Visual Question Answering with Deep Learning

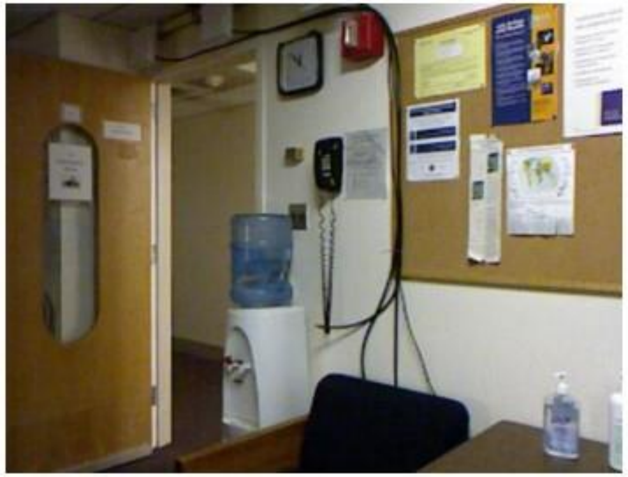
Aahitagni Mukherjee

Shubham Agarwal

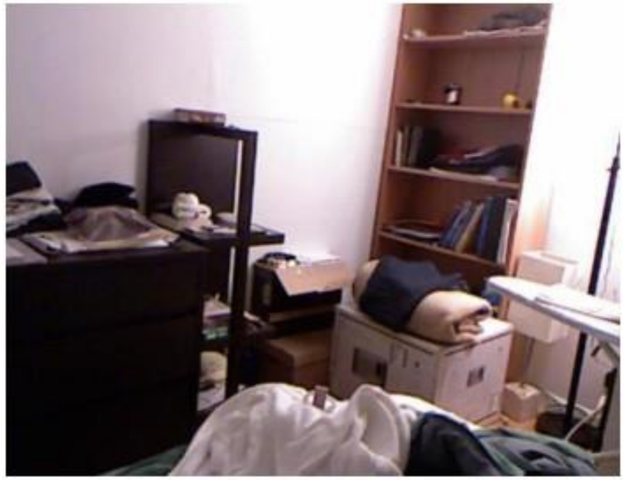
Visual Question Answering

Given an image, and a natural language-like question, find the correct answer to it

- Training on a set of triplets (image, question, answer).
- *Free-form* and *open-ended* questions.
- Answers can be single word or multiple word.



Question: what is the largest blue object in this picture?
Ground truth: water carboy
Proposed CNN: water carboy



Question: what color is the shade of the table lamp close to the bookshelf?
Ground truth: white
Proposed CNN: white

Lin Ma et al 2015

Datasets

- DAQUAR(DAtaset for QUestion Answering on Real-world images) – 1450 images and 12468 questions related to them. On an average 12 words per question.

```
what is on the left side of the white oven on the floor and on right side of the blue armchair in the image1 ?
garbage_bin
what is on the left side of the fire extinguisher and on the right side of the chair in the image1 ?
table
what is between the the two white and black garbage bins in the image1 ?
chair
how many objects are between the fire extinguisher and the white oven on the floor in the image1 ?
3
what is the largest object in this picture in the image1 ?
washing_machine
```

- VQA(Visual Question Answering) dataset – 254,721 images, 764,163 questions, 9,934,119 answers

- Wu-Palmer Similarity Measure(WUPS score) is used for performance evaluation – Script by Malinowski M.

Malinowski et al. 2014
Anton et al. 2015
Wu et al. 1994

Challenges

- The output is to be conditioned on both image and language inputs.
- A better representation of the image content is essential

- Interactions between the two modalities need to appropriately modelled.

Lin Ma et al 2015

Previous Approaches

- Neural-based approach - image representation from a CNN is fed to each hidden layer of a single LSTM. The LSTM then models the concatenation of question and answer.

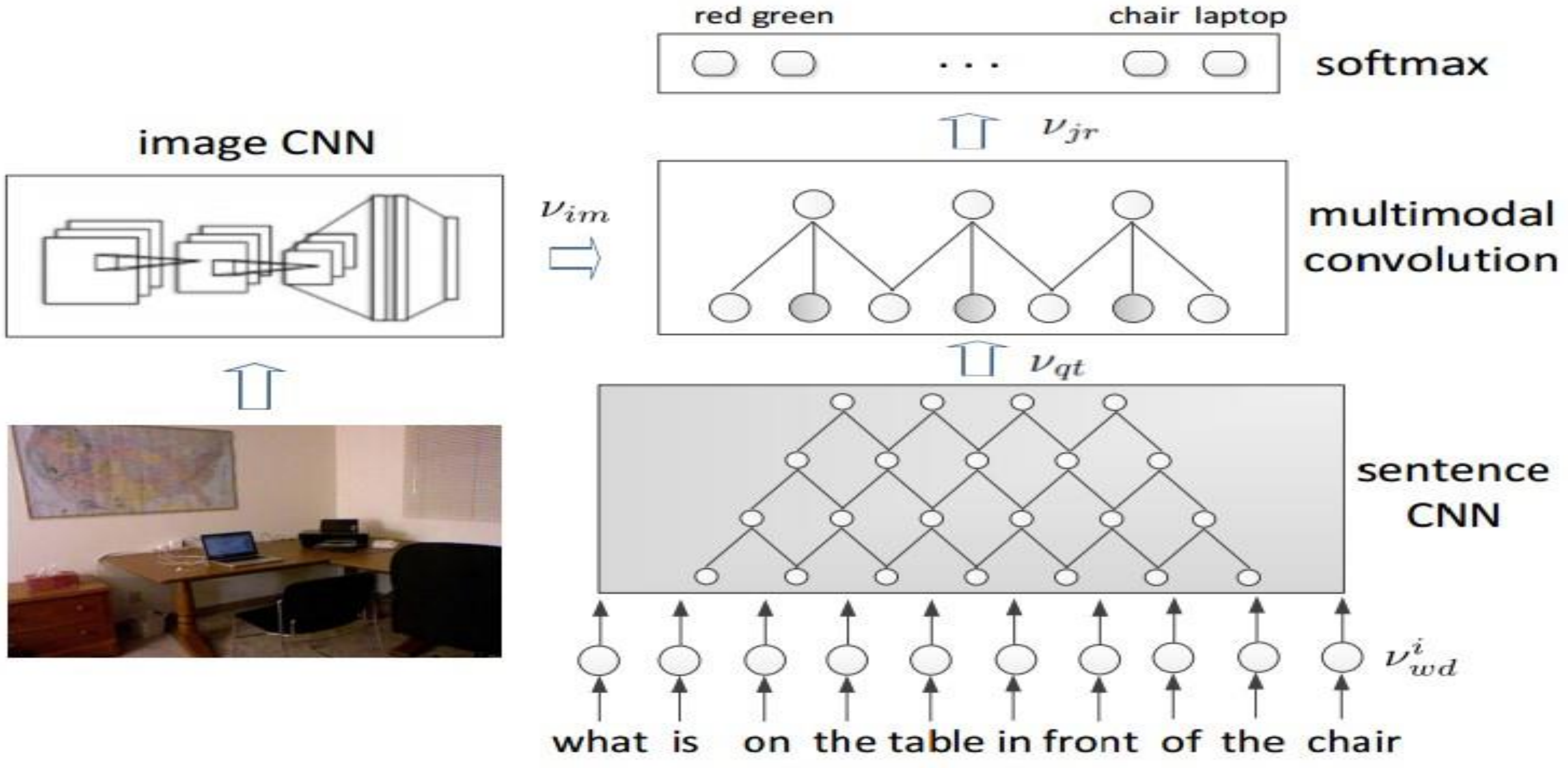
- mQA approach – 4 units - an LSTM to extract the *question representation*, a CNN to extract the *visual representation*, an LSTM for storing the linguistic context in an answer, and a *fusing component* to combine the information from the first three components and generate the answer.

- VIS + LSTM - Here the image is treated as a single word, and the intermediate representation of the input thus obtained is used for classification into the correct class, which is the single word answer.

- CNN approach - uses 3 CNN's - one to extract sentence representation, one for image representation, and the third is a multimodal layer to fuse the two.

Malinowski et al. 2015
Gao et al. 2015
Kiros et al. 2015
Lin Ma et al. 2015

CNN model



Lin Ma et al 2015

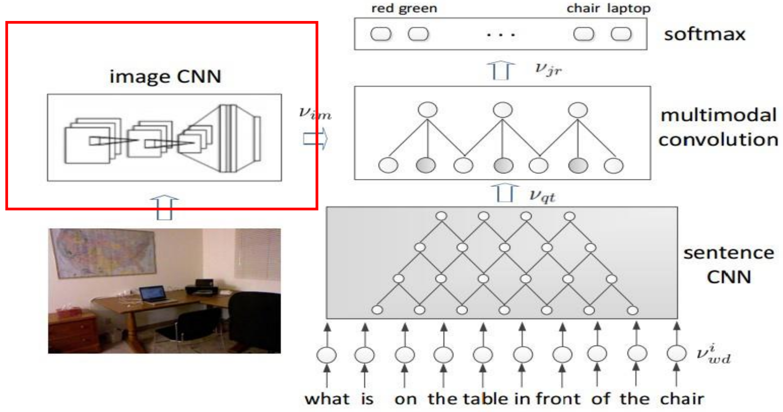


Image CNN

$$\nu_{im} = \sigma(\boldsymbol{w_{im}}(CNN_{im}(I)) + b_{im})$$

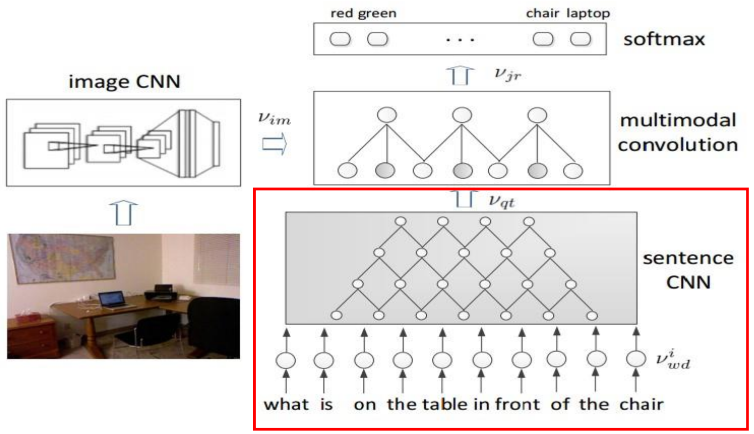
σ: Nonlinear activation function.

$\boldsymbol{w_{im}}|_{d \times 4096}$: Mapping matrix

CNN_{im} takes image as input and outputs a fixed length vector.

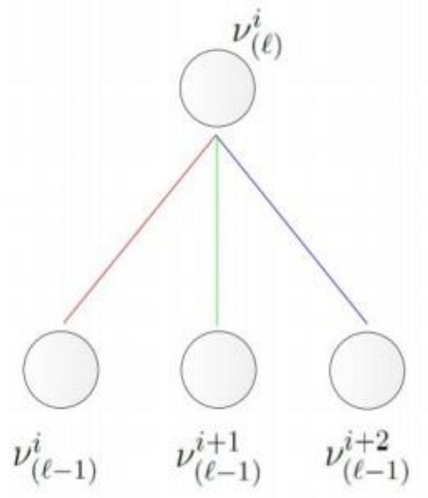
19 weight layers
Input RGB image
conv3-64 conv3-64
Max pool
conv3-128 conv3-128
Max pool
conv3-256 conv3-256 conv3-256 **conv3-256**
Max pool
conv3-512 conv3-512 conv3-512 **conv3-512**
Max pool
conv3-512 conv3-512 conv3-512 **conv3-512**
Max pool
FC - 4096
FC - 4096
FC - 1000
Softmax

Simoyan et al. 2015



Sentence CNN

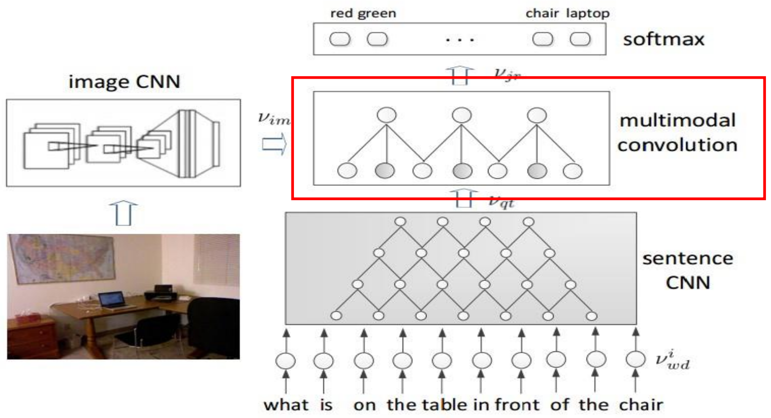
1. For sequential input σ, convolution unit for feature map of type f on the l^{th} layer is $\nu^i_{(l,f)} \stackrel{\text{def}}{=} \sigma(\boldsymbol{w}_{(l,f)}\, \vec{\nu}^i_{(l-1)} + b_{(l,f)})$
2. $\vec{\nu}^i_{(l-1)} \stackrel{\text{def}}{=} \nu^i_{(l-1)} \,||\, \nu^{i+1}_{(l-1)} \,||\, \nu^{i+2}_{l-1}$
3. $\vec{\nu}^i_{(0)} \stackrel{\text{def}}{=} \nu^i_{wd} \,||\, \nu^{i+1}_{wd} \,||\, \nu^{i+2}_{wd}$
4. Max-pooling after each convolution $\nu^i_{(l+1,f)} = max(\nu^{2i}_{(l,f)}, \nu^{2i+1}_{(l,f)})$

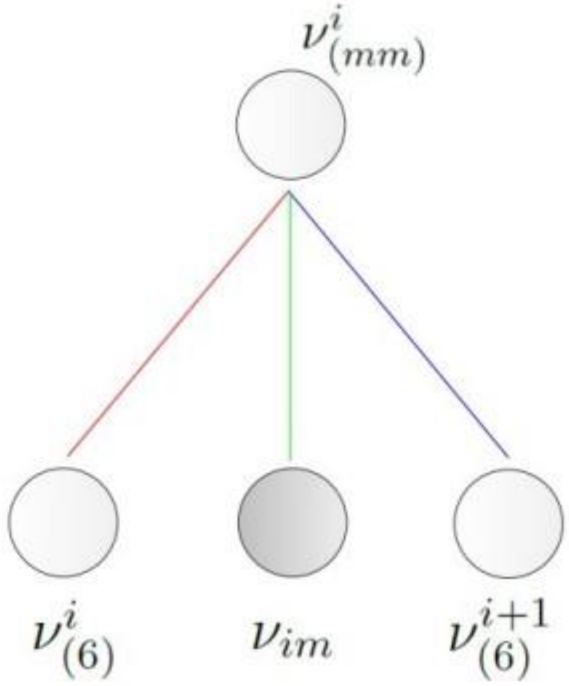


ν^i_{wd} : Skip-gram word embedding of i-th question word

Lin Ma et al 2015

Multimodal Convolutional Layer





input: $\nu_{qt} = [\nu^0_{(6)} ... \nu^n_{(6)}]$

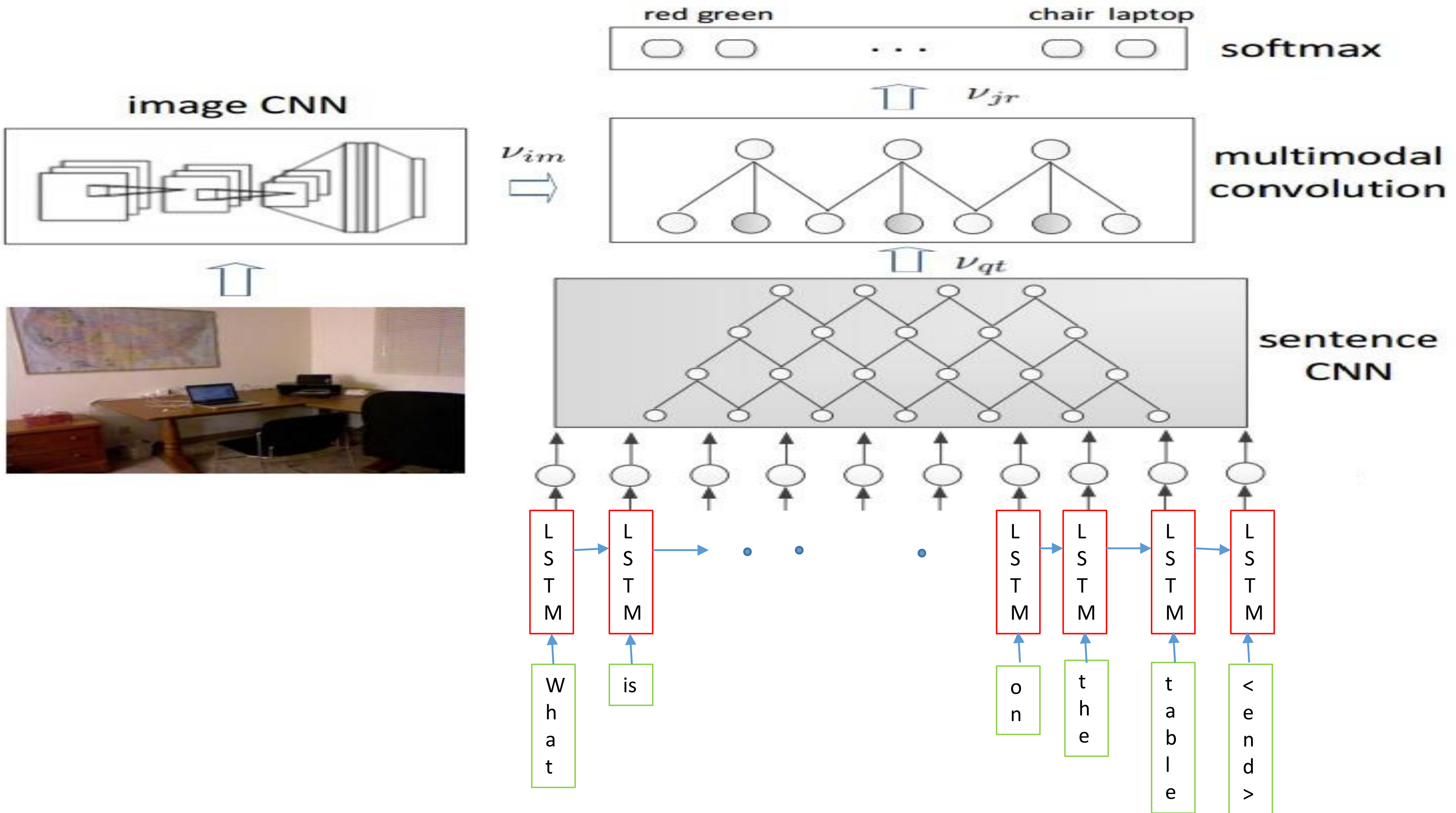
Capturing the interaction between two multimodal inputs

$$\vec{\nu}^{\,i}_6 = \nu^i_6 \,||\, \nu_{im} \,||\, \nu^{i+1}_6$$

$$\nu^i_{(mm,f)} = \sigma(\boldsymbol{w}_{(mm,f)} \vec{\nu}^{\,i}_{(6)} + b_{(mm,f)})$$

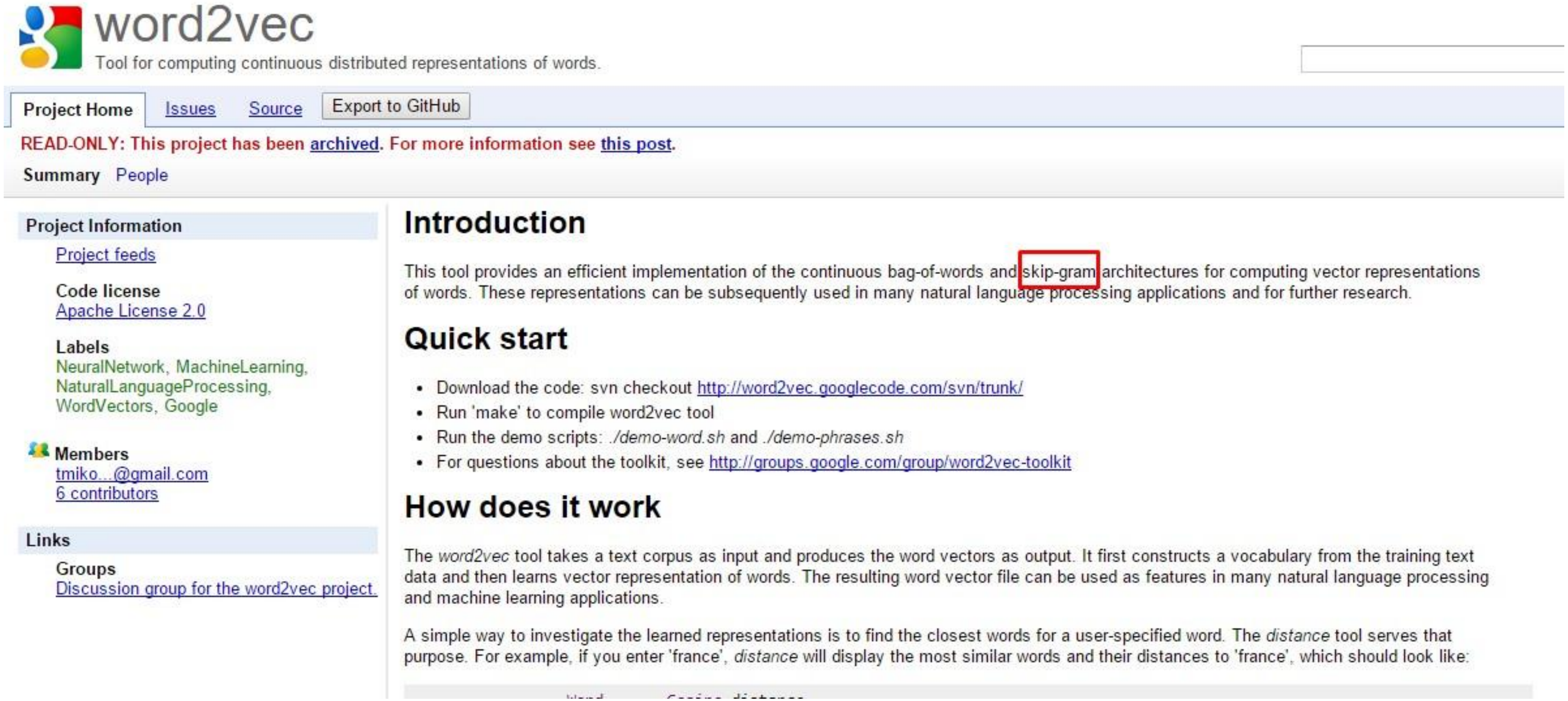
Lin Ma et al 2015

Proposed Modification



Input to LSTM

Skip-gram word embeddings from the question sentence



word2vec

Tool for computing continuous distributed representations of words.

Project Home | Issues | Source | Export to GitHub

READ-ONLY: This project has been archived. For more information see this post.

Summary | People

Project Information

Project feeds

Code license
Apache License 2.0

Labels
NeuralNetwork, MachineLearning, NaturalLanguageProcessing, WordVectors, Google

Members
tmiko...@gmail.com
6 contributors

Links

Groups
Discussion group for the word2vec project.

Introduction

This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.

Quick start

- Download the code: svn checkout http://word2vec.googlecode.com/svn/trunk/
- Run 'make' to compile word2vec tool
- Run the demo scripts: *./demo-word.sh* and *./demo-phrases.sh*
- For questions about the toolkit, see http://groups.google.com/group/word2vec-toolkit

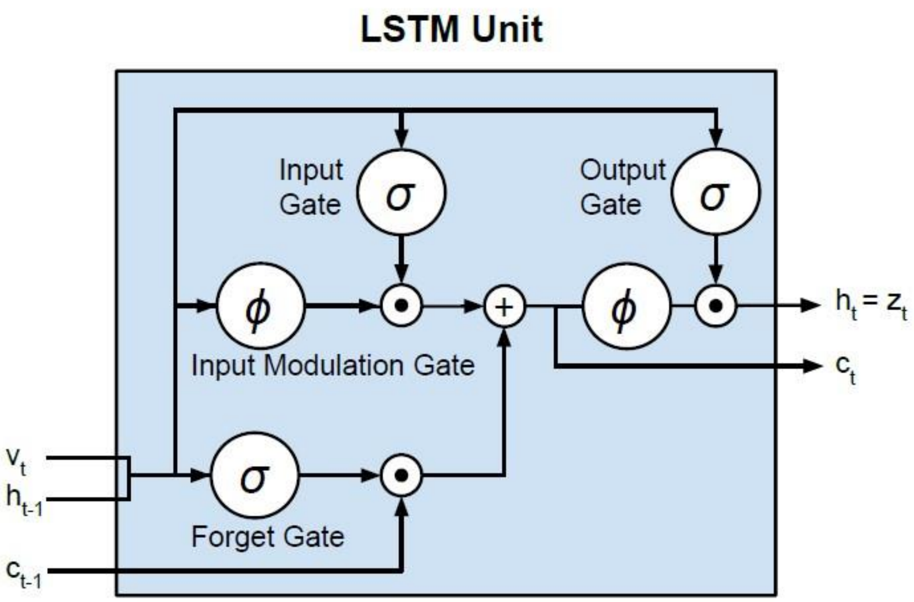
How does it work

The *word2vec* tool takes a text corpus as input and produces the word vectors as output. It first constructs a vocabulary from the training text data and then learns vector representation of words. The resulting word vector file can be used as features in many natural language processing and machine learning applications.

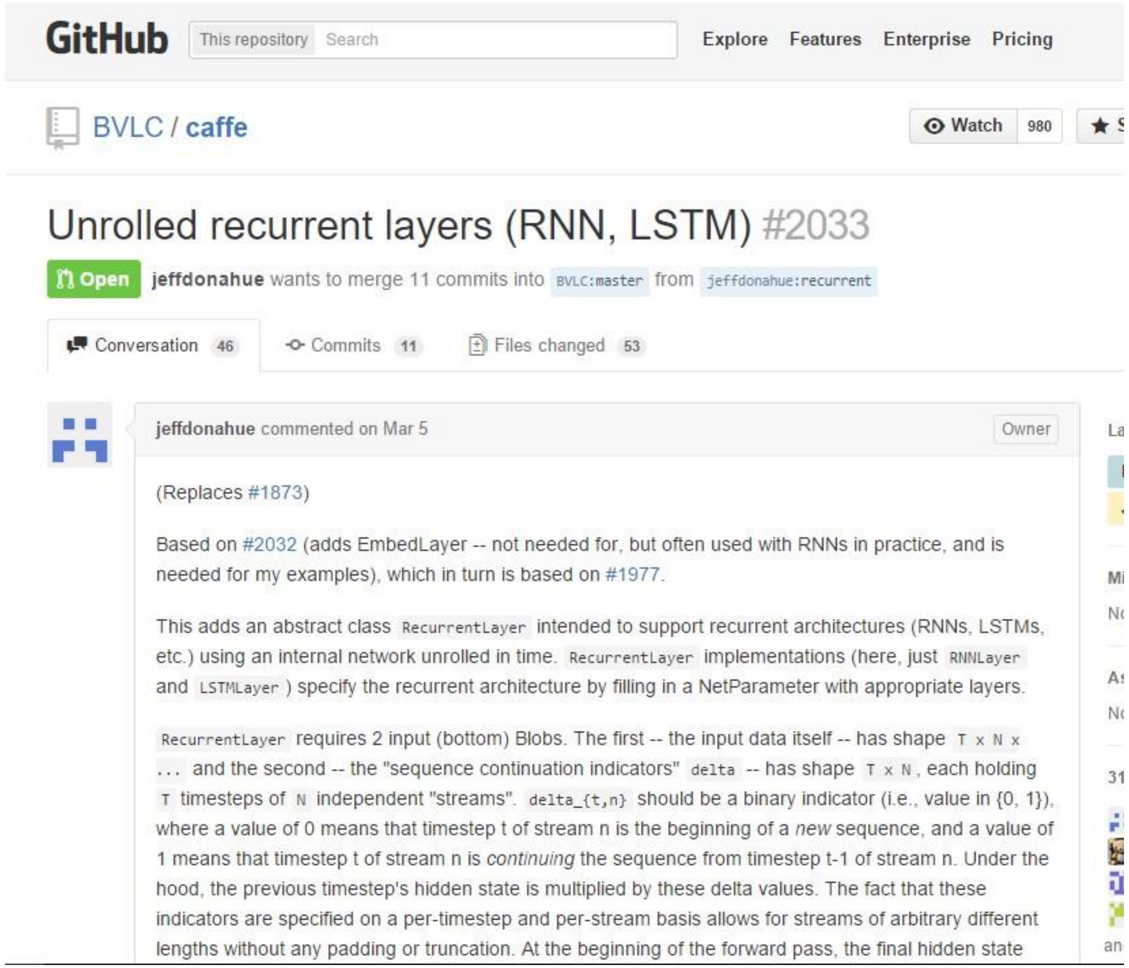
A simple way to investigate the learned representations is to find the closest words for a user-specified word. The *distance* tool serves that purpose. For example, if you enter 'france', *distance* will display the most similar words and their distances to 'france', which should look like:

Mikolov et al. 2013
https://code.google.com/p/word2vec/

LSTM



$$
\begin{aligned}
i_t &= \sigma(W_{vi}v_t + W_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{vf}v_t + W_{hf}h_{t-1} + b_f) \\
o_t &= \sigma(W_{vo}v_t + W_{ho}h_{t-1} + b_o) \\
g_t &= \phi(W_{vg}v_t + W_{hg}h_{t-1} + b_g) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \phi(c_t)
\end{aligned}
$$


GitHub This repository Search Explore Features Enterprise Pricing

BVLC / caffe Watch 980

Unrolled recurrent layers (RNN, LSTM) #2033

Open jeffdonahue wants to merge 11 commits into `BVLC:master` from `jeffdonahue:recurrent`

Conversation 46 Commits 11 Files changed 53

jeffdonahue commented on Mar 5 Owner

(Replaces #1873)

Based on #2032 (adds EmbedLayer -- not needed for, but often used with RNNs in practice, and is needed for my examples), which in turn is based on #1977.

This adds an abstract class `RecurrentLayer` intended to support recurrent architectures (RNNs, LSTMs, etc.) using an internal network unrolled in time. `RecurrentLayer` implementations (here, just `RNNLayer` and `LSTMLayer`) specify the recurrent architecture by filling in a NetParameter with appropriate layers.

`RecurrentLayer` requires 2 input (bottom) Blobs. The first -- the input data itself -- has shape `T x N x ...` and the second -- the "sequence continuation indicators" `delta` -- has shape `T x N`, each holding `T` timesteps of `N` independent "streams". `delta_{t,n}` should be a binary indicator (i.e., value in {0, 1}), where a value of 0 means that timestep t of stream n is the beginning of a *new* sequence, and a value of 1 means that timestep t of stream n is *continuing* the sequence from timestep t-1 of stream n. Under the hood, the previous timestep's hidden state is multiplied by these delta values. The fact that these indicators are specified on a per-timestep and per-stream basis allows for streams of arbitrary different lengths without any padding or truncation. At the beginning of the forward pass, the final hidden state

Malinowski et al. 2015
https://github.com/junhyukoh/caffe-lstm

References

1. [Lin Ma et al. 2015] L. Ma and Z. Lu and H. Li. Learning to Answer Questions From Image using Convolutional Neural Network 2015. arXiv:1506.00333.
2. [Malinowski et al. 2014] M. Malinowski and M. Fritz 2014b. Towards a Visual Turing Challenge. arXiv:1410.8027.
3. [Anton et al. 2015] S. Antol and A. Agrawal and J. Lu and M. Mitchell and D. Batra and C. Lawrence Zitnick and D. Parikh 2015. VQA: Visual Question Answering. arXiv:1505.00468.
4. [Wu et al. 1994] Z. Wu and M. Palmer 1994. Verb Semantics and Lexical Selection. ACL 1994.
5. [Gao et al. 2015] H. Gao and J. Mao and J. Zhou and Z. Huang and L. Wang and W. Xu 2015. Are You Talking to a Machine? Dataset and Methods for Multilingual Image Question Answering. arXiv:1505.05612.
6. [Kiros et al. 2015] M Ren, R. Kiros, R. Zemel. Exploring Models and Data for Image Question Answering 2015. arXiv:1505.02074.
7. [Simonyan et al. 2015] K. Simonyan and A. Zisserman 2014. Very Deep Convolutional Networks for Large-scale Image Recognition. arXiv 1409.1556.
8. [Mikolov et al. 2013] T. Mikolov and K. Chen and G. Corrado and J. Dean 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781.
9. [Malinowski et al. 2015] M. Malinowski and M. Rohrbach and M. Fritz 2015b. Ask Your Neurons: A Neural-based Approach to Answering Questions about Images. arXiv 1505.01121.

CONVOLUTIONAL, LONG SHORT-TERM MEMORY, FULLY CONNECTED DEEP NEURAL NETWORKS

Tara N. Sainath, Oriol Vinyals, Andrew Senior, Haşim Sak

Google, Inc., New York, NY, USA
{tsainath, vinyals, andrewsenior, hasim}@google.com

ABSTRACT

Both Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) have shown improvements over Deep Neural Networks (DNNs) across a wide variety of speech recognition tasks. CNNs, LSTMs and DNNs are complementary in their modeling capabilities, as CNNs are good at reducing frequency variations, LSTMs are good at temporal modeling, and DNNs are appropriate for mapping features to a more separable space. In this paper, we take advantage of the complementarity of CNNs, LSTMs and DNNs by combining them into one unified architecture. We explore the proposed architecture, which we call CLDNN, on a variety of large vocabulary tasks, varying from 200 to 2,000 hours. We find that the CLDNN provides a 4-6% relative improvement in WER over an LSTM, the strongest of the three individual models.

nonlinear hidden layer. If factors of variation in the hidden states could be reduced, then the hidden state of the model could summarize the history of previous inputs more efficiently. In turn, this could make the output easier to predict. Reducing variation in the hidden states can be modeled by having DNN layers after the LSTM layers. This is similar in spirit to the hidden to output model proposed in [4], and also tested for speech, though with RNNs [8].

The model we propose is to feed input features, surrounded by temporal context, into a few CNN layers to reduce spectral variation. The output of the CNN layer is then fed into a few LSTM layers to reduce temporal variations. Then, the output of the last LSTM layer is fed to a few fully connected DNN layers, which transform the features into a space that makes that output easier to classify.

Combining CNN, LSTMs and DNNs has been explored in [9]. However, in that paper the three models were first trained separately and then the three outputs were combined through a combination

Method	WER
LSTM	18.0
CNN+LSTM	17.6
LSTM+DNN	17.6
CLDNN	**17.3**

Table 5. WER, CLDNN

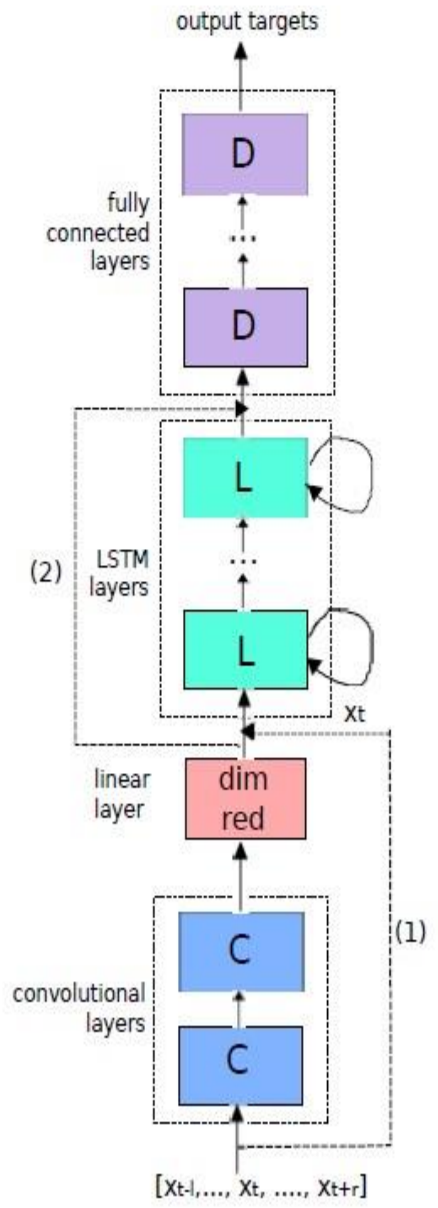


Fig. 1. CLDNN Architecture