2 Related Work

Mnih et al. [8] proposed Deep Q-network(DQN) first time proving that DQN outperforms all previous algorithms and is comparable to human player in certain experimental settings.

Van et al. [11] proposed double DQN to reduce the common overoptimism problem in original DQN algorithm making the learning more stable, reliable and suitable for large scale application.

Dueling DQN represented two estimators: one for state value function and one for state dependent action advantage function generalizing learning across action without changing reinforcement learning algorithms, which resulted in significant improvement over previous DQN algorithms as shown in Atari domain[13].

DQN have proven to be successful at learning policies. However, it has problems in learning with longer term information. To address that issue, Deep Recurrent Q-network was proposed [4]. DRQN replaces the first fully connected layer with a recurrent LSTM. The results showed that DRQN is able to perform better in integrating information through time.

As an extension of DRQN, adding temporal attention onto DRQN was proposed by Chen et al.[9]. Two forms of attention were introduced, including a linear attention and a global attention, whose output was the predicted Q-values.

Xu et al. [14] introduced two spatial attention based models, "soft" and "hard", that learned to describe the content images. The "soft" attention was able to trained by standard back propagation methods. The "hard" stochastic attention could be trained by maximizing an approximate variational lower bound. The learned attention was shown to be able to give more insights of the model generation process and to align well with human intuitions.

2 Methods

In this section, we first briefly review Double Duelling DQN as our baseline. Then we introduce how to add recurrence to DQN, which we call Deep recurrent Q-network(DRQN). Next we illustrate our novel architecture of bringing attention into DRQN. This includes two kinds of model: one considers spatial attention and another considers temporal attention.

2.1 DQN, Double DQN and Dueling DQN

Consider a Markov Decision Process with state $s \in \mathcal{S}$, action $a \in \mathcal{A}$, reward $r \in \mathcal{R}$. The goal of a RL agent is to maximize its expected return by learning a policy π, which maps from state $s \in \mathcal{S}$ to action $a \in \mathcal{A}$. The future discounted return R_t at timestep t can be represented by $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r'_t$. In reinforcement learning settings, We want to learn an optimal Q-function to approximate maximum expected return, $Q^*(s,a) = \max_\pi \mathbb{E}[R_t | s_t = s, a_t = a]$.

1. *DQN*: Deep Q-Network (DQN) [8] approximates the optimal Q function with a deep neural netwrok by using the Bellman equation as an iterative update,

$$Q^*(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q^*(s', a')|s,a] \tag{1}$$

   A Q-network can be trained by minimizing a sequence of loss functions at the current time step t,

$$L_t(\theta_t) = \mathbb{E}_{s,a,r,s'}(y_t - Q_{\theta_t}(s,a))^2 \tag{2}$$

2. *Double DQN*: Double DQN [11] removes upward bias caused by $\max_a Q(s,a,w)$. The current Q-network w is used to select actions and the older Q-network w^- is used to evaluate actions

$$I = (r + \gamma Q(s', \arg\max_{a'} Q(s', a', w), w^-) - Q(s,a,w))^2 \tag{3}$$

3. *Dueling network*: Dueling network [13] split Q-network into two channels: action-independent value function V(s,v) and action-dependent advantage function A(s,a,w). The idea is to generalize learning across actions without imposing any change to the underlying DQN.

$$Q(s,a) = V(s,v) + A(s,a,w) \tag{4}$$

In this paper, all DQN implementations are combined with Double DQN and Duelling network.

2.2 Deep Recurrent Q-network (DRQN)

The concept of deep recurrent Q-network (DRQN) was recently introduced by Hausknecht et al. [5] as a combination of recurrent neural network (RNN) and the deep Q-network. The main contribution is to replace the input to the Q-network which is concatenation of several recent observations, as a longer sequence of recent frames, and append a RNN module (of which the most popular being LSTM) to process the temporal outputs of the Q-network. The Q values used for choosing among actions are required from the final output of the RNN module, as shown in Fig. 1.

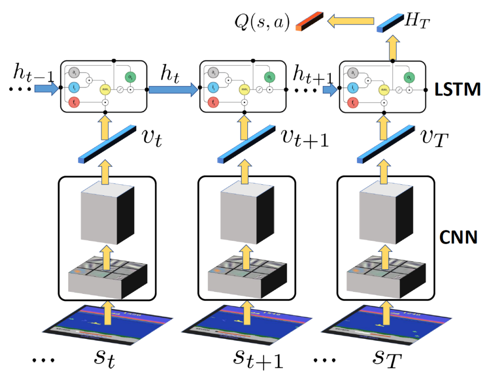

Figure 1: **Architecture of DRQN.** DRQN convolves three times over a single-channel image of the game screen. The resulting activations are processed through time by an LSTM layer. During training, the LSTM are trained with an unrolling of T=10 frames. LSTM outputs of each frame become Q-Values after passing through a fully-connected layer.

As compared with traditional DQN, DRQN offers several advantages including ability to handle longer input sequences and exploration of temporal dependencies, as well as better performance in case of partially observable experiences.

2.3 Attention model

We are also interested in exploring the effectiveness of attention mechanisms in the case of DRQN. Attention model [7] is recently exploited in the topics of image caption generation [14], and object tracking [3]. They show effectiveness in compressing the input information to enable training speedups, and also offering an interpretable visualization about "where" and "what" the agent chooses to focus on.

Attention models are explored in the case of RNN in two streams: temporal attention model and spatial ones. We have managed to combine these two streams of attention in DRQN with some adaption, fully explained as follows.

2.3.1 Temporal Attention DRQN

Temporal attention has recently been justified to boost performance in sequence to sequence learning for classification tasks [9]. However, there is no current literature showing if it can learn more accurate q-values in the context of reinforcement learning.

Inspired by [2], We apply our temporal attention over the output of the LSTM layer in DRQN model, as shown in Fig. 2. Temporal attention mechanism learns weights for LSTM outputs in different time steps. It takes in T hidden states output from LSTM as $\{h_{T+1-i}\}_{i=1}^{T}$. Next it optimizes a vector v_a and compute the weight $\{a_{T+1-i}\}_{i=1}^{T}$ with inner products.

$$a_{T+1-i} = softmax(v_a^T h_{T+1-i}) \tag{5}$$

Then we can compute the combined context vector H_T for computing Q values.

$$H_T = \sum_{i=1}^{T}(a_{T+1-i}h_{T+1-i}) \tag{6}$$

The learned $\{a_{T+1-i}\}_{i=1}^{T}$ can be interpreted as the importance of LSTM output at a given frame. Therefore the optimizing process can be seen as learning to choose which observations are relatively more important than the rest for learning the Q values.

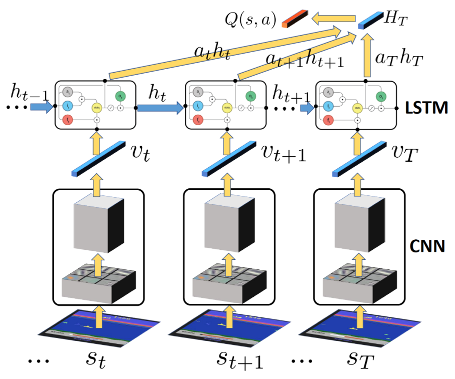

Figure 2: **Architecture of Temporal Attention DRQN.** The context vector H_T for computing Q values is computed as a linear combination of T LSTM output features instead of only one LSTM output feature in DRQN. The weights can be optimized through backpropgation during training.

Temporal attention DRQN is superior than DRQN in the sense that it explicitly considers the past T frame LSTM output features for computing Q value, while this information is only passed by implicitly through LSTM in original DRQN. By increasing the value of T, the model can consider longer sequence of history frames and thus make better action choice.

2.3.2 Spatial Attention DRQN

Spatial attention models [14] learn weights for different areas in an image, and the context feature used for decision making is a combination of spatial features according to the learned weights. According to how to model the "combination", spatial attention models are divided into "hard" attention and "soft" attention [14]. Similar to ideas in [10], we use a "soft" version attention for DRQN.

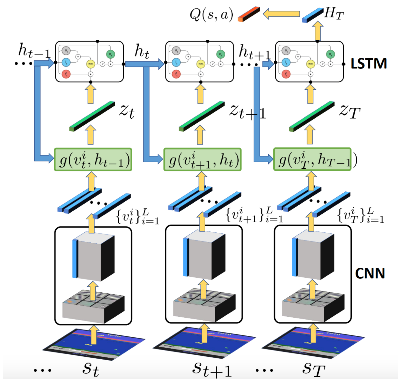

Figure 3: **Architecture of Spatial Attention DRQN.** Feature maps extracted by CNN will first be transformed by region vectors.

The Spatial Attention DRQN architecture, as shown in Fig. 3, contains three types of network: convolution, attention and recurrent. At time step t, suppose the CNN produces a set of D feature maps with size $m \times m$. The attention network transforms these feature maps into a set of region vetors $\{v_t^i\}_{i=1}^T, v_t^i \in \Re^D, L = m^2$ and output $z_t \in \Re^D$, called a context vector. In soft attention mechanism, we assume the context vector z_t can be represented by a weighted sum of all region vectors $\{v_t^i\}_{i=1}^T$, each of which corresponds to the features extracted by the CNN at a different image region. The weights in this sum are chosen in proportion to the importance of this vector (aka the extracted feature in this image region) learnt by the attention network g.

The attention network has region vector v_t^i and hidden state h_{t-1} produced by LSTM layer as input and outputs the corresponding weights for this vector. The attention network g contains one fully-collected layer with Tanh activation and one fully-collected layer followed by softmax layer. The g network can be represented as follows.

$$g(v_t^i, h_{t-1}) = Softmax(Linear(Tanh(W_v v_t^i + W_h h_{t-1}))) \tag{7}$$

Then we can compute the context vector z_t using g.

$$z_t = \sum_{i=1}^{L} g(v_t^i, h_{t-1}) v_t^i \tag{8}$$

The context vector z_t is then fed into a LSTM layer and the output of final LSTM is used to compute the Q values.

The attention network can be seen as a mask over the CNN feature map, where it reweights the region features to get the most informative features for computing Q values. Thus the Spatial Attention DRQN acquires the ability to select and focus on the most important regions when making the action choice. This also helps to reduce the total number of parameters in the network and computational operations needed in training and testing.

3 Experiments

For the experiments, we choose three games from them OpenAI Gym environment: `Frostbite-v0`, `Assault-v0`, and `Seaquest-v0`. Among these three, `Frostbite-v0` is used in the original DRQN paper [4] for its partially observable feature. And both `Assault-v0` and `Seaquest-v0` are shooting games, where temporal attention with recurrent input could potentially help better determine the flying enemies, and ideally spatial attention would be able to find the target explicitly and give a more interpretable mask over the target.

3.1 Network Architecture and Training Details

We denote each fully-connected layer *fc(d)* by its output dimension d, and 2D convolution layer by *conv(k, c, s)* representing kernel size of *k*, strides of *s* across two spatial axes, and *c* channels.

The vanilla DQN is composed of three convolution layers and two fully connected ones: *conv(8, 32, 4)*, *conv(4, 64, 2)*, *conv(3, 64, 1)*, *fc(512)*, *fc(number of actions)*. The convolution layers are rectified with ReLU activation. We resize the observed frames to $84 \times 84 \times 1$ image, and stack most recent 4 frames into a 4-channel image before feeding into the network.

For DRQN, the last by two fully-connected layer *fc(512)* is replace with a LSTM layer with a hidden unit size of 512. We used the most recent 10 frames for the input of LSTM, thus an unroll of LSTM with 10 time steps is performed when we map out the entire architecture.

For Temporal Attention DRQN, an additional vector v_a is learned with the same dimension as the hidden unit, and it is connected to the graph according to Equation 5 and Equation 6.

For (soft) Spatial Attention DRQN, the weight on two fully connected lyaers is instead learned to assign weights to the CNN features and the output of previous LSTM hidden states, according to Equation 7.

In training the above architectures, we implement with Tensorflow, and use an Adam optimizer with a learning rate of 0.0001 across all experiments. All other paramaers are consistent with the standard parameters suggested in homework 2.

3.2 Qualitative Results

We show the visualizations of how our attention mechanisms brings interpretability to the tasks. We pick the game of `Seaquest-v0` to showcase this point in Fig. 4 and Fig. 5.

As can be seen from the results, in the scenario given by Fig. 4, increasing weights are assigned to the input sequence when the submarine is hunting a prey. In this case, a single last frame does less help than a sequence frame in that, by looking back into the past (by looking at weighted sum of features of several frames before), the agent is able to determine the trajectory of the prey for it to better aim

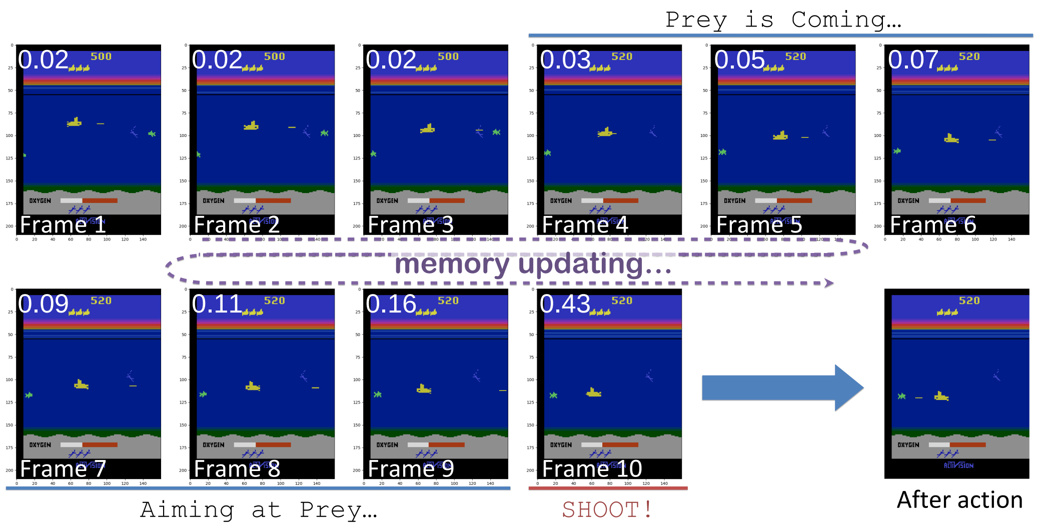

Figure 4: **Qualitative results of Temporal Attention DRQN for the game of `Seaquest-v0`.** The numbers in the upper-left corner of each image is weight assigned to that frame (higher weight indicates more importance).

at. This is also supported by the weights: the last frames enjoy weights of at least 0.09, which is not trivial.

As can be seen from the results, in the scenario given by Fig. 5, the region of potential preys are targeted by the spatial attention model. After one of the three preys is lost, the attention models quickly learns to refocusing at the rest two, and eventually figures out when and where to launch the strike. By learning a mask over the input domain (more specially, the CNN features with correspondence to the input image), the agent is able to grab the context relevant to the task and speeds up training after that point. This is also demonstrated below in the part of quantitative results.

3.3 Quantitative Results and Analysis

Fig. 5 gives the evaluation results of four models on `Seaquest-v0`. All recurrent models perform consistently better than the baseline vanilla DQN. For the DRQN with attention mechanisms, both temporal attention and spatial attention take extra effort to pick up. But once attention is acquired, they help speed up training by casting lights into more relevant information instead of taking in everything. As a result, after surpassing the baseline model, they overtake the baseline model by a large margin.

	`Seaquest-v0`	`Assault-v0`	`Frostbite-v0`
DQN	2011	**812**	302
DRQN	2320	792	342
Temporal Att. DRQN	**6002**	803	**443**
Spatial Att. DRQN	3103	811	293

Table 1: **Quantitative results: evaluation awards per episode during training for all games.**

As is shown in Table 1, the recurrent models especially attention ones are significantly better at the games of `Seaquest-v0` and `Frostbite-v0`. However for the game of `Assault-v0`, they does not help much somehow. This observation echos with [4] and [6] where recurrent models might hurt performance as opposed with simple input with stacked frames, and recurrent mechanisms might fail to learn in some scenarios, either because the environment is actually fully observable, or the spatial attention does not capture the right feature.

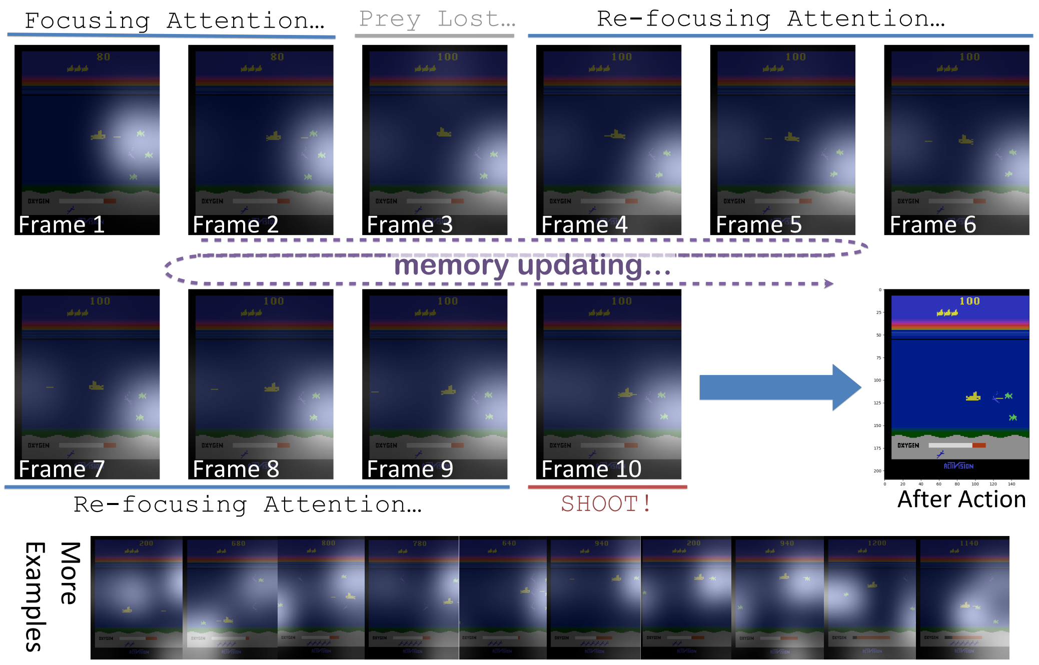

Figure 5: **Qualitative results of Spatial Attention DRQN for the game of** `Seaquest-v0`**.** A mask over the input domain is learned by the attention mechanism. Brighter colors indicate higher weights. The weights are smoothed with an Gaussian kernel in the visualization.

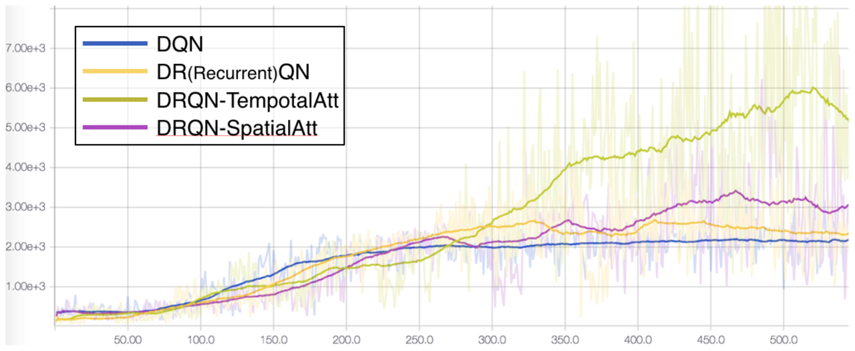

Figure 6: **Quantitative results: evaluation awards per episode during training for the game of** `Seaquest-v0`**.**

4 Discussion

In this project, we explore the role of attention mechanisms in training Atari games.

Specifically, we first introduced spatial attention which is prevalent in computer vision tasks (e.g. video/image captioning) into recurrent DQN, where a spatial mask is learned over the input to weight the spatial features according to their relevance to the task, in an hope to speed up training by learning from filtered features. Second, we mimic the temporal attention usually found in machine translation [1] or chatbots [12] where a LSTM is used for modeling temporal dependence and the temporal outputs are linearly weighted for different importance.

We demonstrate attention models with recurrent architectures outperform vanilla DQN in two out of the three Atati games we test on. We are also able to explain how these mechanisms help in the training tasks by looking at the visual output from the attention model.

The drawback of our methods are: (1) they are not universally applicable to all environments in that they may backfire when applied; (2) they introduce more learnable weights into the network which might potentially slow down training speed per iteration; (3) the LSTM is often unstable in training; a possibly way out this is to stabilize LSTM and expand its capacity by introducing bidirectional LSTM or using layers of LSTM instead of just one layer.

References

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

[2] Clare Chen, Vincent Ying, and Dillon Laird. Deep q-learning with recurrent neural networks. *stanford cs229 course report*, 2016.

[3] Misha Denil, Loris Bazzani, Hugo Larochelle, and Nando de Freitas. Learning where to attend with deep architectures for image tracking. *Neural computation*, 24(8):2151–2184, 2012.

[4] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015.

[5] Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015.

[6] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

[7] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[9] Wenjie Pei, Tadas Baltrušaitis, David MJ Tax, and Louis-Philippe Morency. Temporal attention-gated model for robust sequence classification. *arXiv preprint arXiv:1612.00385*, 2016.

[10] Ivan Sorokin, Alexey Seleznev, Mikhail Pavlov, Aleksandr Fedorov, and Anastasiia Ignateva. Deep attention recurrent q-network. *arXiv preprint arXiv:1512.01693*, 2015.

[11] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR, abs/1509.06461*, 2015.

[12] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[13] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[14] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81, 2015.