Learning to Compose Neural Networks for Question Answering

Jacob Andreas and **Marcus Rohrbach** and **Trevor Darrell** and **Dan Klein**
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
{jda,rohrbach,trevor,klein}@eecs.berkeley.edu

Abstract

We describe a question answering model that applies to both images and structured knowledge bases. The model uses natural language strings to automatically assemble neural networks from a collection of composable modules. Parameters for these modules are learned jointly with network-assembly parameters via reinforcement learning, with only (world, question, answer) triples as supervision. Our approach, which we term a *dynamic neural module network*, achieves state-of-the-art results on benchmark datasets in both visual and structured domains.

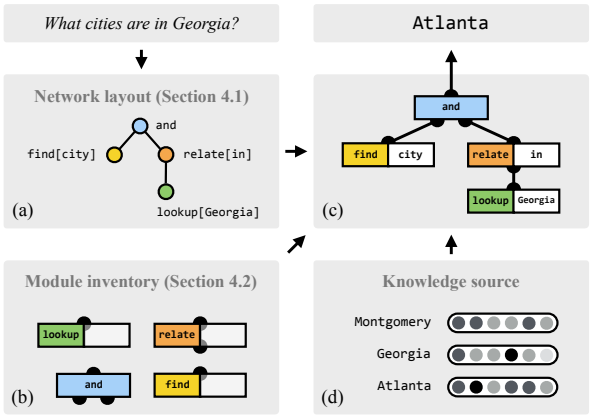

Figure 1: A learned syntactic analysis (a) is used to assemble a collection of neural modules (b) into a deep neural network (c), and applied to a world representation (d) to produce an answer.

1 Introduction

This paper presents a compositional, attentional model for answering questions about a variety of world representations, including images and structured knowledge bases. The model translates from questions to dynamically assembled neural networks, then applies these networks to world representations (images or knowledge bases) to produce answers. We take advantage of two largely independent lines of work: on one hand, an extensive literature on answering questions by mapping from strings to logical representations of meaning; on the other, a series of recent successes in deep neural models for image recognition and captioning. By constructing neural networks instead of logical forms, our model leverages the best aspects of both linguistic compositionality and continuous representations.

Our model has two components, trained jointly: first, a collection of neural "modules" that can be freely composed (Figure 1b); second, a network layout predictor that assembles modules into complete deep networks tailored to each question (Figure 1a). Previous work has used manually-specified modular structures for visual learning (Andreas et al., 2016). Here we:

- *learn* a network structure predictor jointly with module parameters themselves
- *extend* visual primitives from previous work to reason over structured world representations

Training data consists of (world, question, answer) triples: our approach requires no supervision of network layouts. We achieve state-of-the-art performance on two markedly different question answering tasks: one with questions about natural images, and another with more compositional questions about United States geography.[1]

2 Deep networks as functional programs

We begin with a high-level discussion of the kinds of composed networks we would like to learn.

[1]We have released our code at http://github.com/jacobandreas/nmn2

Proceedings of NAACL-HLT 2016, pages 1545–1554,
San Diego, California, June 12-17, 2016.

Andreas et al. (2016) describe a heuristic approach for decomposing visual question answering tasks into sequence of modular sub-problems. For example, the question *What color is the bird?* might be answered in two steps: first, "where is the bird?" (Figure 2a), second, "what color is that part of the image?" (Figure 2c). This first step, a generic **module** called `find`, can be expressed as a fragment of a neural network that maps from image features and a lexical item (here *bird*) to a distribution over pixels. This operation is commonly referred to as the *attention mechanism*, and is a standard tool for manipulating images (Xu et al., 2015) and text representations (Hermann et al., 2015).

The first contribution of this paper is an extension and generalization of this mechanism to enable fully-differentiable reasoning about more structured semantic representations. Figure 2b shows how the same module can be used to focus on the entity *Georgia* in a non-visual grounding domain; more generally, by representing every entity in the universe of discourse as a feature vector, we can obtain a distribution over entities that corresponds roughly to a logical set-valued denotation.

Having obtained such a distribution, existing neural approaches use it to immediately compute a weighted average of image features and project back into a labeling decision—a `describe` module (Figure 2c). But the logical perspective suggests a number of novel modules that might operate on attentions: e.g. combining them (by analogy to conjunction or disjunction) or inspecting them directly without a return to feature space (by analogy to quantification, Figure 2d). These modules are discussed in detail in Section 4. Unlike their formal counterparts, they are differentiable end-to-end, facilitating their integration into learned models. Building on previous work, we learn behavior for a collection of heterogeneous modules from (world, question, answer) triples.

The second contribution of this paper is a model for learning to assemble such modules compositionally. Isolated modules are of limited use—to obtain expressive power comparable to either formal approaches or monolithic deep networks, they must be composed into larger structures. Figure 2 shows simple examples of composed structures, but for realistic question-answering tasks, even larger networks are required. Thus our goal is to automatically induce variable-free, tree-structured computation descriptors. We can use a familiar functional notation from formal semantics (e.g. Liang et al., 2011) to represent these computations.[2] We write the two examples in Figure 2 as

```
(describe[color] find[bird])
```

and

```
(exists find[state])
```

respectively. These are **network layouts**: they specify a structure for arranging modules (and their lexical parameters) into a complete network. Andreas et al. (2016) use hand-written rules to deterministically transform dependency trees into layouts, and are restricted to producing simple structures like the above for non-synthetic data. For full generality, we will need to solve harder problems, like transforming *What cities are in Georgia?* (Figure 1) into

```
(and
    find[city]
    (relate[in] lookup[Georgia]))
```

In this paper, we present a model for learning to select such structures from a set of automatically generated candidates. We call this model a **dynamic neural module network**.

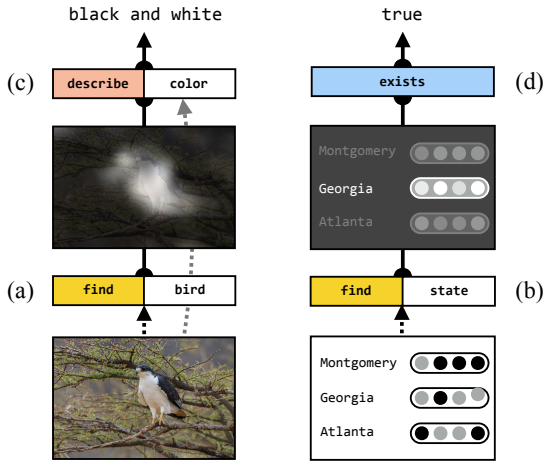

Figure 2: Simple neural module networks, corresponding to the questions *What color is the bird?* and *Are there any states?* (a) A neural `find` module for computing an attention over pixels. (b) The same operation applied to a knowledge base. (c) Using an attention produced by a lower module to identify the color of the region of the image attended to. (d) Performing quantification by evaluating an attention directly.

[2] But note that unlike formal semantics, the behavior of the primitive functions here is itself unknown.

3 Related work

There is an extensive literature on database question answering, in which strings are mapped to logical forms, then evaluated by a black-box execution model to produce answers. Supervision may be provided either by annotated logical forms (Wong and Mooney, 2007; Kwiatkowski et al., 2010; Andreas et al., 2013) or from (world, question, answer) triples alone (Liang et al., 2011; Pasupat and Liang, 2015). In general the set of primitive functions from which these logical forms can be assembled is fixed, but one recent line of work focuses on inducing new predicates functions automatically, either from perceptual features (Krishnamurthy and Kollar, 2013) or the underlying schema (Kwiatkowski et al., 2013). The model we describe in this paper has a unified framework for handling both the perceptual and schema cases, and differs from existing work primarily in learning a differentiable execution model with continuous evaluation results.

Neural models for question answering are also a subject of current interest. These include approaches that model the task directly as a multiclass classification problem (Iyyer et al., 2014), models that attempt to embed questions and answers in a shared vector space (Bordes et al., 2014) and attentional models that select words from documents sources (Hermann et al., 2015). Such approaches generally require that answers can be retrieved directly based on surface linguistic features, without requiring intermediate computation. A more structured approach described by Yin et al. (2015) learns a query execution model for database tables without any natural language component. Previous efforts toward unifying formal logic and representation learning include those of Grefenstette (2013) and Krishnamurthy and Mitchell (2013).

The visually-grounded component of this work relies on recent advances in convolutional networks for computer vision (Simonyan and Zisserman, 2014), and in particular the fact that late convolutional layers in networks trained for image recognition contain rich features useful for other downstream vision tasks, while preserving spatial information. These features have been used for both image captioning (Xu et al., 2015) and visual question answering (Yang et al., 2015).

Most previous approaches to visual question answering either apply a recurrent model to deep representations of both the image and the question (Ren et al., ; Malinowski et al., 2015), or use the question to compute an attention over the input image, and then answer based on both the question and the image features attended to (Yang et al., 2015; Xu and Saenko, 2015). Other approaches include the simple classification model described by Zhou et al. (2015) and the dynamic parameter prediction network described by Noh et al. (2015). All of these models assume that a fixed computation can be performed on the image and question to compute the answer, rather than adapting the structure of the computation to the question.

As noted, Andreas et al. (2016) previously considered a simple generalization of these attentional approaches in which small variations in the network structure per-question were permitted, with the structure chosen by (deterministic) syntactic processing of questions. Other approaches in this general family include the "universal parser" sketched by Bottou (2014), and the recursive neural networks of Socher et al. (2013), which use a fixed tree structure to perform further linguistic analysis without any external world representation. We are unaware of previous work that succeeds in simultaneously learning both the parameters for and structures of instance-specific neural networks.

4 Model

Recall that our goal is to map from questions and world representations to answers. This process involves the following variables:

1. w a world representation
2. x a question
3. y an answer
4. z a network layout
5. θ a collection of model parameters

Our model is built around two distributions: a **layout model** $p(z|x;\theta_\ell)$ which chooses a layout for a sentence, and a **execution model** $p_z(y|w;\theta_e)$ which applies the network specified by z to w.

For ease of presentation, we introduce these models in reverse order. We first imagine that z is always observed, and in Section 4.1 describe how to evaluate and learn modules parameterized by θ_e within

fixed structures. In Section 4.2, we move to the real scenario, where z is unknown. We describe how to predict layouts from questions and learn θ_e and θ_ℓ jointly without layout supervision.

4.1 Evaluating modules

Given a layout z, we assemble the corresponding modules into a full neural network (Figure 1c), and apply it to the knowledge representation. Intermediate results flow between modules until an answer is produced at the root. We denote the output of the network with layout z on input world w as $[\![z]\!]_w$; when explicitly referencing the substructure of z, we can alternatively write $[\![m(h^1, h^2)]\!]$ for a top-level module m with submodule outputs h^1 and h^2. We then define the execution model:

$$p_z(y|w) = ([\![z]\!]_w)_y \tag{1}$$

(This assumes that the root module of z produces a distribution over labels y.) The set of possible layouts z is restricted by module *type constraints*: some modules (like `find` above) operate directly on the input representation, while others (like `describe` above) also depend on input from specific earlier modules. Two base types are considered in this paper are Attention (a distribution over pixels or entities) and Labels (a distribution over answers).

Parameters are tied across multiple instances of the same module, so different instantiated networks may share some parameters but not others. Modules have both *parameter arguments* (shown in square brackets) and ordinary inputs (shown in parentheses). Parameter arguments, like the running `bird` example in Section 2, are provided by the layout, and are used to specialize module behavior for particular lexical items. Ordinary inputs are the result of computation lower in the network. In addition to parameter-specific weights, modules have global weights shared across all instances of the module (but not shared with other modules). We write $A, a, B, b, \ldots$ for global weights and u^i, v^i for weights associated with the parameter argument i. $\oplus$ and $\odot$ denote (possibly broadcasted) elementwise addition and multiplication respectively. The complete set of global weights and parameter-specific weights constitutes θ_e. *Every* module has access to the world representation, represented as a collection of vectors $w^1, w^2, \ldots$ (or W expressed as a matrix). The nonlinearity σ denotes a rectified linear unit.

The modules used in this paper are shown below, with names and type constraints in the first row and a description of the module's computation following.

Lookup (→ Attention)

`lookup[i]` produces an attention focused entirely at the index $f(i)$, where the relationship f between words and positions in the input map is known ahead of time (e.g. string matches on database fields).

$$[\![\texttt{lookup[}i\texttt{]}]\!] = e_{f(i)} \tag{2}$$

where e_i is the basis vector that is 1 in the ith position and 0 elsewhere.

Find (→ Attention)

`find[i]` computes a distribution over indices by concatenating the parameter argument with each position of the input feature map, and passing the concatenated vector through a MLP:

$$[\![\texttt{find[}i\texttt{]}]\!] = \text{softmax}(a \odot \sigma(Bv^i \oplus CW \oplus d)) \tag{3}$$

Relate (Attention → Attention)

`relate` directs focus from one region of the input to another. It behaves much like the `find` module, but also conditions its behavior on the current region of attention h. Let $\bar{w}(h) = \sum_k h_k w^k$, where h_k is the k^{th} element of h. Then,

$$[\![\texttt{relate[}i\texttt{]}(h)]\!] = \text{softmax}(a \odot \sigma(Bv^i \oplus CW \oplus D\bar{w}(h) \oplus e)) \tag{4}$$

And (Attention* → Attention)

`and` performs an operation analogous to set intersection for attentions. The analogy to probabilistic logic suggests multiplying probabilities:

$$[\![\texttt{and}(h^1, h^2, \ldots)]\!] = h^1 \odot h^2 \odot \cdots \tag{5}$$

Describe (Attention → Labels)

`describe[i]` computes a weighted average of w under the input attention. This average is then used to predict an answer representation. With $\bar{w}$ as above,

$$[\![\texttt{describe[}i\texttt{]}(h)]\!] = \text{softmax}(A\sigma(B\bar{w}(h) + v^i)) \tag{6}$$

Exists (Attention → Labels)

`exists` is the existential quantifier, and inspects the incoming attention directly to produce a label, rather than an intermediate feature vector like `describe`:

$$[\![\texttt{exists]}(h)]\!] = \text{softmax}\Big(\big(\max_k h_k\big)a + b\Big) \tag{7}$$

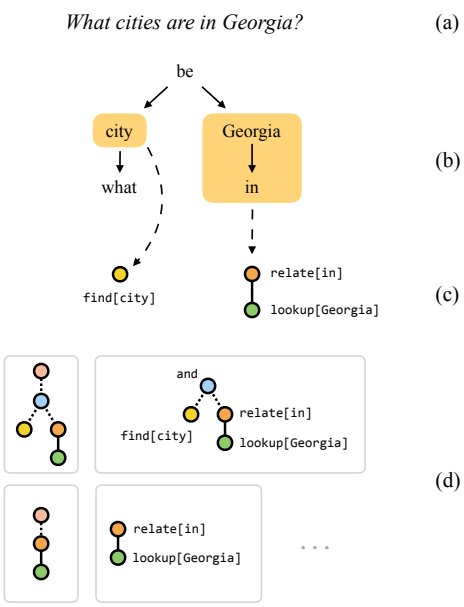

Figure 3: Generation of layout candidates. The input sentence (a) is represented as a dependency parse (b). Fragments of this dependency parse are then associated with appropriate modules (c), and these fragments are assembled into full layouts (d).

With z observed, the model we have described so far corresponds largely to that of Andreas et al. (2016), though the module inventory is different—in particular, our new `exists` and `relate` modules do not depend on the two-dimensional spatial structure of the input. This enables generalization to non-visual world representations.

Learning in this simplified setting is straightforward. Assuming the top-level module in each layout is a `describe` or `exists` module, the fully- instantiated network corresponds to a distribution over labels conditioned on layouts. To train, we maximize $\sum_{(w,y,z)} \log p_z(y|w;\theta_e)$ directly. This can be understood as a parameter-tying scheme, where the decisions about which parameters to tie are governed by the observed layouts z.

4.2 Assembling networks

Next we describe the layout model $p(z|x;\theta_\ell)$. We first use a fixed syntactic parse to generate a small set of candidate layouts, analogously to the way a semantic grammar generates candidate semantic parses in previous work (Berant and Liang, 2014).

A semantic parse differs from a syntactic parse in two primary ways. First, lexical items must be mapped onto a (possibly smaller) set of semantic primitives. Second, these semantic primitives must be combined into a structure that closely, but not exactly, parallels the structure provided by syntax. For example, *state* and *province* might need to be identified with the same field in a database schema, while *all states have a capital* might need to be identified with the correct (*in situ*) quantifier scope.

While we cannot avoid the structure selection problem, continuous representations simplify the lexical selection problem. For modules that accept a vector parameter, we associate these parameters with *words* rather than semantic tokens, and thus turn the combinatorial optimization problem associated with lexicon induction into a continuous one. Now, in order to learn that *province* and *state* have the same denotation, it is sufficient to learn that their associated parameters are close in some embedding space—a task amenable to gradient descent. (Note that this is easy only in an optimizability sense, and not an information-theoretic one—we must still learn to associate each independent lexical item with the correct vector.) The remaining combinatorial problem is to arrange the provided lexical items into the right computational structure. In this respect, layout prediction is more like syntactic parsing than ordinary semantic parsing, and we can rely on an off-the-shelf syntactic parser to get most of the way there. In this work, syntactic structure is provided by the Stanford dependency parser (De Marneffe and Manning, 2008).

The construction of layout candidates is depicted in Figure 3, and proceeds as follows:

1. Represent the input sentence as a dependency tree.
2. Collect all nouns, verbs, and prepositional phrases that are attached directly to a wh-word or copula.
3. Associate each of these with a layout fragment: Ordinary nouns and verbs are mapped to a single `find` module. Proper nouns to a single `lookup` module. Prepositional phrases are mapped to a depth-2 fragment, with a `relate` module for the preposition above a `find` module for the enclosed head noun.
4. Form subsets of this set of layout fragments. For each subset, construct a layout candidate by

joining all fragments with an `and` module, and inserting either a `measure` or `describe` module at the top (each subset thus results in two parse candidates.)

All layouts resulting from this process feature a relatively flat tree structure with at most one conjunction and one quantifier. This is a strong simplifying assumption, but appears sufficient to cover most of the examples that appear in both of our tasks. As our approach includes both categories, relations and simple quantification, the range of phenomena considered is generally broader than previous perceptually-grounded QA work (Krishnamurthy and Kollar, 2013; Matuszek et al., 2012).

Having generated a set of candidate parses, we need to score them. This is a ranking problem; as in the rest of our approach, we solve it using standard neural machinery. In particular, we produce an LSTM representation of the question, a feature-based representation of the query, and pass both representations through a multilayer perceptron (MLP). The query feature vector includes indicators on the number of modules of each type present, as well as their associated parameter arguments. While one can easily imagine a more sophisticated parse-scoring model, this simple approach works well for our tasks.

Formally, for a question x, let $h_q(x)$ be an LSTM encoding of the question (i.e. the last hidden layer of an LSTM applied word-by-word to the input question). Let $\{z_1, z_2, \ldots\}$ be the proposed layouts for x, and let $f(z_i)$ be a feature vector representing the ith layout. Then the score $s(z_i|x)$ for the layout z_i is

$$s(z_i|x) = a^\top \sigma(Bh_q(x) + Cf(z_i) + d) \quad (8)$$

i.e. the output of an MLP with inputs $h_q(x)$ and $f(z_i)$, and parameters $\theta_\ell = \{a, B, C, d\}$. Finally, we normalize these scores to obtain a distribution:

$$p(z_i|x;\theta_\ell) = e^{s(z_i|x)} \Big/ \sum_{j=1}^{n} e^{s(z_j|x)} \quad (9)$$

Having defined a layout selection module $p(z|x;\theta_\ell)$ and a network execution model $p_z(y|w;\theta_e)$, we are ready to define a model for predicting answers given only (world, question) pairs. The key constraint is that we want to minimize evaluations of $p_z(y|w;\theta_e)$ (which involves expensive application of a deep network to a large input representation), but can tractably evaluate $p(z|x;\theta_\ell)$ for all z (which involves application of a shallow network to a relatively small set of candidates). This is the opposite of the situation usually encountered semantic parsing, where calls to the query execution model are fast but the set of candidate parses is too large to score exhaustively.

In fact, the problem more closely resembles the scenario faced by agents in the reinforcement learning setting (where it is cheap to score actions, but potentially expensive to execute them and obtain rewards). We adopt a common approach from that literature, and express our model as a stochastic policy. Under this policy, we first *sample* a layout z from a distribution $p(z|x;\theta_\ell)$, and then apply z to the knowledge source and obtain a distribution over answers $p(y|z, w;\theta_e)$.

After z is chosen, we can train the execution model directly by maximizing $\log p(y|z, w;\theta_e)$ with respect to θ_e as before (this is ordinary backpropagation). Because the hard selection of z is non-differentiable, we optimize $p(z|x;\theta_\ell)$ using a policy gradient method. The gradient of the reward surface J with respect to the parameters of the policy is

$$\nabla J(\theta_\ell) = \mathbb{E}[\nabla \log p(z|x;\theta_\ell) \cdot r] \quad (10)$$

(this is the REINFORCE rule (Williams, 1992)). Here the expectation is taken with respect to rollouts of the policy, and r is the reward. Because our goal is to select the network that makes the most accurate predictions, we take the reward to be identically the negative log-probability from the execution phase, i.e.

$$\mathbb{E}[(\nabla \log p(z|x;\theta_\ell)) \cdot \log p(y|z, w;\theta_e)] \quad (11)$$

Thus the update to the layout-scoring model at each timestep is simply the gradient of the log-probability of the chosen layout, scaled by the accuracy of that layout's predictions. At training time, we approximate the expectation with a single rollout, so at each step we update θ_ℓ in the direction $(\nabla \log p(z|x;\theta_\ell)) \cdot \log p(y|z, w;\theta_e)$ for a single $z \sim p(z|x;\theta_\ell)$. θ_e and θ_ℓ are optimized using ADADELTA (Zeiler, 2012) with $\rho = 0.95$, $\varepsilon = 1\text{e}{-6}$ and gradient clipping at a norm of 10.

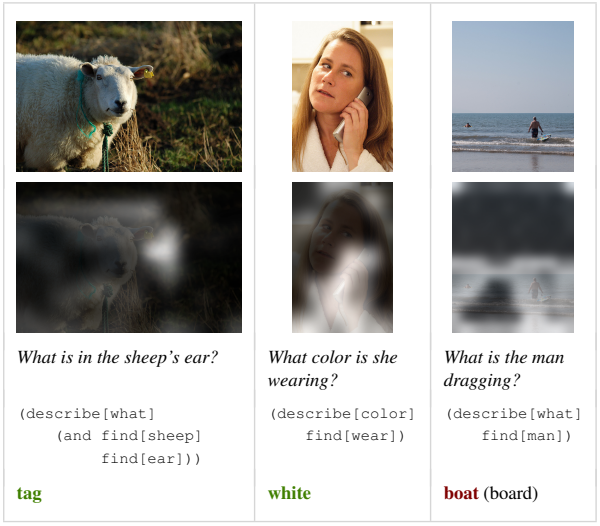

Figure 4: Sample outputs for the visual question answering task. The second row shows the final attention provided as input to the top-level `describe` module. For the first two examples, the model produces reasonable parses, attends to the correct region of the images (the ear and the woman's clothing), and generates the correct answer. In the third image, the verb is discarded and a wrong answer is produced.

5 Experiments

The framework described in this paper is general, and we are interested in how well it performs on datasets of varying domain, size and linguistic complexity. To that end, we evaluate our model on tasks at opposite extremes of both these criteria: a large visual question answering dataset, and a small collection of more structured geography questions.

5.1 Questions about images

Our first task is the recently-introduced Visual Question Answering challenge (VQA) (Antol et al., 2015). The VQA dataset consists of more than 200,000 images paired with human-annotated questions and answers, as in Figure 4.

We use the VQA 1.0 release, employing the development set for model selection and hyperparameter tuning, and reporting final results from the evaluation server on the test-standard set. For the experiments described in this section, the input feature representations w_i are computed by the the fifth convolutional layer of a 16-layer VGGNet after pooling (Simonyan and Zisserman, 2014). Input images are scaled to 448×448 before computing their representations. We found that performance on this task was best if the candidate layouts were relatively simple: only `describe`, `and` and `find` modules are used, and layouts contain at most two conjuncts.

	test-dev				test-std
	Yes/No	Number	Other	All	All
Zhou (2015)	76.6	35.0	42.6	55.7	55.9
Noh (2015)	80.7	37.2	41.7	57.2	57.4
NMN	77.7	37.2	39.3	54.8	55.1
NMN*	79.7	37.1	42.8	57.3	–
D-NMN	80.5	37.4	43.1	57.9	**58.0**

Table 1: Results on the VQA test server. NMN is the parameter-tying model from Andreas et al. (2015), while NMN* is a reimplementation using the same image processing pipeline as D-NMN. The model with dynamic network structure prediction achieves the best published results on this task.

One weakness of this basic framework is a difficulty modeling prior knowledge about answers (of the form *bears are brown*). This kinds of linguistic "prior" is essential for the VQA task, and easily incorporated. We simply introduce an extra hidden layer for recombining the final module network output with the input sentence representation $h_q(x)$ (see Equation 8), replacing Equation 1 with:

$$\log p_z(y|w,x) = (Ah_q(x) + B[\![z]\!]_w)_y \qquad (12)$$

(Now modules with output type Labels should be understood as producing an answer embedding rather than a distribution over answers.) This allows the question to influence the answer directly.

Results are shown in Table 1. The use of dynamic networks provides a small gain, most noticeably on yes/no questions. We achieve state-of-the-art results on this task, outperforming a highly effective visual bag-of-words model (Zhou et al., 2015), a model with dynamic network parameter prediction (but fixed network structure) (Noh et al., 2015), and a previous approach using neural module networks with no structure prediction (Andreas et al., 2016). For this last model, we report both the numbers from the original paper, and a reimplementation of the model that uses the same image preprocessing as the dynamic module network experiments in this paper. A more conventional attentional model has also been applied to this task (Yang et al., 2015); while we also outperform their reported performance, the evaluation uses different train/test split, so results are not directly comparable.

Model	Accuracy	
	GeoQA	GeoQA+Q
LSP-F	48	–
LSP-W	51	–
NMN	51.7	35.7
D-NMN	**54.3**	**42.9**

Table 2: Results on the GeoQA dataset, and the GeoQA dataset with quantification. Our approach outperforms both a purely logical model (LSP-F) and a model with learned perceptual predicates (LSP-W) on the original dataset, and a fixed-structure NMN under both evaluation conditions.

Some examples are shown in Figure 4. In general, the model learns to focus on the correct region of the image, and tends to consider a broad window around the region. This facilitates answering questions like *Where is the cat?*, which requires knowledge of the surroundings as well as the object in question.

5.2 Questions about geography

The next set of experiments we consider focuses on GeoQA, a geographical question-answering task first introduced by Krishnamurthy and Kollar (2013). This task was originally paired with a visual question answering task much simpler than the one just discussed, and is appealing for a number of reasons. In contrast to the VQA dataset, GeoQA is quite small, containing only 263 examples. Two baselines are available: one using a classical semantic parser backed by a database, and another which induces logical predicates using linear classifiers over both spatial and distributional features. This allows us to evaluate the quality of our model relative to other perceptually grounded logical semantics, as well as strictly logical approaches.

The GeoQA domain consists of a set of entities (e.g. states, cities, parks) which participate in various relations (e.g. north-of, capital-of). Here we take the world representation to consist of two pieces: a set of category features (used by the `find` module) and a different set of relational features (used by the `relate` module). For our experiments, we use a subset of the features originally used by Krishnamurthy et al. The original dataset includes no quantifiers, and treats the questions *What cities are in Texas?* and *Are there any cities in Texas?* identically. Because we are interested in testing the parser's ability to predict a variety of different structures, we introduce a new version of the dataset, GeoQA+Q, which distinguishes these two cases, and expects a Boolean answer to questions of the second kind.

Is Key Largo an island?

```
(exists (and lookup[key-largo] find[island]))
```

yes: correct

What national parks are in Florida?

```
(and find[park] (relate[in] lookup[florida]))
```

everglades: correct

What are some beaches in Florida?

```
(exists (and lookup[beach]
             (relate[in] lookup[florida])))
```

yes (daytona-beach): wrong parse

What beach city is there in Florida?

```
(and lookup[beach] lookup[city]
     (relate[in] lookup[florida]))
```

[none] (daytona-beach): wrong module behavior

Figure 5: Example layouts and answers selected by the model on the GeoQA dataset. For incorrect predictions, the correct answer is shown in parentheses.

Results are shown in Table 2. As in the original work, we report the results of leave-one-environment-out cross-validation on the set of 10 environments. Our dynamic model (D-NMN) outperforms both the logical (LSP-F) and perceptual models (LSP-W) described by (Krishnamurthy and Kollar, 2013), as well as a fixed-structure neural module net (NMN). This improvement is particularly notable on the dataset with quantifiers, where dynamic structure prediction produces a 20% relative improvement over the fixed baseline. A variety of predicted layouts are shown in Figure 5.

6 Conclusion

We have introduced a new model, the *dynamic neural module network*, for answering queries about both structured and unstructured sources of information. Given only (question, world, answer) triples as training data, the model learns to assemble neural networks on the fly from an inventory of neural models, and simultaneously learns weights for these modules so that they can be composed into novel structures. Our approach achieves state-of-the-art results on two tasks. We believe that the success of this work derives from two factors:

Continuous representations improve the expressiveness and learnability of semantic parsers: by replacing discrete predicates with differentiable neural network fragments, we bypass the challenging combinatorial optimization problem associated with induction of a semantic lexicon. In structured world representations, neural predicate representations allow the model to invent reusable attributes and relations not expressed in the schema. Perhaps more importantly, we can extend compositional question-answering machinery to complex, continuous world representations like images.

Semantic structure prediction improves generalization in deep networks: by replacing a fixed network topology with a dynamic one, we can tailor the computation performed to each problem instance, using deeper networks for more complex questions and representing combinatorially many queries with comparatively few parameters. In practice, this results in considerable gains in speed and sample efficiency, even with very little training data.

These observations are not limited to the question answering domain, and we expect that they can be applied similarly to tasks like instruction following, game playing, and language generation.

Acknowledgments

JA is supported by a National Science Foundation Graduate Fellowship. MR is supported by a fellowship within the FIT weltweit-Program of the German Academic Exchange Service (DAAD). This work was additionally supported by DARPA, AFRL, DoD MURI award N000141110688, NSF awards IIS-1427425 and IIS-1212798, and the Berkeley Vision and Learning Center.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *Proceedings of the International Conference on Computer Vision*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, volume 7, page 92.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Léon Bottou. 2014. From machine learning to machine reasoning. *Machine learning*, 94(2):133–149.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The Stanford typed dependencies representation. In *Proceedings of the International Conference on Computational Linguistics*, pages 1–8.

Edward Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. *Joint Conference on Lexical and Computational Semantics*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*.

Jayant Krishnamurthy and Tom Mitchell. 2013. Vector space semantic parsing: A framework for compositional vector space models. In *Proceedings of the ACL Workshop on Continuous Vector Space Models and their Compositionality*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233, Cambridge, Massachusetts.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics.

In *Proceedings of the Human Language Technology Conference of the Association for Computational Linguistics*, pages 590–599, Portland, Oregon.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2015. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the International Conference on Computer Vision*.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine Learning*.

Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. 2015. Image question answering using convolutional neural network with dynamic parameter prediction. *arXiv preprint arXiv:1511.05756*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems*.

K Simonyan and A Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, volume 45, page 960.

Huijuan Xu and Kate Saenko. 2015. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. *arXiv preprint arXiv:1511.05234*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2015. Stacked attention networks for image question answering. *arXiv preprint arXiv:1511.02274*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables. *arXiv preprint arXiv:1512.00965*.

Matthew D Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2015. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167*.