Few-Shot Object Recognition from Machine-Labeled Web Images

Zhongwen Xu* Linchao Zhu* Yi Yang
CAI, University of Technology Sydney
{zhongwen.s.xu, zhulinchao7, yee.i.yang}@gmail.com

Abstract

*With the tremendous advances made by Convolutional Neural Networks (ConvNets) on object recognition, we can now easily obtain adequately reliable machine-labeled annotations easily from predictions by off-the-shelf ConvNets. In this work, we present an "abstraction memory" based framework for few-shot learning, building upon machine-labeled image annotations. Our method takes large-scale machine-annotated dataset (*e.g.*, OpenImages) as an external memory bank. In the external memory bank, the information is stored in the memory slots in the form of key-value, in which image feature is regarded as the key and the label embedding serves as the value. When queried by the few-shot examples, our model selects visually similar data from the external memory bank and writes the useful information obtained from related external data into another memory bank,* i.e. *abstraction memory. Long Short-Term Memory (LSTM) controllers and attention mechanisms are utilized to guarantee the data written to the abstraction memory correlates with the query example. The abstraction memory concentrates information from the external memory bank to make the few-shot recognition effective. In the experiments, we first confirm that our model can learn to conduct few-shot object recognition on clean human-labeled data from the ImageNet dataset. Then, we demonstrate that with our model, machine-labeled image annotations are very effective and abundant resources for performing object recognition on novel categories. Experimental results show that our proposed model with machine-labeled annotations achieves great results, with only a 1% difference in accuracy between the machine-labeled annotations and the human-labeled annotations.*

1. Introduction

Innovations in the architecture of Convolutional Neural Networks (ConvNets) [21, 30, 27, 12] have resulted in tremendous improvements in image classification in the

*Indicates equal contribution.

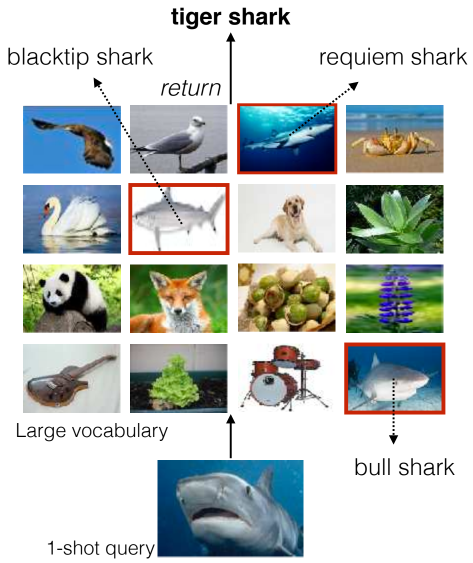


Figure 1. Given a large vocabulary of labels and their corresponding images, we conduct few-shot learning on a novel category which is not in the vocabulary and only has a handful of positive examples. The image examples in the vocabulary are stored in the external memory of our model, and the image example from the novel category queries the external memory. Our model returns helpful information according to visual similarity and LSTM controllers. The retrieved information, *i.e.*, visual features and their corresponding labels, are combined to classify this query image example.

past few years. With the increase in the capacity of neural networks, the demand for more labeled data in richer categories is rising. However, it is impractical and very expensive to manually label a dataset 10 times larger than ImageNet. This prompts us to design a new paradigm that can utilize the machine-labeled image annotations to enable rapid learning from novel object categories. Figure 1 illustrates the proposed task. Our major question in this work is as follows: Can we use machine-labeled web image annotations to rapidly conduct object recognition for *novel categories* with *only a handful of examples*?

We propose a new memory component in neural networks, namely *abstraction memory*, to concentrate information from the external memory bank, *e.g.*, large-scale object recognition datasets like ImageNet [7] and Open-Images [19], based on few-shot image queries. Previous methods which attempt to learn from different categories or datasets usually use a larger dataset for pre-training and then conduct fine-tuning on a relatively small dataset. The information of the large datasets is encoded in the learnable weights of the neural networks. In contrast to previous works, our model utilizes a content-based addressing mechanism with a Long Short-Term Memory (LSTM) controller to automatically decide where to read from and where to write into the memory. The neural network applies a soft attention mechanism [3] to the query image to find the appropriate information to read from the external memory and write into another memory. The abstraction memory records helpful information for the specific few-shot object recognition, so that the classification network can utilize readouts from the abstraction memory to recognize the objects from novel categories.

Previous methods only discover the relationship of the word embeddings [16, 23] between the category labels, whereas we fully utilize the visual similarity between the examples of few-shot categories and the external memory bank to make the proposed framework more robust to noisy labels. If the external memory data is inconsistent with its label, this sample will be rejected during the visual matching process. This property makes the use of a large-scale machine annotated dataset, *e.g.*, OpenImages [19] feasible. The machine-labeled annotations for images could be predicted by off-the-shelf ConvNet models (*e.g.*, ResNets [12]), but although these annotations are reasonably good, they are not perfect. In this scenario, the external dataset can also consist of images obtained by querying keywords in search engines (*e.g.*, Google Images), and images crawled from social image sharing sites (*e.g.*, Flickr). In the experiment section, we show that the results of our proposed method using machine-annotated data differ from human-labeled data by a minor gap $\approx 1\%$.

When novel categories arrive, the network accesses and queries the external memory, retrieves the related information, and writes into abstraction memory. We organize the memory in the data structure `key:value`, which was first proposed in Key-Value Memory Networks (KV-MemNNs) [24]. We note that we have the implementation of our model, including LSTM controllers, abstraction memory, and reading mechanisms, differs significantly from KV-MemNNs. Moreover, KV-MemNNs were developed in natural language understanding area, and their memory access is limited to the most recent few sentences. We extend the key-value storage concept into computer vision applications by novel modifications to enable scalability. We formulate the image embedding as the `key` and the word embedding of the annotated label as the `value`. The additional memory for abstraction extracts information from the external memory and learns task-specific representation for the few-shot learning while maintaining efficiency.

Our contributions are as follows.

1. We propose a novel task for learning few-shot object recognition on machine-labeled image annotations. We demonstrate that with sufficiently reliable machine-labeled annotations, it is possible to achieve excellent performance with a only a minor deviation in accuracy (about 1%) compared to learning from human-labeled annotations;

2. We propose the incorporation of a novel memory component, namely abstraction memory, into the Memory Networks [36] structure. The abstraction memory alleviates the time-consuming content-based addressing of the external memory, enabling the model to be scalable and efficient;

3. We utilize both visual embeddings and label embeddings in a form of key-value to make the system robust to imperfect labeling. This enables the model to learn from the machine-labeled web images to obtain rich signals for visual representation, which is very suitable for real-world vision application. We conduct few-shot learning of unseen visual categories, making rapid and accurate predictions without extensive iterations of positive examples.

We demonstrate the advantages of our method over state-of-the-art models such as Matching Networks [33], KV-MemNNs [24], Exemplar-SVMs [22], and Nearest Neighbors [5] on few-shot object recognition tasks.

2. Related Work

Learning Visual Features from the Web. Chen *et al.* [6] proposed a never ending image learner (NEIL) to extract common sense relationships and predict instance-level labels on web images. NEIL bootstraps the image classifier by training from top-ranked images in Google images as positive samples, and a semi-supervised learning method is then used to mine object relationships. Divvala *et al.* [8] leveraged Google Books to enrich the visual categories into very broad ranges, including actions, interactions, and attributes. These works focus on mining relationships between objects and intra-class; however, these approaches are prone to error, since classification mistakes will accumulate along the iteration procedure due to the bootstrapping nature. Joulin *et al.* [15] argued that ConvNets can learn from scratch in a weakly-supervised way, by utilizing 100M Flickr images annotated with noisy captions. Our work utilizes established state-of-the-art human-level ConvNets to alleviate the error that could come from seed images. We focus on a different task of rapidly learning few-shot classification by benefiting from the rich vocabulary of the web resources.

External Memory in Neural Networks. Neural Turing Machines (NTMs) [11] and Memory networks (MemNNs) [36] are two recently proposed families of neural networks augmented with external memory structure. NTMs are fully differentiable attempts of Turing machine neural network implementation which learn to read from and write into external memory. NTMs have demonstrated success on tasks of learning simple algorithms such as copying input strings and reversing input strings. MemNN was proposed to reason from facts/story for question answering, building relationships between "story", "question" and "answer". End-to-end Memory Networks (MemN2N) [28] eliminate the requirements of strong supervision of MemNNs and train the networks in an end-to-end fashion. Key-Value Memory Networks (KV-MemNNs) [24] incorporate structural information in the form of key-value which delivers more flexible ways to store knowledge bases or documents. Though yielding excellent performance on toy question-answering benchmarks, Memory Networks applications are still limited in natural language understanding domains. We recognize the great expressive power of neural networks augmented with external memory, and build upon these works to learn rapid visual classification from machine-labeled images.

One-shot Learning. Training neural networks notoriously requires thousands of examples for each category, which means that conventional neural models are highly data inefficient. Fei-Fei *et al*. [9] pioneered one-shot learning of object categories and provided an important insight: taking advantage of knowledge learned from previous categories, it is possible to learn about a category from just one, or a handful of images [9]. Inspired by Bayesian Program Learning (BPL) for concept abstraction in Lake *et al*. [20] and augmented memory neural structures [11, 36], Memory-Augmented Neural Networks (MANNs) [26] utilize a meta-learning paradigm to learn the binding of samples and labels from shuffled training batches. Matching networks [33] employ metric learning and improve over MANNs significantly by utilizing the attention kernel and the set-to-set framework [32].

3. Proposed Approach

3.1. Preliminaries

We briefly introduce some technical preliminaries on Memory Network variants before discussing our proposed model.

Memory Networks (MemNNs) [36] are a new family of learning models which augment neural networks with *external memory*. The major innovation of Memory Networks is the long-term memory component $\boldsymbol{\mathcal{M}}$, which enables the neural networks to reason and access the information from long-term storage. End-to-End Memory Networks (MemN2N) [28] implement Memory Networks in a continuous form, so that end-to-end training becomes feasible. The recently proposed Key-Value Memory Networks (KV-MemNNs) [24] extend MemNNs [36] and MemN2N [28] with structural information storage in the memory slots. Instead of having only single vector representation in the memory component, as in MemN2N, KV-MemNNs make use of pairs of vectors in the memory slots, *i.e.*, `key: value`. The incorporation of the structural storage of the Key-Value form into the memory slots brings much more flexibility, which enriches the expressive power of the neural networks. The Key-Value property makes information retrieval from the external memory natural.

The Memory Network variants (MemNNs, MemN2N, and KV-MemNNs) have been proposed for natural language understanding, and researchers often only validate these models on question answering tasks such as bAbI tasks [35].

3.2. Model Overview

In this work, we propose a novel Memory Networks architecture to tackle the few-shot visual object recognition problem. It retains the key-value structure, but in contrast to KV-MemNNs, we utilize Long Short-Term Memory (LSTM) as a "controller" when accessing and writing to memory. Moreover, we introduce a novel memory component, namely *abstraction memory*, to enable task-specific feature learning and obtain scalability. The distinct nature of our proposed abstraction memory makes the neural network "remember" the ever present external memories, analogous to the memory cell $\boldsymbol{c}$ in LSTMs but much more expressive. The incorporation of abstraction memory enables stochastic external memory training, *i.e.*, we can sample batches from the a huge external memory pool. In contrast to our work, existing Memory Networks limit their access to external memory to a very small number, *e.g.*, MemN2N limit their access to external memory to the most recent 50 sentences [28].

The overview of our model is shown in Figure 2. The whole procedure of our proposed model is illustrated as follows. Note that we re-formulate `key: value` as (key, value) in the rest of this work.

$$\begin{aligned}
\boldsymbol{q}, \boldsymbol{\mathcal{M}}_{\text{ext}} &= \texttt{EMBED}(I, \{\mathcal{I}_{\text{web}}, \mathcal{L}_{\text{web}}\}) && (1)\\
(\boldsymbol{z}_{\text{key}}, \boldsymbol{z}_{\text{val}}) &= \texttt{READ}(\boldsymbol{q}, \boldsymbol{\mathcal{M}}_{\text{ext}}), && (2)\\
\boldsymbol{\mathcal{M}}_{\text{abs}} &\leftarrow \texttt{WRITE}(\boldsymbol{q}, (\boldsymbol{z}_{\text{key}}, \boldsymbol{z}_{\text{val}}), \boldsymbol{\mathcal{M}}_{\text{abs}}), && (3)\\
(\boldsymbol{u}_{\text{key}}, \boldsymbol{u}_{\text{val}}) &= \texttt{READ}(\boldsymbol{q}, \boldsymbol{\mathcal{M}}_{\text{abs}}), && (4)\\
\hat{\boldsymbol{y}} &= \texttt{CLS}([\boldsymbol{u}_{\text{key}}, \boldsymbol{u}_{\text{val}}]). && (5)
\end{aligned}$$

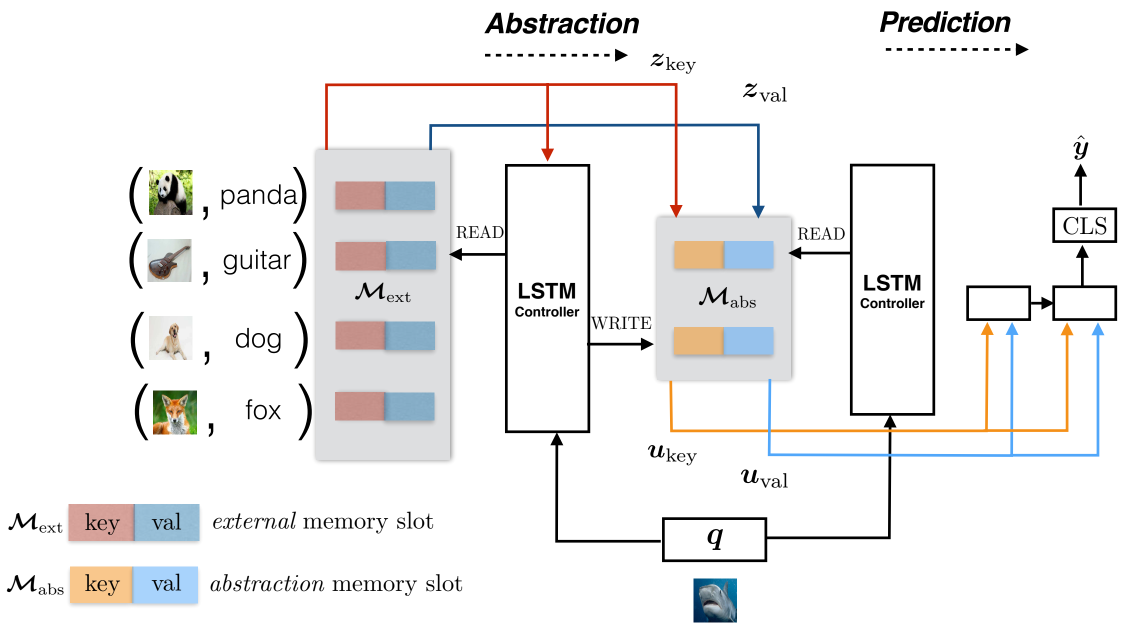

Abstraction
Prediction
z_{key}
z_{val}
panda
guitar
dog
fox
$\mathcal{M}_{\text{ext}}$
READ
LSTM
Controller
WRITE
$\mathcal{M}_{\text{abs}}$
READ
LSTM
Controller
$\hat{y}$
CLS
u_{key}
u_{val}
q
$\mathcal{M}_{\text{ext}}$ key val *external* memory slot
$\mathcal{M}_{\text{abs}}$ key val *abstraction* memory slot

Figure 2. An illustration of our proposed model. Best viewed in color.

We elaborate on each of the operation in the procedure, and all of the following operations are parameterized by neural networks:

1. `Embed` is a transformation from the raw inputs to their feature representation. We denote the network to extract image feature as Φ_{img} and the one to extract vector representation for label as Φ_{label}. Given an image I from a novel category, and a group of web images with labels, denoted as $\mathcal{I}_{\text{web}}$ and $\mathcal{L}_{\text{web}}$, where $\mathcal{I}$ is the image set and $\mathcal{L}$ is the label set, the input image I is sampled from unseen categories, and the embedded feature for the query image is referred to as query $\boldsymbol{q}$ following the notation in Memory Networks. The web images are embedded in the external memory $\mathcal{M}_{\text{ext}}$ through the same embedding networks Φ_{img} and Φ_{label};

2. `READ` takes the query $\boldsymbol{q}$ as input and conducts content-based addressing on the external memory $\mathcal{M}_{\text{ext}}$, to find related information according to a similarity metric with $\boldsymbol{q}$. The external memory is also called the support set in Memory Networks. The output of `READ` is a pair of vectors in key-value form, *i.e.*, $(\boldsymbol{z}_{\text{key}}, \boldsymbol{z}_{\text{val}})$, as shown in Eqn. (2);

3. `WRITE` takes a query $\boldsymbol{q}$ and key-value pair $(\boldsymbol{z}_{\text{key}}, \boldsymbol{z}_{\text{val}})$ as inputs to conduct a write operation. The content-based addressing is based on matching input with $\mathcal{M}_{\text{abs}}$, and then updating the content of the corresponding abstraction memory slots as in Eqn. (3);

4. `READ` from abstraction memory (Eqn. (4)) is for the classification stage. Take the input query $\boldsymbol{q}$ to match the abstraction memory $\mathcal{M}_{\text{abs}}$. The obtained pairs of vectors (*i.e.*, $(\boldsymbol{u}_{\text{key}}, \boldsymbol{u}_{\text{val}})$) are concatenated to be fed into the classification network;

5. `CLS` operation takes the readout key-value $(\boldsymbol{z}_{\text{key}}, \boldsymbol{z}_{\text{val}})$ and concatenates them into one vector $\boldsymbol{z}_{\text{cls}} = [\boldsymbol{z}_{\text{key}}, \boldsymbol{z}_{\text{val}}]$. Then $\boldsymbol{z}_{\text{cls}}$ goes through a Fully-Connected (`FC`) layer where: $\texttt{FC}(\boldsymbol{x}) = \boldsymbol{w}^{\top}\boldsymbol{x}+\boldsymbol{b}$, and a `Softmax` layer.

   Section 3.3.4 shows an LSTM variant of the `CLS` operation.

3.3. Model Components

3.3.1 Long Short-Term Memory

In our model, Long Short-Term Memory (LSTM) [14] plays an important role in the `READ`, `WRITE` and `CLS` procedures and serves as the *controller* of the memory addressing. LSTM is a special form of Recurrent Neural Networks (RNNs). LSTM addresses the vanishing gradient problem [4] of RNNs by introducing an *internal* memory cell to encode information from the previous steps. LSTM has resurged due to the success of sequence to sequence modeling [29] on machine translation [3], image captioning [34, 17, 37], video classification [38], video captioning [31, 25], *etc.* Following the notations of Zaremba *et al.* [39] and Xu *et al.* [37] and assuming $\boldsymbol{x}_t \in \mathbb{R}^D$, $\boldsymbol{T}_{D+d,4d} : \mathbb{R}^{D+d} \to \mathbb{R}^{4d}$ denotes an affine transformation from $\mathbb{R}^{D+d}$ to $\mathbb{R}^{4d}$, LSTM is implemented as:

$$\begin{pmatrix} \boldsymbol{i}_t \\ \boldsymbol{f}_t \\ \boldsymbol{o}_t \\ \boldsymbol{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \boldsymbol{T}_{D+d,4d} \begin{pmatrix} \boldsymbol{x}_t \\ \boldsymbol{h}_{t-1} \end{pmatrix} \tag{6}$$

$$\boldsymbol{c}_t = \boldsymbol{f} \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t \tag{7}$$

$$\boldsymbol{h}_t = \boldsymbol{o} \odot \tanh(\boldsymbol{c}_t), \tag{8}$$

where $\boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{c}_t, \boldsymbol{o}_t$ are the input, forget, memory, output gates respectively, σ and $\tanh$ are element-wise activation functions, $\boldsymbol{x}_t$ is the input to the LSTM in the t-th step and $\boldsymbol{h}_t$ is the hidden state of the LSTM in the t-th step.

For simplicity of notation, we denote one computational step of the LSTM recurrence as a function `LSTM`, defined as:

$$\boldsymbol{h}_t = \texttt{LSTM}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}). \tag{9}$$

3.3.2 Reading from the Memory

In this section, we describe the mechanism for reading information from the memory. Given an external memory with buffer size N_1, $\boldsymbol{\mathcal{M}} = \{(\boldsymbol{m}^1_{\text{key}}, \boldsymbol{m}^1_{\text{val}}), (\boldsymbol{m}^2_{\text{key}}, \boldsymbol{m}^2_{\text{val}}), \ldots, (\boldsymbol{m}^{N_1}_{\text{key}}, \boldsymbol{m}^{N_1}_{\text{val}})\}$, where each memory slot $\boldsymbol{m}^i$ is encoded as a key-value structure, *i.e.*, $(\boldsymbol{m}^i_{\text{key}}, \boldsymbol{m}^i_{\text{val}})$, or equivalently $\boldsymbol{m}^i_{\text{key}} : \boldsymbol{m}^i_{\text{val}}$. $\boldsymbol{m}^i_{\text{key}} \in \mathbb{R}^{d_1}, \boldsymbol{m}^i_{\text{val}} \in \mathbb{R}^{d_2}$, where d_1 is the dimension of the image embedding (*i.e.*, the `key` part) in the memory slot, and d_2 denotes the dimension of the label embedding (*i.e.*, the `val` part) in the memory slot. We use the tuple notation $(\boldsymbol{m}^i_{\text{key}}, \boldsymbol{m}^i_{\text{val}})$ in the rest. We apply the reading mechanisms from the set-to-set framework [32] on the memory bank. For each time step t, we have:

$$\begin{aligned} \boldsymbol{q}_t &= \texttt{LSTM}(\boldsymbol{0}, \boldsymbol{q}^*_{t-1}) && (10) \\ e_{i,t} &= \boldsymbol{q}_t^\top \boldsymbol{m}^i_{\text{key}} && (11) \\ a_{i,t} &= \texttt{Softmax}(e_{i,t}) && (12) \\ \boldsymbol{z}^t_{\text{key}} &= \sum_i a_{i,t} \boldsymbol{m}^i_{\text{key}} && (13) \\ \boldsymbol{z}^t_{\text{val}} &= \sum_i a_{i,t} \boldsymbol{m}^i_{\text{val}} && (14) \\ \boldsymbol{q}^*_t &= [\boldsymbol{q}_t, \boldsymbol{z}^t_{\text{key}}]. && (15) \end{aligned}$$

$(\boldsymbol{m}^i_{\text{key}}, \boldsymbol{m}^i_{\text{val}})$, $i = 1, 2, \ldots, N_1$, are all the memory slots stored in $\boldsymbol{\mathcal{M}}$. When the query $\boldsymbol{q}_t$ comes, it conducts dot product with all of the `key` parts of the memory slot $\boldsymbol{m}^i_{\text{key}}$ (Eqn. (11)), to obtain the similarity metric $e_{i,t}$ between query image $\boldsymbol{q}_t$ and the image in the memory slot $\boldsymbol{m}^i_{\text{key}}$. The `Softmax` operation of Eqn. (12) generates an attention weight $a_{i,t}$ over the whole memory $\boldsymbol{\mathcal{M}}$. Then, Eqn. (13) and Eqn. (14) utilize the learned attention weight $a_{i,t}$ to read the `key` part and the `value` part, *i.e.*, label embedding, from the external memory. The readout operation blends all of the key/value vectors $\boldsymbol{m}^i_{\text{key}}/\boldsymbol{m}^i_{\text{val}}$ with the attention weight $a_{i,t}$ to obtain the readout vectors $\boldsymbol{z}^t_{\text{key}}$ and $\boldsymbol{z}^t_{\text{val}}$. Lastly, $\boldsymbol{z}^t_{\text{key}}$ is concatenated with query $\boldsymbol{q}_t$, producing $\boldsymbol{q}^*_t$ to be fed into the next step as the input of LSTM (Eqn. (10)). The above reading procedure loops over the memory for T timesteps, obtaining T readout pairs of vectors, *i.e.*, $\{(\boldsymbol{z}^1_{\text{key}}, \boldsymbol{z}^1_{\text{val}}), (\boldsymbol{z}^2_{\text{key}}, \boldsymbol{z}^2_{\text{val}}), \ldots, (\boldsymbol{z}^T_{\text{key}}, \boldsymbol{z}^T_{\text{val}})\}$. The LSTM controller takes no input but computes the recurrent state to control the reading operation. For more details, please refer to the vector version (the memory slot is in the form of a vector instead of a key-value) of this reading mechanism [32].

After T-step `READ` operations over memory $\boldsymbol{\mathcal{M}}$ (which could be either $\boldsymbol{\mathcal{M}}_{\text{ext}}$ or $\boldsymbol{\mathcal{M}}_{\text{abs}}$), we can obtain:

$$\boldsymbol{\mathcal{Z}} = \{(\boldsymbol{z}^1_{\text{key}}, \boldsymbol{z}^1_{\text{val}}), (\boldsymbol{z}^2_{\text{key}}, \boldsymbol{z}^2_{\text{val}}), \ldots, (\boldsymbol{z}^T_{\text{key}}, \boldsymbol{z}^T_{\text{val}})\}. \tag{16}$$

3.3.3 Abstraction Memory

We propose to utilize a novel memory component, namely *abstraction memory*, in our implementation of Memory Networks. The abstraction memory has the following properties:

1. It learns task-specific representation for the few-shot object recognition task;

2. It attempts to tackle the problem of efficiency of content-based addressing over a large external memory pool.

Abstraction memory is a *writable* memory bank $\boldsymbol{\mathcal{M}}_{\text{abs}}$, with buffer size N_2. It satisfies $N_2 < N_1$, where N_1 is the buffer size of the external memory bank $\boldsymbol{\mathcal{M}}_{\text{ext}}$. We denote $\boldsymbol{\mathcal{M}}_{\text{abs}} = \{(\tilde{\boldsymbol{m}}^1_{\text{key}}, \tilde{\boldsymbol{m}}^1_{\text{val}}), (\tilde{\boldsymbol{m}}^2_{\text{key}}, \tilde{\boldsymbol{m}}^2_{\text{val}}), \ldots, (\tilde{\boldsymbol{m}}^{N_2}_{\text{key}}, \tilde{\boldsymbol{m}}^{N_2}_{\text{val}})\}$, where $\tilde{\boldsymbol{m}}^i_{\text{key}} \in \mathbb{R}^{\tilde{d}_1}$, $\tilde{\boldsymbol{m}}^i_{\text{val}} \in \mathbb{R}^{\tilde{d}_2}$, $\tilde{d}_1$ is the dimension of the `key` vector stored in the memory slot, and $\tilde{d}_2$ is the dimension of the `value` part stored in the memory slot.

Writing. Unlike the *external* memory bank, the *abstraction* memory bank is "writable", which means the neural networks can learn to update the memory slots in the storage by remembering and abstracting what is important for specific tasks. The memory update is according to an embedding (*i.e.*, through an `FC` layer) of the readout $(\boldsymbol{z}_{\text{key}}, \boldsymbol{z}_{\text{val}})$ from the larger external memory bank $\boldsymbol{\mathcal{M}}_{\text{ext}}$.

Following the writing operation proposed in Neural Turing Machines (NTMs) [11], we conduct the differentiable `WRITE` operation on the abstraction memory bank $\boldsymbol{\mathcal{M}}_{\text{abs}}$. The LSTM controller produces *erase* vectors $\boldsymbol{e}_{\text{key}} \in \mathbb{R}^{\tilde{d}_1}$, $\boldsymbol{e}_{\text{val}} \in \mathbb{R}^{\tilde{d}_2}$, and *add* vectors $\boldsymbol{a}_{\text{key}} \in \mathbb{R}^{\tilde{d}_1}$, $\boldsymbol{a}_{\text{val}} \in \mathbb{R}^{\tilde{d}_2}$. Note that each element of the erase vector satisfies $0 < \boldsymbol{e}^i_{\text{key}} < 1$

and $0 < \boldsymbol{e}_{\text{val}}^{i} < 1$, which can be implemented by passing through a Sigmoid function $\sigma(x)$.

For each memory slot $\tilde{\boldsymbol{m}}^{i}$, the `WRITE` operation conducts the following updates in the abstraction memory bank $\boldsymbol{\mathcal{M}}_{\text{abs}}$. For each timestep t, we have

$$\tilde{\boldsymbol{m}}_{\text{key}}^{i} \leftarrow \tilde{\boldsymbol{m}}_{\text{key}}^{i}(\mathbf{1} - w_{i,t}\boldsymbol{e}_{\text{key}}) + w_{i,t}\boldsymbol{a}_{\text{key}}, \quad (17)$$

$$\tilde{\boldsymbol{m}}_{\text{val}}^{i} \leftarrow \tilde{\boldsymbol{m}}_{\text{val}}^{i}(\mathbf{1} - w_{i,t}\boldsymbol{e}_{\text{val}}) + w_{i,t}\boldsymbol{a}_{\text{val}}. \quad (18)$$

The vector $\boldsymbol{w}_t$ is used for addressing mechanisms in the `WRITE` operation [11]. However, in contrast to NTMs, we do not utilize location-based addressing but only content-based addressing over the abstraction memory $\boldsymbol{\mathcal{M}}_{\text{abs}}$. The vector $\boldsymbol{w}_t$ can be calculated as in Eqn. (11) and Eqn. (12), by replacing $\boldsymbol{m}_{\text{key}}$ of $\boldsymbol{\mathcal{M}}_{\text{ext}}$ into $\tilde{\boldsymbol{m}}_{\text{key}}$ of $\boldsymbol{\mathcal{M}}_{\text{abs}}$.

3.3.4 Label Prediction

When it reaches the prediction stage, our model reads $(\boldsymbol{u}_{\text{key}}, \boldsymbol{u}_{\text{val}})$ from the abstraction memory $\mathcal{M}_{\text{abs}}$, as shown in Eqn. (4). The reading mechanism has been illustrated in Section 3.3.2. Reading from the memory is a recurrent process, with T timesteps, and we can fetch readouts $\boldsymbol{\mathcal{U}} = \{[\boldsymbol{u}_{\text{key}}^{1}, \boldsymbol{u}_{\text{val}}^{1}], [\boldsymbol{u}_{\text{key}}^{2}, \boldsymbol{u}_{\text{val}}^{2}], \ldots, [\boldsymbol{u}_{\text{key}}^{T}, \boldsymbol{u}_{\text{val}}^{T}]\}$ to obtain enough information for few-shot classification, where $[\boldsymbol{u}_{\text{key}}^{i}, \boldsymbol{u}_{\text{val}}^{i}]$ denotes the concatenation of two vectors into one. We then run an LSTM on top of the sequence $\boldsymbol{\mathcal{U}}$, obtain the final state output $\boldsymbol{h}_T$ from the LSTM, and then feed $\boldsymbol{h}_T$ into an `FC` layer and a `Softmax` layer to output the prediction $\hat{\boldsymbol{y}}$.

In this way, our model fully utilizes the readout vectors with both visual information and label embedding information to conduct classification. These readout vectors are from abstraction memory, which learns to adapt in specific tasks, *e.g.*, few-shot object recognition.

3.4. Training

We apply a standard cross entropy loss between the prediction $\hat{\boldsymbol{y}}$ and the groundtruth $\boldsymbol{y}$, where $\boldsymbol{y}$ is the one-hot representation of the groundtruth label.

All of the operations and components in our model are fully differentiable, which means we can train our model with stochastic gradient descent (SGD) in an end-to-end way.

3.5. Inference

In the inference (testing) stage, we do not make the external memory $\boldsymbol{\mathcal{M}}_{\text{ext}}$ available, since the abstraction memory $\boldsymbol{\mathcal{M}}_{\text{abs}}$ has stored all of the required information in the form of key-value in the memory slots. Thus, on the inference stage, we only run the prediction process (*c.f.* Section 3.3.4) on the fetched vectors from $\boldsymbol{\mathcal{M}}_{\text{abs}}$. The predicted label is obtained by an `argmax` operation over the softmax probability output $\hat{\boldsymbol{y}}$.

4. Experiments

We evaluate our proposed model using two different external image sources, *i.e.*, ImageNet [7] dataset and Open-Images [19] dataset. In this section, we describe the specific model configurations used in the experiments, and show the results of the few-shot recognition model trained from clean human-labeled annotations and machine-labeled annotations. Our model is implemented using TensorFlow [1].

4.1. Preprocessing

We use features from top convolutional layers as image embeddings. In all our experiments, we use the last layer activations before the final classification from the ResNet-200 [13] model pretrained on ImageNet [7]. This single model achieved top-5 error of 5.79% on the ILSVRC 2012 validation set. Following standard image preprocessing practice, images are first resized to 256 on the short side and the central 224×224 subregion is cropped; we thus obtain an image embedding with the feature dimension of 2,048. We apply the word embedding from the state-of-the-art language modeling model [16] in our label to word embedding mapping. We follow the instructions provided by the authors to extract embeddings for each word in the vocabulary, and embeddings are averaged if there are multiple words for one category. The embedding length is 1,024, thus we have the embedding matrix of $|V|$ by 1,024, where $|V|$ is the size of the vocabulary V. The ResNet for visual feature extraction and the label embedding matrix will *not* be updated during training.

4.2. Model Specifications

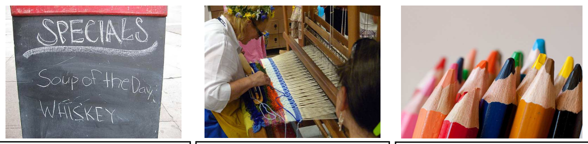
For all the LSTM models, we use one-layer LSTM with hidden unit size of 1,024. In particular, we utilize Layer Normalization [2] for the gates and states in the cell, which we found was crucial to train our model. Layer Normalization helps to stabilize the learning procedure in RNNs, without which we could not train the network successfully. Dropout is used in the input and output of LSTM and we set the Dropout probability to 0.5. The default model parameters are described as follows. We use $N_1 = 1,000$ memory slots for the external memory bank and $N_2 = 500$ memory slots for the abstraction memory. Both `key` and `value` vectors stored in the abstraction memory have the dimensionality of 512. The controller iterates $T = 5$ times when abstracting information from the external memory banks. We use the default model parameters in all the experiments unless otherwise stated.

Our model is trained with an ADAM optimizer [18] with learning rate of 1×10^{-4} and clip the norm of the global gradients at 10 to avoid the gradient exploding problem [29]. Weights in the neural network are initialized with Glorot uniform initialization [10] and weight decay of 1×10^{-4} is applied for regularization.

4.3. Datasets

ImageNet: ImageNet is a widely used image classification benchmark. There are two sets in the ImageNet dataset. One part is used in the ILSVRC classification competitions, namely ILSVRC 2012 CLS. This part contains exactly 1,000 classes with about 1,200 images per class, with well-verified human-labeled annotations. The other set of ImageNet is the whole set, which consists of about 21,000 categories.

OpenImages. The recently released OpenImages dataset [19] consisting of web images with machine-labeled annotations. We utilize the validation set with 167,057 images to conduct our experiment, since both machine-labeled annotations and human-labeled annotations are provided only in the validation set. There are 7,844 distinct labels in OpenImages, whose label vocabulary and diversity is much richer than the ILSVRC 2012 CLS dataset. Since this dataset is relatively new, we provide example images in Figure 3. We can see that the OpenImages dataset has a wider vocabulary than the ImageNet ILSVRC 2012 dataset, which is beneficial for generalization to novel categories.



“/m/0f1tz”,“calligraphy”,0.5	“/m/0dp7g”,“weaving”,0.8	“/m/01sdr”,“color”,1.0
“/m/07s6nbt”,“text”, 0.9	“/m/0jjw”,“art”,0.8	“/m/063w2”,“pencil”,0.9
“/m/07b9p1”,“plaque”,0.5	“/m/0cl71”,“loom”,0.7	“/m/02cqfm”,“close-up”,0.7

Figure 3. Sample images from the OpenImages dataset. Annotations on the images are shown in the bottom. The annotations listed are “label id”, “label name”, “confidence” tuples.

4.4. Few-shot Learning with Human-labeled annotations

We first validate our model on the task of few-shot classification using human-labeled clean data.

For few-shot image classification, the training set has only a small number of examples, and the basic task can be denoted as N-way k-shot classification (following the notation of Matching Networks [33]), in which N class images need to be classified and each class is provided with k labeled examples (k is usually less than 10).

Dataset. We now construct our dataset for few-shot learning. We select 100 classes for learning by randomly choosing 100 categories from the entire 21,000 categories in the ImageNet dataset, excluding the 1,000 categories in the ILSVRC 2012 CLS vocabulary. For testing, there are 200 images per category and the training set has k examples per category. We use settings of $k = 1$, $k = 5$, $k = 10$, *i.e.*, there are 1 example, 5 examples and 10 examples in the training set.

Methods	1-shot	5-shot	10-shot
k-NN (l_1)	38.8	57.0	62.9
k-NN (l_2)	38.6	56.4	62.1
E-SVM	45.1	62.3	68.0
KV-MemNNs	43.2 ($\pm$0.4)	66.6 ($\pm$0.2)	72.8 ($\pm$0.2)
Ours	**45.8**	**68.0**	**73.5**

Table 1. Comparison between our model and other methods. Results are reported on our 100-way testing set.

4.4.1 Comparison with other methods

In this experiment, we use image-label pairs from the ILSVRC 2012 CLS dataset as external memory. We use all 1,000 categories for learning. We conduct experiments on 1-shot, 5-shot, 10-shot tasks and compare our results with several algorithms. The results are shown in Table 1.

***k*-NN** and **Exemplar-SVMs**. k-Nearest Neighbors (k-NN) is a simple but effective classifier when very few training examples are provided. We utilize ResNet-200 features and consider two distance metrics, *i.e.*, l_1 and l_2, for pairwise distance calculation. And we set $k = 5$. Exemplar-SVMs (E-SVM) [22] train an SVM for each positive example and ensemble them to obtain the final score. The method was widely used in object detection in the pre-ConvNet era. We use the same ResNet-200 features and set $C = 0.1$. The results show that our method outperforms k-NN for both l_1 and l_2 distance with a large margin and it also outperforms E-SVMs. Note that on 5-shot and 10-shot tasks, our model achieves better performance than the E-SVMs with larger margin. The results show that our model takes advantage of the large number of image-label pairs in the external memory by learning relationships between the examples and the external data.

KV-MemNNs. By utilizing the interpretation of image embedding as `key` and label embedding as `value` as in our model, KV-MemNNs can also be trained to conduct few-shot learning. However, due to the design of KV-MemNNs, few-shot prediction has to rely on the external memory, while the image classification datasets used in our work are too large to be stored in. This property means that KV-MemNNs conduct non-deterministic classification prediction, which is not desirable. It is unrealistic to search over all image-pairs in the external memory during each training iteration. In the testing, it is also time-consuming to traverse the whole external memory. As a workaround, we randomly sample 1,000 pairs from the external memory for matching during both training and testing. We report the mean classification results and the standard deviation in 20 runs. The result shows that our abstraction memory extracts valuable information from the large external memory and is much more compact than the original memory banks.

Matching Networks. We also compare our method with

Methods	5-way 1-shot classification
Matching Networks	90.1
Ours	**93.9**

Table 2. Comparison between our model and Matching Networks on the 5-way 1-shot task.

the recently proposed Matching Networks [33]. Matching Networks use two embedding functions that consider set context. However, as LSTM is used for the embedding, the size of the support set is limited. In [33], the number of categories is usually set to 5 for ImageNet experiments (5-way). For fair comparison, we conduct our experiment on the 5-way 1-shot task. We randomly choose 5 categories from the previously used 100-category set. The testing set has the same number of instances per category. The result is shown in Table 2, which demonstrates that our method outperforms the Matching Network. Our model builds an explicit connection between the few training examples and the external memory, which benefits greatly from a large vocabulary.

We visualize the query results between the external memory and the query in Figure 4.

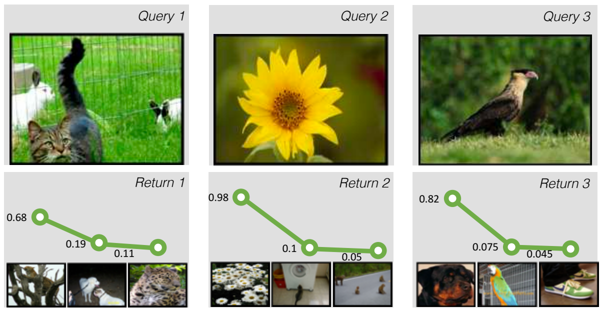


Figure 4. We show the query results returns from the external memory. The scores are the softmax probabilities. Only top-3 results are shown.

4.5. Few-shot Learning with Machine-labeled Annotations

In this experiment, we replace the external memory source with the OpenImages dataset. The machine-labeled images are much easier to obtain but are noisier. We train our model to learn from such noisy web images.

We construct the external memory using the OpenImages dataset. We use four different settings, which are: 1,000 vocabulary with human-labeled images, 1,000 vocabulary with machine-labeled images, 6,000 vocabulary with human-labeled images, and 6,000 vocabulary with machine-labeled images. Note that although the OpenImages dataset is machine-labeled, the validation set in the original dataset is also validated by human raters. The results are shown in Table 3, which demonstrates that machine-labeled external memory can serve as a good source for few-shot learning, which is less accurate than human-labeled external memory by only about 1%

Methods	1,000	6,000
Machine-labeled	66.6	67.4
Human-labeled	67.7	68.2

Table 3. Results on the OpenImages dataset. The results are reported on the 100-way 5-shot task.

As the vocabulary size grows, we observe that performance improves. This shows that given a large vocabulary, our model is able to reason among the external memory in a more effective way. Larger vocabulary will be explored in the future.

4.6. Hyperparamter Study

We conduct the hyperparameter study on the memory slots numbers, *i.e.*, N_1 for the external memory and N_2 for the abstraction memory. Table 4 shows the comparisons among different combinations of memory slots in 5-shot recognition on ImageNet dataset, which demonstrates that our proposed model is robust to the change of memory slots.

$N_1 : N_2$	Accuracy (%)
500 : 500	67.6
1000 : 250	67.9
1000 : 500	68.0
1000 : 1000	67.7
2000 : 500	68.1

Table 4. Comparisons among the numbers of memory slots.

5. Conclusion

In this paper, we propose a novel Memory Networks architecture specifically tailored to tackle the few-shot learning problem on object recognition. By incorporating a novel memory component into the Key-Value Memory Networks, we enable rapid learning from seeing only a handful of positive examples by abstracting and remembering the presented external memory. We utilize LSTM controllers for reading and writing operations into the memory. We demonstrate that our proposed model achieves better performance than other state-of-the-art methods. Furthermore, we obtain similar performance by utilizing machine-labeled annotations compared to human-labeled annotations.

Acknowledgments. Our work is partially supported by the Data to Decisions Cooperative Research Centre (www.d2dcrc.com.au), Google Faculty Award, and an Australian Government Research Training Program Scholarship. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the TITAN X (Pascal) GPU used for this research.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. In *OSDI*. USENIX Association, 2016. 6

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 6

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. 2, 4

[4] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *TNN*, 5(2):157–166, 1994. 4

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 2

[6] X. Chen, A. Shrivastava, and A. Gupta. NEIL: Extracting visual knowledge from web data. In *CVPR*, 2013. 2

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*. 2, 6

[8] S. K. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014. 2

[9] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *TPAMI*, 28(4):594–611, 2006. 3

[10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 6

[11] A. Graves, G. Wayne, and I. Danihelka. Neural Turing machines. *arXiv preprint arXiv:1410.5401*, 2014. 3, 5, 6

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2

[13] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 6

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 4

[15] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *ECCV*, 2016. 2

[16] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016. 2, 6

[17] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015. 4

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[19] I. Krasin, T. Duerig, N. Alldrin, A. Veit, S. Abu-El-Haija, S. Belongie, D. Cai, Z. Feng, V. Ferrari, V. Gomes, A. Gupta, D. Narayanan, C. Sun, G. Chechik, and K. Murphy. OpenImages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from* `https://github.com/openimages`, 2016. 2, 6, 7

[20] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 3

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[22] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *ICCV*, 2011. 2, 7

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013. 2

[24] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. Key-value memory networks for directly reading documents. *EMNLP*, 2016. 2, 3

[25] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *CVPR*, 2016. 4

[26] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. In *ICML*, 2016. 3

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[28] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-to-end memory networks. In *NIPS*, 2015. 3

[29] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014. 4, 6

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1

[31] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence-video to text. In *ICCV*, 2015. 4

[32] O. Vinyals, S. Bengio, and M. Kudlur. Order matters: Sequence to sequence for sets. In *ICLR*, 2016. 3, 5

[33] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *NIPS*, 2016. 2, 3, 7, 8

[34] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015. 4

[35] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015. 3

[36] J. Weston, S. Chopra, and A. Bordes. Memory networks. In *ICLR*, 2015. 2, 3

[37] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 4

[38] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 4

[39] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014. 4