Visual Reasoning with Multi-hop Feature Modulation

Florian Strub[1(✉)], Mathieu Seurin[1], Ethan Perez[2,3], Harm de Vries[2], Jérémie Mary[4], Philippe Preux[1], Aaron Courville[2,5], and Olivier Pietquin[6]

[1] Univ. Lille, CNRS, Inria, UMR 9189 CRIStAL, Villeneuve-d'Ascq, France
florian.strub@inria.fr
[2] MILA, Université de Montréal, Montreal, Canada
[3] Rice University, Houston, USA
[4] Criteo, Paris, France
[5] CIFAR Fellow, Toronto, Canada
[6] Google Brain, Mountain View, USA

Abstract. Recent breakthroughs in computer vision and natural language processing have spurred interest in challenging multi-modal tasks such as visual question-answering and visual dialogue. For such tasks, one successful approach is to condition image-based convolutional network computation on language via Feature-wise Linear Modulation (FiLM) layers, i.e., per-channel scaling and shifting. We propose to generate the parameters of FiLM layers going up the hierarchy of a convolutional network in a multi-hop fashion rather than all at once, as in prior work. By alternating between attending to the language input and generating FiLM layer parameters, this approach is better able to scale to settings with longer input sequences such as dialogue. We demonstrate that multi-hop FiLM generation significantly outperforms prior state-of-the-art on the GuessWhat?! visual dialogue task and matches state-of-the art on the ReferIt object retrieval task, and we provide additional qualitative analysis.

Keywords: Deep learning · Computer vision · Multi-modal learning
Natural language

1 Introduction

Computer vision has witnessed many impressive breakthroughs over the past decades in image classification [15,27], image segmentation [30], and object detection [12] by applying convolutional neural networks to large-scale, labeled datasets, often exceeding human performance. These systems give outputs such as class labels, segmentation masks, or bounding boxes, but it would be more natural for humans to interact with these systems through natural language. To this

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01228-1_48) contains supplementary material, which is available to authorized users.

© Springer Nature Switzerland AG 2018
V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11209, pp. 808–831, 2018.
https://doi.org/10.1007/978-3-030-01228-1_48

ReferIt	*GuessWhat?!*	
- The girl with a sweater	Is it a person?	Yes
- The fourth person	Is it a girl?	Yes
- The girl holding a white frisbee	Does she have a blue frisbee?	No

Fig. 1. The ReferIt and GuessWhat?! tasks. In ReferIt, a single expression identifies the selected object (with blue bounding box), while GuessWhat?! identifies objects through a sequence of yes/no questions. (Color figure online)

end, the research community has introduced various multi-modal tasks, such as image captioning [48], referring expressions [23], visual question-answering [1,34], visual reasoning [21], and visual dialogue [5,6].

These tasks require models to effectively integrate information from both vision and language. One common approach is to process both modalities independently with large unimodal networks before combining them through concatenation [34], element-wise product [25,31], or bilinear pooling [11]. Inspired by the success of attention in machine translation [3], several works have proposed to incorporate various forms of spatial attention to bias models towards focusing on question-specific image regions [47,48]. However, spatial attention sometimes only gives modest improvements over simple baselines for visual question answering [20] and can struggle on questions involving multi-step reasoning [21].

More recently, [38,44] introduced Feature-wise Linear Modulation (FiLM) layers as a promising approach for vision-and-language tasks. These layers apply a per-channel scaling and shifting to a convolutional network's visual features, conditioned on an external input such as language, *e.g.*, captions, questions, or full dialogues. Such feature-wise affine transformations allow models to dynamically highlight the key visual features for the task at hand. The parameters of FiLM layers which scale and shift features or feature maps are determined by a separate network, the so-called *FiLM generator*, which predicts these parameters using the external conditioning input. Within various architectures, FiLM has outperformed prior state-of-art for visual question-answering [38,44], multimodal translation [7], and language-guided image segmentation [40].

However, the best way to design the FiLM generator is still an open question. For visual question-answering and visual reasoning, prior work uses single-hop FiLM generators that predict all FiLM parameters at once [38,44]. That is, a Recurrent Neural Network (RNN) sequentially processes input language tokens and then outputs all FiLM parameters via a Multi-Layer Perceptron (MLP). In this paper, we argue that using a *Multi-hop FiLM Generator* is better suited for tasks involving longer input sequences and multi-step reasoning such as dialogue. Even for shorter input sequence tasks, single-hop FiLM generators can require a large RNN to achieve strong performance; on the CLEVR visual reasoning task [21] which only involves a small vocabulary and templated questions, the FiLM generator in [38] uses an RNN with 4096 hidden units that comprises

almost 90% of the model's parameters. Models with Multi-hop FiLM Generators may thus be easier to scale to more difficult tasks involving human-generated language involving larger vocabularies and more ambiguity.

As an intuitive example, consider the dialogue in Fig. 1 through which one speaker localizes the second girl in the image, the one who does not "have a blue frisbee". For this task, a single-hop model must determine upfront what steps of reasoning to carry out over the image and in what order; thus, it might decide in a single shot to highlight feature maps throughout the visual network detecting either non-blue colors or girls. In contrast, a multi-hop model may first determine the most immediate step of reasoning necessary (*i.e.*, locate the girls), highlight the relevant visual features, and then determine the next immediate step of reasoning necessary (*i.e.*, locate the blue frisbee), and so on. While it may be appropriate to reason in either way, the latter approach may scale better to longer language inputs and/or to ambiguous images where the full sequence of reasoning steps is hard to determine upfront, which can even be further enhanced by having intermediate feedback while processing the image.

In this paper, we therefore explore several approaches to generating FiLM parameters in multiple hops. These approaches introduce an intermediate context embedding that controls the language and visual processing, and they alternate between updating the context embedding via an attention mechanism over the language sequence (and optionally by incorporating image activations) and predicting the FiLM parameters. We evaluate our approach on ReferIt [23] and GuessWhat?! [6], two vision-and-language tasks illustrated in Fig. 1. We show that Multi-hop FiLM generation significantly outperforms single-hop FiLM models and prior state-of-the-art. For GuessWhat?!, our best model only updates the context embedding using the language input, while for ReferIt, incorporating visual feedback to update the context embedding improves performance.

In summary, this paper makes the following contributions:

- We introduce the Multi-hop FiLM architecture and demonstrate that our approach significantly improves or matches the state-of-the-art on the GuessWhat?! Oracle task, GuessWhat?! Guesser task and ReferIt Guesser task.
- We show that the Multi-hop FiLM architecture outperforms single-hop models on vision-and-language tasks involving complex visual reasoning.
- We find that including visual feedback into the context embedding of the Multi-hop FiLM Generator is helpful for tasks that do not include object category labels, such as ReferIt.

2 Background

In this section, we explain the prerequisites to understanding our model: RNNs, attention mechanisms, and FiLM. We subsequently use these building blocks to propose a Multi-hop FiLM model.

2.1 Language Processing

One common approach in natural language processing is to use an RNN to encode some linguistic input sequence l into a fixed-size embedding. The input (such as a question or dialogue) consists of a sequence of words $\omega_{1:T}$ of length T, where each word ω_t is contained within a predefined vocabulary $\mathcal{V}$. We embed each input token via a learned look-up table e and obtain a dense word-embedding $\boldsymbol{e}_{\omega_t} = e(\omega_t)$. The sequence of embeddings $\{\boldsymbol{e}_{\omega_t}\}_{t=1}^{T}$ is then fed to a RNN, which produces a sequence of hidden states $\{\boldsymbol{s}_t\}_{t=1}^{T}$ by repeatedly applying a transition function f: $\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{e}_{\omega_t})$ To better handle long-term dependencies in the input sequence, we use a Gated Recurrent Unit (GRU) [4] with layer normalization [2] as transition function. In this work, we use a bidirectional GRU, which consists of one forward GRU, producing hidden states $\overrightarrow{\boldsymbol{s}_t}$ by running from ω_1 to ω_T, and a second backward GRU, producing states $\overleftarrow{\boldsymbol{s}_t}$ by running from ω_T to ω_1. We concatenate both unidirectional GRU states $\boldsymbol{s}_t = [\overrightarrow{\boldsymbol{s}_t}; \overleftarrow{\boldsymbol{s}_t}]$ at each step t to get a final GRU state, which we then use as the compressed embedding $\boldsymbol{e}_l$ of the linguistic sequence l.

2.2 Attention Mechanism

The form of attention we consider was first introduced in the context of machine translation [3,33]. This mechanism takes a weighted average of the hidden states of an encoding RNN based on their relevance to a decoding RNN at various decoding time steps. Subsequent *spatial* attention mechanisms have extended the original mechanism to image captioning [48] and other vision-and-language tasks [24,47]. More formally, given an arbitrary linguistic embedding $\boldsymbol{e}_l$ and image activations $\boldsymbol{F}_{w,h,c}$ where w, h, c are the width, height, and channel indices, respectively, of the image features $\boldsymbol{F}$ at one layer, we obtain a final visual embedding $\boldsymbol{e}_v$ as follows:

$$\xi_{w,h} = MLP(g(\boldsymbol{F}_{\boldsymbol{w},\boldsymbol{h},.}, \boldsymbol{e}_{\boldsymbol{l}})); \quad \alpha_{w,h} = \frac{\exp(\xi_{w,h})}{\sum_{w',h'} \exp(\xi_{w',h'})}; \; \boldsymbol{e}_v = \sum_{w,h} \alpha_{w,h} \boldsymbol{F}_{\boldsymbol{w},\boldsymbol{h},.}, \tag{1}$$

where MLP is a multi-layer perceptron and $g(.,.)$ is an arbitrary fusion mechanism (concatenation, element-wise product, etc.). We will use Multi-modal Low-rank Bilinear (MLB) attention [24] which defines $g(.,.)$ as:

$$g(\boldsymbol{F}_{w,h,\cdot}, \boldsymbol{e}_{\boldsymbol{l}}) = \tanh(\boldsymbol{U}^T \boldsymbol{F}_{w,h,\cdot}) \circ \tanh(\boldsymbol{V}^T \boldsymbol{e}_{\boldsymbol{l}}), \tag{2}$$

where $\circ$ denotes an element-wise product and where $\boldsymbol{U}$ and $\boldsymbol{V}$ are trainable weight matrices. We choose MLB attention because it is parameter efficient and has shown strong empirical performance [22,24].

2.3 Feature-Wise Linear Modulation

Feature-wise Linear Modulation was introduced in the context of image stylization [8] and extended and shown to be highly effective for multi-modal tasks such as visual question-answering [7,38,44].

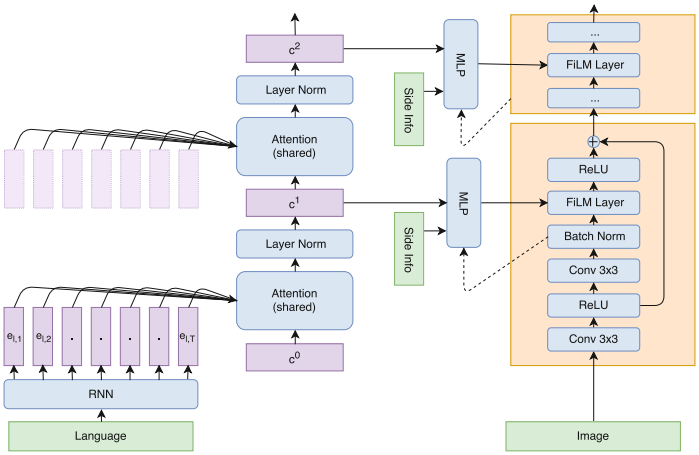

Fig. 2. Overview of the Multi-hop FiLM architecture for applying FiLM. Inputs, Layers, and activation are respectively colored in green, blue and purple. Note the initial FiLM architecture directly uses the $\boldsymbol{e}_{l,T}$ to predict the FiLM parameters. (Color figure online)

A Feature-wise Linear Modulation (FiLM) layer applies a per-channel scaling and shifting to the convolutional feature maps. Such layers are parameter efficient (only two scalars per feature map) while still retaining high capacity, as they are able to scale up or down, zero-out, or negate whole feature maps. In vision-and-language tasks, another network, the so-called FiLM generator h, predicts these modulating parameters from the linguistic input $\boldsymbol{e}_l$. More formally, a FiLM layer computes a modulated feature map $\hat{\boldsymbol{F}}_{w,h,c}$ as follows:

$$[\,\boldsymbol{\gamma}\,;\,\boldsymbol{\beta}\,] = h(\boldsymbol{e}_l); \qquad \hat{\boldsymbol{F}}_{.,.,c} = \gamma_c \boldsymbol{F}_{.,.,c} + \beta_c, \tag{3}$$

where $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are the scaling and shifting parameters which modulate the activations of the original feature map $\boldsymbol{F}_{.,.,c}$. We will use the superscript $k \in [1; K]$ to refer to the k^{th} FiLM layer in the network.

FiLM layers may be inserted throughout the hierarchy of a convolutional network, either pre-trained and fixed [6] or trained from scratch [38]. Prior FiLM-based models [7,38,44] have used a single-hop FiLM generator to predict the FiLM parameters in all layers, *e.g.* an MLP which takes the language embedding $\boldsymbol{e}_l$ as input [7,38,44].

3 Multi-hop FiLM Architecture

In this section, we introduce the Multi-hop FiLM architecture (shown in Fig. 2) to predict the parameters of FiLM layers in an iterative fashion, to better scale to longer input sequences such as in dialogue. Another motivation was to better

disantangle the linguistic reasoning from the visual one by iteratively attending to both pipelines.

We introduce a context vector $\boldsymbol{c}^k$ that acts as a controller for the linguistic and visual pipelines. We initialize the context vector with the final state of a bidirectional RNN $\boldsymbol{s}_T$ and repeat the following procedure for each of the FiLM layers in sequence (from lowest to highest convolutional layer): first, the context vector is updated by performing attention over RNN states (extracting relevant language information), and second, the context is used to predict a layer's FiLM parameters (dynamically modulating the visual information). Thus, the context vector enables the model to perform multi-hop reasoning over the linguistic pipeline while iteratively modulating the image features. More formally, the context vector is computed as follows:

$$\begin{cases} \boldsymbol{c}^0 = \boldsymbol{s}_T \\ \boldsymbol{c}^k = \sum_t \kappa_t^k(\boldsymbol{c}^{k-1}, \boldsymbol{s}_t)\boldsymbol{s}_t, \end{cases} \tag{4}$$

where:

$$\kappa_t^k(\boldsymbol{c}^{k-1}, \boldsymbol{s}_t) = \frac{\exp(\chi_t^k)}{\sum_t \exp(\chi_t^k)}; \qquad \chi_t^k(\boldsymbol{c}^{k-1}, \boldsymbol{s}_t) = MLP_{Attn}(g'(\boldsymbol{c}^k, \boldsymbol{s}_t)), \tag{5}$$

where the dependence of χ_t^k on $(\boldsymbol{c}^{k-1}, \boldsymbol{s}_t)$ may be omitted to simplify notation. MLP_{Attn} is a network (shared across layers) which aids in producing attention weights. g' can be any fusion mechanism that facilitates selecting the relevant context to attend to; here we use a simple dot-product following [33], thus $g'(\boldsymbol{c}^k, \boldsymbol{s}_t) = \boldsymbol{c}^k \circ \boldsymbol{s}_t$. Finally, FiLM is carried out using a layer-dependent neural network MLP_{FiLM}^k:

$$[\,\boldsymbol{\gamma}^k\,;\,\boldsymbol{\beta}^k\,] = MLP_{FiLM}^k(\boldsymbol{c}^k); \qquad \hat{\boldsymbol{F}}_{w,h,c}^k = \gamma_c^k \boldsymbol{F}_{\cdot,\cdot,c}^k + \beta_c^k. \tag{6}$$

As a regularization, we append a normalization-layer [2] on top of the context vector after each attention step.

External Information. Some tasks provide additional information which may be used to further improve the visual modulation. For instance, GuessWhat?! provides spatial features of the ground truth object to models which must answer questions about that object. Our model incorporates such features by concatenating them to the context vector before generating FiLM parameters.

Visual Feedback. Inspired by the co-attention mechanism [31,54], we also explore incorporating visual feedback into the Multi-hop FiLM architecture. To do so, we first extract the image or crop features $\boldsymbol{F}^k$ (immediately before modulation) and apply a global mean-pooling over spatial dimensions. We then concatenate this visual state into the context vector $\boldsymbol{c}^k$ before generating the next set of FiLM parameters.

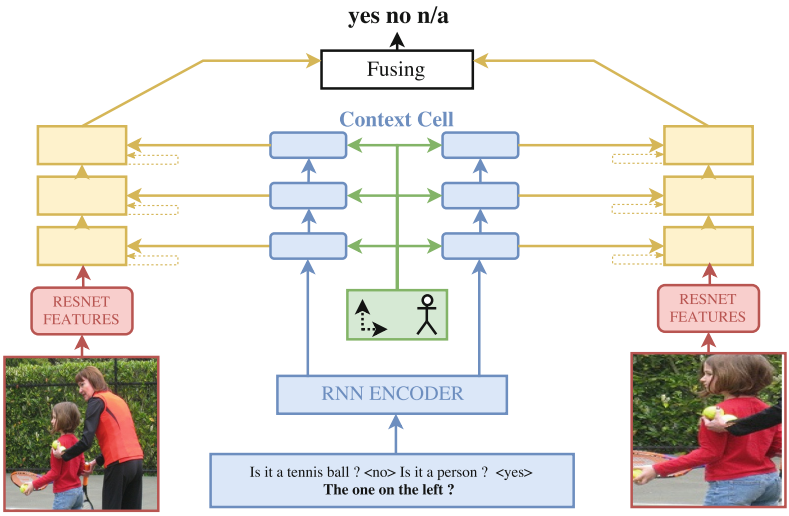

Fig. 3. Overall model. Consists of a visual pipeline (red and yellow) and linguistic pipeline (blue) and incorporates any additional contextual information (green). (Color figure online)

4 Experiments

In this section, we first introduce the ReferIt and GuessWhat?! datasets and respective tasks and then describe our overall Multi-hop FiLM architecture (Fig. 3)[1].

4.1 Dataset

ReferIt [23,51] is a cooperative two-player game. The first player (the Oracle) selects an object in a rich visual scene, for which they must generate an expression that refers to it (*e.g.* "the person eating an ice-cream"). the second player (the Guesser) must then select an object within the image. There are four ReferIt datasets exist: RefClef, RefCOCO, RefCOCO+ and RefCOCOg. The first dataset contains 130K references over 20K images from the ImageClef dataset [35], while the three other datasets respectively contain 142K, 142K and 86K references over 20K, 20k and 27K images from the MSCOCO dataset [29]. Each dataset has small differences. RefCOCO and RefClef were constructed using different image sets. RefCOCO+ forbids certain words to prevent object references from being too simplistic, and RefCOCOg only relies on images containing 2–4 objects from the same category. RefCOCOg also contains longer and more complex sentences than RefCOCO (8.4 vs. 3.5 average words). Here, we will show results on both the Guesser and Oracle tasks.

[1] The code and hyperparameters are available at https://github.com/GuessWhatGame.

GuessWhat?! [6] is a cooperative three-agent game in which players see the picture of a rich visual scene with several objects. One player (the Oracle) is randomly assigned an object in the scene. The second player (Questioner) aims to ask a series of yes-no questions to the Oracle to collect enough evidence to allow the third player (Guesser) to correctly locate the object in the image. The GuessWhat?! dataset is composed of 131K successful natural language dialogues containing 650k question-answer pairs on over 63K images from MSCOCO [29]. Dialogues contain 5.2 question-answer pairs and 34.4 words on average. Here, we will focus on the Guesser and Oracle tasks.

4.2 Task Descriptions

Game Features. Both games consist of triplets $(\mathcal{I}, l, o)$, where $\mathcal{I} \in \mathbb{R}^{3\times M\times N}$ is an RGB image and l is some language input (i.e. a series of words) describing an object o in $\mathcal{I}$. The object o is defined by an object category, a pixel-wise segmentation, an RGB crop of $\mathcal{I}$ based on bounding box information, and hand-crafted spatial information $\boldsymbol{x}_{spatial}$, where

$$\boldsymbol{x}_{spatial} = [x_{min}, y_{min}, x_{max}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}] \tag{7}$$

We replace words with two or fewer occurrences with an <*unk*> token.

The Oracle Task. Given an image $\mathcal{I}$, an object o, a question q, and a sequence δ of previous question-answer pairs $(\boldsymbol{q}, a)_{1:\delta}$ where $a \in \{\text{Yes}, \text{No}, \text{N/A}\}$, the oracle's task is to produce an answer a that correctly answers the question q. In our experiments, we will use the symbol (D) when the previous question-answer pairs are concatenated with the question q to obtain a single sequence of tokens s. Similarly, we will use the symbol (Q) when dropping the previous question-answers. The oracle is trained using cross-entropy loss.

The Guesser Task. Given an image $\mathcal{I}$, a list of objects $O = o_{1:\Phi}$, a target object $o^* \in O$ and the dialogue $\mathcal{D}$, the guesser needs to output a probability σ_ϕ that each object o_ϕ is the target object o^*. Following [17], the Guesser is evaluated by selecting the object with the highest probability of being correct. Note that even if the individual probabilities σ_ϕ are between 0 and 1, their sum can be greater than 1. More formally, the Guesser loss and error are computed as follows:

$$L_{Guesser} = \frac{-1}{N_{games}} \sum_{n}^{N_{games}} \frac{1}{\Phi^n} \sum_{\phi}^{\Phi} \log(p(o^*|\mathcal{I}^n, o^n_\phi, \mathcal{D}^n)) \tag{8}$$

$$E_{Guesser} = \frac{-1}{N_{games}} \sum_{n}^{N_{games}} \mathbb{1}(o^* \neq o_{\text{argmax}_\phi\, \sigma^n_\phi}) \tag{9}$$

where $\mathbb{1}$ is the indicator function and Φ^n the number of objects in the n^{th} game.

4.3 Model

We use similar models for both ReferIt and GuessWhat?! and provide its architectural details in this subsection.

Object Embedding. The object category is fed into a dense look-up table $\boldsymbol{e}_{cat}$, and the spatial information is scaled to $[-1; 1]$ before being up-sampled via non-linear projection to $\boldsymbol{e}_{spat}$. We do not use the object category in ReferIt models.

Visual Pipeline. We first resized the image and object crop to 448×448 before extracting $14 \times 14 \times 1024$ dimensional features from a ResNet-152 [15] (block3) pre-trained on ImageNet [41]. Following [38], we feed these features to a 3×3 convolution layer with Batch Normalization [19] and Rectified Linear Unit [37] (ReLU). We then stack four modulated residual blocks (shown in Fig. 2), each producing a set of feature maps $\boldsymbol{F}^k$ via (in order) a 1×1 convolutional layer (128 units), ReLU activations, a 3×3 convolutional layer (128 units), and an untrainable Batch Normalization layer. The residual block then modulates $\boldsymbol{F}^k$ with a FiLM layer to get $\hat{\boldsymbol{F}}^k$, before again applying ReLU activations. Lastly, a residual connection sums the activations of both ReLU outputs. After the last residual block, we use a 1×1 convolution layer (512 units) with Batch Normalization and ReLU followed by MLB attention [24] (256 units and 1 glimpse) to obtain the final embedding $\boldsymbol{e}_v$. Note our model uses two independent visual pipeline modules: one to extract modulated image features $\boldsymbol{e}_v^{img}$, one to extract modulated crop features $\boldsymbol{e}_v^{crop}$.

To incorporate spatial information, we concatenate two coordinate feature maps indicating relative x and y spatial position (scaled to $[-1, 1]$) with the image features before each convolution layer (except for convolutional layers followed by FiLM layers). In addition, the pixel-wise segmentations $S \in \{0, 1\}^{M \times N}$ are rescaled to 14×14 floating point masks before being concatenated to the feature maps.

Linguistic Pipeline. We compute the language embedding by using a word-embedding look-up (200 dimensions) with dropout followed by a Bi-GRU (512×2units) with Layer Normalization [2]. As described in Sect. 3, we initialize the context vector with the last RNN state $\boldsymbol{c}^0 = \boldsymbol{s}_T$. We then attend to the other Bi-GRU states via an attention mechanism with a linear projection and ReLU activations and regularize the new context vector with Layer Normalization.

FiLM Parameter Generation. We concatenate spatial information $\boldsymbol{e}_{spat}$ and object category information $\boldsymbol{e}_{cat}$ to the context vector. In some experiments, we also concatenate a fourth embedding consisting of intermediate visual features $\boldsymbol{F}^k$ after mean-pooling. Finally, we use a linear projection to map the embedding to FiLM parameters.

Final Layers. We first generate our final embedding by concatenating the output of the visual pipelines $\boldsymbol{e}_{final} = [\boldsymbol{e}_v^{img}; \boldsymbol{e}_v^{crop}]$ before applying a linear projection (512 units) with ReLU and a softmax layer.

Training Process. We train our model end-to-end with Adam [26] (learning rate 3.10^{-4}), a dropout ratio of 0.5, weight decay of 5.10^{-6} for convolutional network

Table 1. ReferIt Guesser test error.

Referit Split by	RefCOCO (unc)			RefCOCO+ (unc)			RefCOCOg (google)
Report on	Valid	TestA	TestB	Valid	TestA	TestB	Val
MMI [36]	-	71.7%	71.1%	-	58.4%	51.2%	59.3%
visDif + MMI [51]	-	74.6%	76.6%	-	59.2%	55.6%	64.0%
NEG Bag [36]	-	75.6%	78.0%	-	-	-	68.4%
Joint-SLR [52]	78.9%	78.0%	80.7%	61.9%	64.0%	59.2%	-
PLAN [54]	81.7%	80.8%	81.3%	64.2%	66.3%	61.5%	69.5%
MAttN [50]	**85.7**%	85.3%	**84.6**%	71.0%	75.1%	**66.2**%	-
Baseline + MLB	77.6%	79.6%	77.2%	60.8%	59.7%	66.2%	63.1%
Single-hop FiLM	83.4%	85.8%	80.9%	72.1%	77.3%	63.9%	67.8%
Multi-hop FiLM	83.5%	86.5%	81.3%	73.4%	77.7%	64.5%	69.8%
Multi-hop FiLM (+img)	84.9%	**87.4**%	83.1%	**73.8**%	**78.7**%	65.8%	**71.5**%

layers, and a batch size of 64. We report results after early stopping on the validation set with a maximum of 15 epochs.

4.4 Baseline Models

In our experiments, we re-implement several baseline models to benchmark the performance of our models. The standard *Baseline* is a straightforward concatenation of the image and object crop features after mean pooling, the linguistic embedding, and the spatial embedding and the category embedding (GuessWhat?! only), followed by the same final layers described in our proposed model. We refer to a model which uses the MLB attention mechanism to pool the visual features as *Baseline+MLB*. We also implement a *Single-hop FiLM* mechanism which is equivalent to setting all context vectors equal to the last state of the Bi-GRU $\boldsymbol{e}_{l,T}$. Finally, we experiment with injecting intermediate visual features into the FiLM Generator input, and we refer to the model as *Multi-hop FiLM (+img)*.

4.5 Results

ReferIt Guesser. We report the best test error of the outlined methods on the ReferIt Guesser task in Table 1. Note that RefCOCO and RefCOCO+ split test sets into TestA and TestB, only including expression referring towards people and objects, respectively. We do not report [50] and [52] scores on RefCOCOg as the authors use a different split (umd). Our initial baseline achieves 77.6%, 60.8%, 63.1%, 73.4% on the RefCOCO, RefCOCO+, RefCOCOg, RefClef datasets, respectively, performing comparably to state-of-the-art models. We observe a significant improvements using a FiLM-based architecture, jumping to 84.9%, 87.4%, 73.8%, 71.5%, respectively, and outperforming most prior methods and achieving comparably performance with the concurrent MAttN [50] model. Interestingly, MAttN and Multi-hop FiLM are built in two different manners; while

Table 2. GuessWhat?! Oracle test error.

Oracle models	Quest.	Dial.	Object	Image	Crop	Test error
Dominant class (no)	✗	✗	✗	✗	✗	50.9%
Question only [6]	✓	✗	✗	✗	✗	41.2%
Image only [6]	✗	✗	✗	✓	✗	46.7%
Crop only [6]	✗	✗	✗	✗	✓	43.0%
No-Vision (Q) [6]	✓	✗	✓	✗	✗	21.5%
No-Vision (D)	✗	✓	✓	✗	✗	20.6%
Baseline (Q)	✓	✗	✓	✓	✓	23.3%
Baseline (D)	✗	✓	✓	✓	✓	22.4%
Baseline + MLB (Q)	✓	✗	✓	✓	✓	21.8%
Baseline + MLB (D)	✗	✓	✓	✓	✓	21.1%
MODERN [44]	✓	✗	✓	✗	✓	19.5%
Single-hop FiLM (Q)	✓	✗	✓	✓	✓	17.8%
Single-hop FiLM (D)	✗	✓	✓	✓	✓	17.6%
Multi-hop FiLM	✗	✓	✓	✓	✓	**16.9%**
Multi-hop FiLM (+img)	✗	✓	✓	✓	✓	17.1%

the former has three specialized reasoning blocks, our model uses a generic feature modulation approach. These architectural differences surface when examining test splits: MAttN achieves excellent results on referring expression towards objects while Multi-hop FiLM performs better on referring expressions towards people.

GuessWhat?! Oracle. We report the best test error of several variants of GuessWhat?! Oracle models in Table 2. First, we baseline any visual or language biases by predicting the Oracle's target answer using only the image (46.7% error) or the question (41.1% error). As first reported in [6], we observe that the baseline methods perform worse when integrating the image and crop inputs (21.1%) rather than solely using the object category and spatial location (20.6%). On the other hand, concatenating previous question-answer pairs to answer the current question is beneficial in our experiments. Finally, using Single-hop FiLM reduces the error to 17.6% and Multi-hop FiLM further to 16.9%, outperforming the previous best model by 2.4%.

GuessWhat?! Guesser. We provide the best test error of the outlined methods on the GuessWhat?! Guesser task in Table 3. As a baseline, we find that random object selection achieves an error rate of 82.9%. Our initial model baseline performs significantly worse (38.3%) than concurrent models (36.6%), highlighting that successfully jointly integrating crop and image features is far from trivial. However, Single-hop FiLM manages to lower the error to 35.6%. Finally, Multi-hop FiLM architecture outperforms other models with a final error of 30.5%.

Table 3. GuessWhat?! Guesser test error.

Guesser Error	Test Error
Random	82.9%
LSTM [6]	38.7%
LSTM + Img [6]	39.5%
PLAN [54]	36.6%
MLB-Baseline (crop)	38.3%
Single-hop FiLM	35.6%
Multi-hop FiLM	**30.5%**

Guesser Error	Crop	Image	Crop/Img
Baseline	38.3%	40.0%	45.1%
Single-hop FiLM	35.3%	35.7%	35.6%
Multi-hop FiLM	32.3%	35.0%	**30.5%**
Multi-hop FiLM (no cat.)	33.1%	40%	33.4%

5 Discussion

Single-hop FiLM vs. Multi-hop FiLM. In the GuessWhat?! task, Multi-hop FiLM outperforms Single-hop FiLM by 6.1% on the Guesser task but only 0.7% on the Oracle task. We think that the small performance gain for the Oracle task is due to the nature of the task; to answer the current question, it is often not necessary to look at previous question-answer pairs, and in most cases this task does not require a long chain of reasoning. On the other hand, the Guesser task needs to gather information across the whole dialogue in order to correctly retrieve the object, and it is therefore more likely to benefit from multi-hop reasoning. The same trend can be observed for ReferIt. Single-hop FiLM and Multi-hop FiLM perform similarly on RefClef and RefCOCO, while we observe 1.3% and 2% gains on RefCOCO+ and RefCOCOg, respectively. This pattern of performance is intuitive, as the former datasets consist of shorter referring expressions (3.5 average words) than the latter (8.4 average words in RefCOCOg), and the latter datasets also consist of richer, more complex referring expressions due *e.g.* to taboo words (RefCOCO+). In short, our experiments demonstrate that Multi-hop FiLM is better able reason over complex linguistic sequences.

Reasoning Mechanism. We conduct several experiments to better understand our method. First, we assess whether Multi-hop FiLM performs better because of increased network capacity. We remove the attention mechanism over the linguistic sequence and update the context vector via a shared MLP. We observe that this change significantly hurts performance across all tasks, *e.g.*, increasing the Multi-hop FiLM error of the Guesser from 30.5 to 37.3%. Second, we investigate how the model attends to GuessWhat?! dialogues for the Oracle and Guesser tasks, providing more insight into how to the model reasons over the language input. We first look at the top activation in the (crop) attention layers to observe where the most prominent information is. Note that similar trends are observed for the image pipeline. As one would expect, the Oracle is focused on a specific word in the last question 99.5% of the time, one which is crucial to answer the question at hand. However, this ratio drops to 65% in the Guesser task, suggesting the model is reasoning in a different way. If we then extract the top 3 activations per layer, the attention points to $<yes>$ or $<no>$ tokens (respectively) at least once, 50% of the time for the Oracle and Guesser, showing that the attention is able to correctly split the dialogue into question-answer

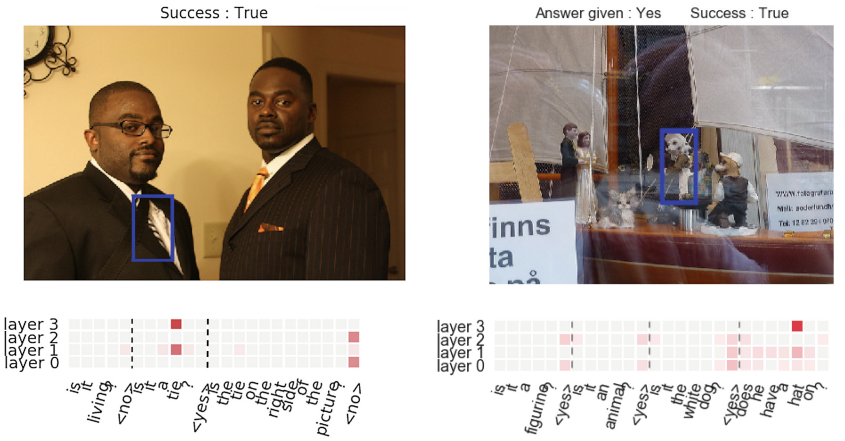

Fig. 4. Guesser and Oracle attention mechanism in the crop visual pipeline.

pairs. Finally, we plot the attention masks for each FiLM layer to have a better intuition of this reasoning process in Fig. 4.

Crop vs. Image. We also evaluate the impact of using the image and/or crop on the final error for the Guesser task Table 3. Using the image alone (while still including object category and spatial information) performs worse than using the crop. However, using image and crop together inarguably gives the lowest errors, though prior work has not always used the crop due to architecture-specific GPU limitations [44].

Visual Feedback. We explore whether adding visual feedback to the context embedding improves performance. While it has little effect on the GuessWhat?! Oracle and Guesser tasks, it improves the accuracy on ReferIt by 1–2%. Note that ReferIt does not include class labels of the selected object, so the visual feedback might act as a surrogate for this information. To further investigate this hypothesis, we remove the object category from the GuessWhat?! task and report results in Table 5 in the supplementary material. In this setup, we indeed observe a relative improvement 0.4% on the Oracle task, further confirming this hypothesis.

Pointing Tasks. In GuessWhat?!, the Guesser must select an object among a list of items. For the task to be natural, the system should directly point out the object as a human would might. Thus, we provide an initial baseline that scores up to 84.0% error in the supplementary material in Table 7.

6 Related Work

The ReferIt game [23] has been a testbed for various vision-and-language tasks over the past years, including object retrieval [32,36,50–52,54], semantic image

segmentation [16,39], and generating referring descriptions [32,51,52]. To tackle object retrieval, [36,50,51] extract additional visual features such as relative object locations and [32,52] use reinforcement learning to iteratively train the object retrieval and description generation models. Closer to our work, [17,54] use the full image and the object crop to locate the correct object. While some previous work relies on task-specific modules [50,51], our approach is general and can be easily extended to other vision-and-language tasks.

The GuessWhat?! game [6] can be seen as a dialogue version of the ReferIt game, one which additionally draws on visual question answering ability. [28,42,53] make headway on the dialogue generation task via reinforcement learning. However, these approaches are bottlenecked by the accuracy of Oracle and Guesser models, despite existing modeling advances [44,54]; accurate Oracle and Guesser models are crucial for providing a meaningful learning signal for dialogue generation models, so we believe the Multi-hop FiLM architecture will facilitate high quality dialogue generation as well.

A special case of Feature-wise Linear Modulation was first successfully applied to image style transfer [8], whose approach modulates image features according to some image style (*i.e.*, cubism or impressionism). [44] extended this approach to vision-and-language tasks, injecting FiLM-like layers along the entire visual pipeline of a pre-trained ResNet. [38] demonstrates that a convolutional network with FiLM layers achieves strong performance on CLEVR [21], a task that focuses on answering reasoning-oriented, multi-step questions about synthetic images. Subsequent work has demonstrated that FiLM and variants thereof are effective for video object segmentation where the conditioning input is the first image's segmentation (instead of language) [49] and language-guided image segmentation [40]. Even more broadly, [9] overviews the strength of FiLM-related methods across machine learning domains, ranging from reinforcement learning to generative modeling to domain adaptation.

There are other notable models that decompose reasoning into different modules. For instance, Neural Turing Machines [13,14] divide a model into a controller with read and write units. Memory networks use an attention mechanism to answer a query by reasoning over a linguistic knowledge base [43,45] or image features [46]. A memory network updates a query vector by performing several attention hops over the memory before outputting a final answer from this query vector. Although Multi-hop FiLM computes a similar context vector, this intermediate embedding is used to predict FiLM parameters rather than the final answer. Thus, Multi-hop FiLM includes a second reasoning step over the image.

Closer to our work, [18] designed networks composed of Memory, Attention, and Control (MAC) cells to perform visual reasoning. Similar to Neural Turing Machines, each MAC cell is composed of a control unit that attends over the language input, a read unit that attends over the image and a write unit that fuses both pipelines. Though conceptually similar to Multi-hop FiLM models, Compositional Attention Networks differ structurally, for instance using a dynamic neural architecture and relying on spatial attention rather than FiLM.

7 Conclusion

In this paper, we introduce a new way to exploit Feature-wise Linear Modulation (FiLM) layers for vision-and-language tasks. Our approach generates the parameters of FiLM layers going up the visual pipeline by attending to the language input in multiple hops rather than all at once. We show Multi-hop FiLM Generator architectures are better able to handle longer sequences than their single-hop counterparts. We outperform state-of-the-art vision-and-language models with significant performance gains on the ReferIt object retrieval task and GuessWhat?! visual dialogue task. Finally, we believe that this Multi-hop FiLM Generator approach is generic and can extended to a variety of vision-and-language tasks, particularly those requiring complex visual reasoning.

Acknowledgements. The authors would like to acknowledge the stimulating research environment of the SequeL Team. We also thank Vincent Dumoulin for helpful discussions. We acknowledge the following agencies for research funding and computing support: CHISTERA IGLU and CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015–2020, NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR.

Additional Results

ReferIt ImageClef

See Table 4.

Table 4. ReferIt Guesser test error.

Referit	RefClef (berkeley) test
SCRC [17]	72.7%
Baseline + MLB	74.6%
Single-hop FiLM	84.0%
Multi-hop FiLM	84.3%
Multi-hop FiLM +(img)	**85.1**%

Category-Less Oracle

Table 5. GuessWhat?! Oracle test error without category.

Oracle models	Quest.	Dial.	Spat.	Image	Crop	Test error
Baseline + MLB	✗	✓	✓	✓	✓	26.7%
Single-hop FiLM	✗	✓	✓	✓	✓	19.5%
Multi-hop FiLM	✗	✓	✓	✓	✓	18.9%
Multi-hop FiLM (+img loop)	✗	✓	✓	✓	✓	**18.4%**

Category-Less Guesser

See Table 6.

Table 6. GuessWhat?! Guesser test error without category.

Guesser error	Crop	Image	Crop/Img
PLAN [54]	-	-	40.3%
Multi-hop FiLM	35.3%	39.8%	33.9%
Multi-hop FiLM (+img)	34.3%	40.1%	**33.2%**

Guesser Pointing

Table 7. Guesser pointing errors for different IoU thresholds.

Guesser model	IoU > 0.3	IoU >0.5	IoU > 0.7
Baseline	81.4%	92.0%	98.2%
FiLM	74.0%	85.9%	94.7%
Multi-hop FiLM	73.4%	84.6%	93.7%
Multi-hop FiLM (+img)	**71.9%**	**84.0%**	**93.6%**

So far, the guesser has selected its answer among a provided list of objects. A more natural task would be for the guesser to directly point out the object as a human might. Thus, we introduce such a pointing task as a new benchmark for GuessWhat?!. This task is to locate the intended object based on a series of questions and answers, but instead of selecting the object from a list, the guesser must output a bounding box around the object of its guess. This box is defined as the 4-tuple (x, y, width, height), where (x, y) is the coordinate of the top left corner of the box, within the original image $\mathcal{I}$, given an input dialogue. Note that this new task is more difficult, as the model does not have access to the list of objects. The original task also includes important side information, namely object category and (x, y)-position [6] which ease the object retrieval.

We assess bounding box accuracy using the Intersection Over Union (IoU) metric: the area of the intersection of predicted and ground truth bounding boxes, divided by the area of their union. In prior literature [10,12], an object is usually considered found if the IoU is greater than 0.5.

$$\textbf{IoU} = \frac{|\text{bboxA} \cap \text{bboxB}|}{|\text{bboxA} \cup \text{bboxB}|} = \frac{|\text{bboxA} \cap \text{bboxB}|}{|\text{bboxA}| + |\text{bboxB}| - |\text{bboxA} \cap \text{bboxB}|} \tag{10}$$

We report model error in Table 7. Interestingly, the baseline obtains 92.0% error while Multi-hop FiLM Generator obtains 84.0% error. As previously discussed, we also note that re-injecting visual features into the Multi-hop FiLM Generator's context cell is also beneficial. The error rates are relatively high, though also in line with those of similar pointing tasks such as SCRC [16,17] (around 90%) on ReferIt.

Attention Mask

See Figs. 5, 6, 7, 8 and 9.

Fig. 5. Oracle attention masks with attention on the last question (crop pipeline)

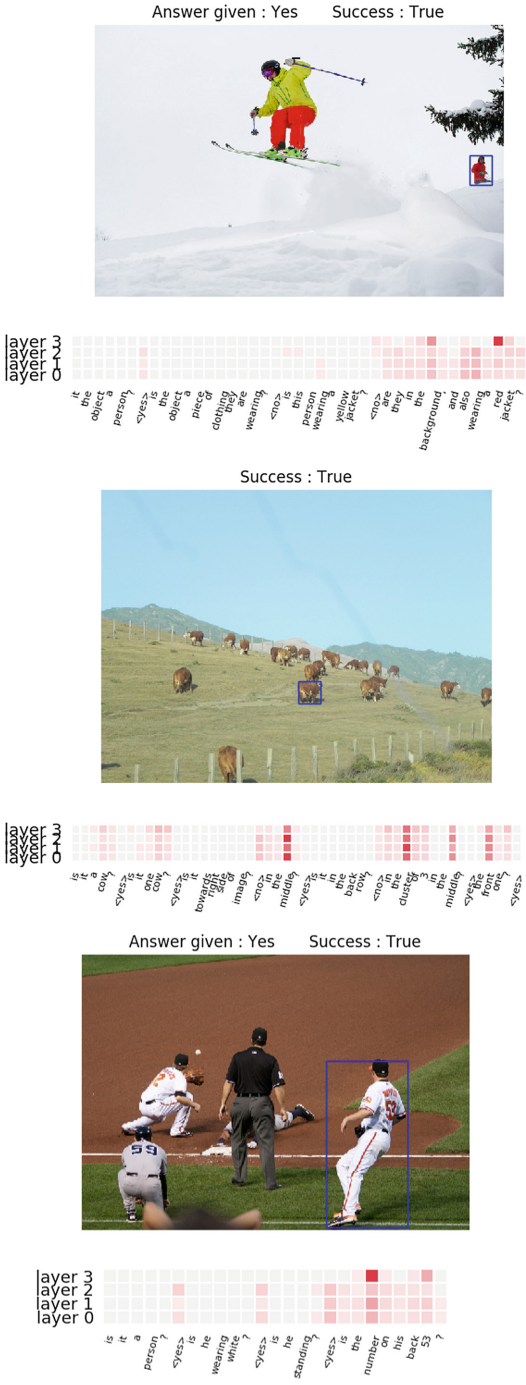

Fig. 6. Oracle attention masks with complex reasoning (crop pipeline)

Fig. 7. Oracle attention masks with negative results (crop pipeline)

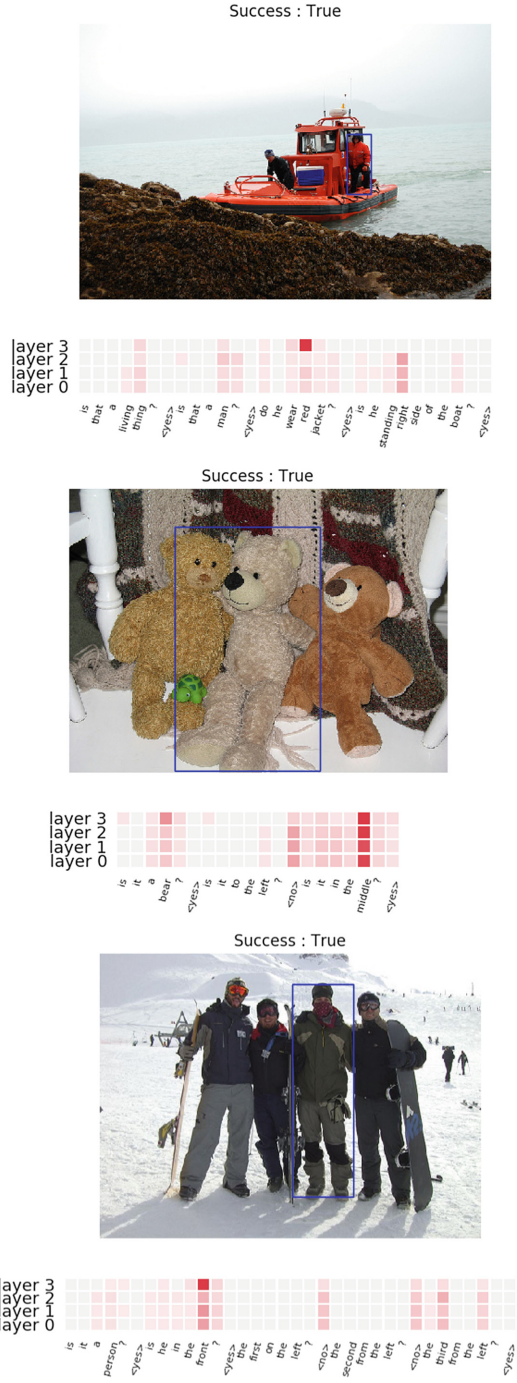

Fig. 8. Guesser attention masks with positive results (crop pipeline)

Fig. 9. Guesser attention masks with negative results (crop pipeline).

References

1. Antol, S., et al.: VQA: visual question answering. In: Proceedings of ICCV (2015)
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. In: Deep Learning Symposium (NIPS) (2016)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of ICLR (2015)
4. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: Proceedings of ICML (2015)
5. Das, A., et al.: Visual dialog. In: Proceedings of CVPR (2017)
6. De Vries, H., Strub, F., Chandar, S., Pietquin, O., Larochelle, H., Courville, A.: Guesswhat?! Visual object discovery through multi-modal dialogue. In: Proceedings of CVPR (2017)
7. Delbrouck, J.B., Dupont, S.: Modulating and attending the source image during encoding improves multimodal translation. In: Visually-Grounded Interaction and Language Workshop (NIPS) (2017)
8. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. In: Proceedings of ICLR (2017)
9. Dumoulin, V., et al.: Feature-wise transformations. Distill (2018). https://doi.org/10.23915/distill.00011, https://distill.pub/2018/feature-wise-transformations
10. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
11. Fukui, A., Park, D.H., Yang, D., Rohrbach, A., Darrell, T., Rohrbach, M.: Multimodal compact bilinear pooling for visual question answering and visual grounding. In: Proceedings of EMNLP (2016)
12. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of of CVPR (2014)
13. Graves, A., Wayne, G., Danihelka, I.: Neural turing machines. arXiv preprint arXiv:1410.5401 (2014)
14. Graves, A., et al.: Hybrid computing using a neural network with dynamic external memory. Nature **538**(7626), 471 (2016)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of CVPR (2016)
16. Hu, R., Rohrbach, M., Darrell, T.: Segmentation from natural language expressions. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 108–124. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_7
17. Hu, R., Xu, H., Rohrbach, M., Feng, J., Saenko, K., Darrell, T.: Natural language object retrieval. In: Proceedings of CVPR (2016)
18. Hudson, D.A., Manning, C.D.: Compositional attention networks for machine reasoning. In: Proceedings of ICL (2018)
19. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of ICML (2015)
20. Jabri, A., Joulin, A., van der Maaten, L.: Revisiting visual question answering baselines. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9912, pp. 727–739. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_44
21. Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C.L., Girshick, R.: CLEVR: a diagnostic dataset for compositional language and elementary visual reasoning. In: Proceedings of CVPR (2017)

22. Kafle, K., Kanan, C.: Visual question answering: datasets, algorithms, and future challenges. Comput. Vis. Image Underst. **163**, 3–20 (2017)
23. Kazemzadeh, S., Ordonez, V., Matten, M., Berg, T.: ReferitGame: referring to objects in photographs of natural scenes. In: Proceedings of EMNLP (2014)
24. Kim, J.H., On, K.W., Lim, W., Kim, J., Ha, J.W., Zhang, B.T.: Hadamard product for low-rank bilinear pooling. In: Proceedings of ICLR (2017)
25. Kim, J.H., et al.: Multimodal residual learning for visual QA. In: Proceedings of NIPS (2016)
26. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of ICLR (2014)
27. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of of NIPS (2012)
28. Lee, S.W., Heo, Y.J., Zhang, B.T.: Answerer in questioner's mind for goal-oriented visual dialogue. In: Visually-Grounded Interaction and Language Workshop (NIPS) (2018)
29. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
30. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of CVPR (2015)
31. Lu, J., Yang, J., Batra, D., Parikh, D.: Hierarchical question-image co-attention for visual question answering. In: Proceedings of NIPS (2016)
32. Luo, R., Shakhnarovich, G.: Comprehension-guided referring expressions. In: Proceedings of CVPR (2017)
33. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of EMNLP (2015)
34. Malinowski, M., Rohrbach, M., Fritz, M.: Ask your neurons: a neural-based approach to answering questions about images. In: Proceedings of ICCV (2015)
35. Mller, H., Clough, P., Deselaers, T., Caputo, B.: ImageCLEF: Experimental Evaluation in Visual Information Retrieval. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-15181-1
36. Nagaraja, V.K., Morariu, V.I., Davis, L.S.: Modeling context between objects for referring expression understanding. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 792–807. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_48
37. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proceedings of ICML (2010)
38. Perez, E., Strub, F., De Vries, H., Dumoulin, V., Courville, A.: Film: visual reasoning with a general conditioning layer. In: Proceedings of AAAI (2018)
39. Rohrbach, A., Rohrbach, M., Hu, R., Darrell, T., Schiele, B.: Grounding of textual phrases in images by reconstruction. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 817–834. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_49
40. Rupprecht, C., Laina, I., Navab, N., Hager, G.D., Tombari, F.: Guide me: interacting with deep networks. In: Proceedings of CVPR (2018)
41. Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
42. Strub, F., De Vries, H., Mary, J., Piot, B., Courville, A., Pietquin, O.: End-to-end optimization of goal-driven and visually grounded dialogue systems. In: Proceedings of IJCAI (2017)

43. Sukhbaatar, S., Weston, J., Fergus, R., et al.: End-to-end memory networks. In: Proceedings of NIPS (2015)
44. de Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., Courville, A.C.: Modulating early visual processing by language. In: Proceedings of NIPS (2017)
45. Weston, J., Chopra, S., Bordes, A.: Memory networks. arXiv preprint arXiv:1410.3916 (2014)
46. Xiong, C., Merity, S., Socher, R.: Dynamic memory networks for visual and textual question answering. In: Proceedings of ICML (2016)
47. Xu, H., Saenko, K.: Ask, attend and answer: exploring question-guided spatial attention for visual question answering. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 451–466. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_28
48. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. In: Proceedings of ICML (2015)
49. Yang, L., Wang, Y., Xiong, X., Yang, J., Katsaggelos, A.K.: Efficient video object segmentation via network modulation. In: Proceedings of CVPR (2018)
50. Yu, L., et al.: MAttNet: modular attention network for referring expression comprehension. In: Proceedings of CVPR (2018)
51. Yu, L., Poirson, P., Yang, S., Berg, A.C., Berg, T.L.: Modeling context in referring expressions. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 69–85. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_5
52. Yu, L., Tan, H., Bansal, M., Berg, T.L.: A joint speakerlistener-reinforcer model for referring expressions. In: Proceedings of CVPR (2016)
53. Zhu, Y., Zhang, S., Metaxas, D.: Reasoning about fine-grained attribute phrases using reference games. In: Visually-Grounded Interaction and Language Workshop (NIPS) (2017)
54. Zhuang, B., Wu, Q., Shen, C., Reid, I.D., van den Hengel, A.: Parallel attention: a unified framework for visual object discovery through dialogs and queries. In: Proceedings of CVPR (2018)