CSE 5194.01 Deep Learning - Applications

VISUAL QUESTION ANSWERING

September 24, 2017

Akshay Mehra, Christopher Menart, Debanjan Nandi, Rohit Kumar Srivastava
The Ohio State University
Computer Science and Engineering

{*mehra*.42, *menart*.1, *nandi*.20, *srivastava*.141} @*osu.edu*

1 Overview

Our Visual QA Project attempted to run and modify a Keras implementation [4] of the original baseline model for Visual Question Answering [3]. Our model uses Convolutional Neural Network (CNN) for image recognition and a Neural Language Model for modeling questions. Embedding features for the image and the questions are combined using point-wise multiplication and processed together by a Multi Layer Perceptron. Our best model uses GloVe vectors to generate embeddings for each word in the question and then uses a 1D Convolutional Neural Network to generate embeddings for each question. The standard Resnet 152 model [5], trained on Imagenet [7], is used to generate image features. This model achieves an accuracy of about 61% compared to the baseline accuracy of 58%.

2 Baseline

The baseline model uses VGGNet[6] features to describe images. VGG vectors are embedded for the task at hand using 2 LSTM layers with Dropout. Image and question features are combined using point-wise multiplication and passed through a single fully-connected layer followed by a softmax. The original model achieves a top-1 accuracy of about 58% when trained for 10 epochs. Since the top result on the 2017 VQA leaderboard has an overall accuracy of 70.92%, this is still a reasonable starting point.

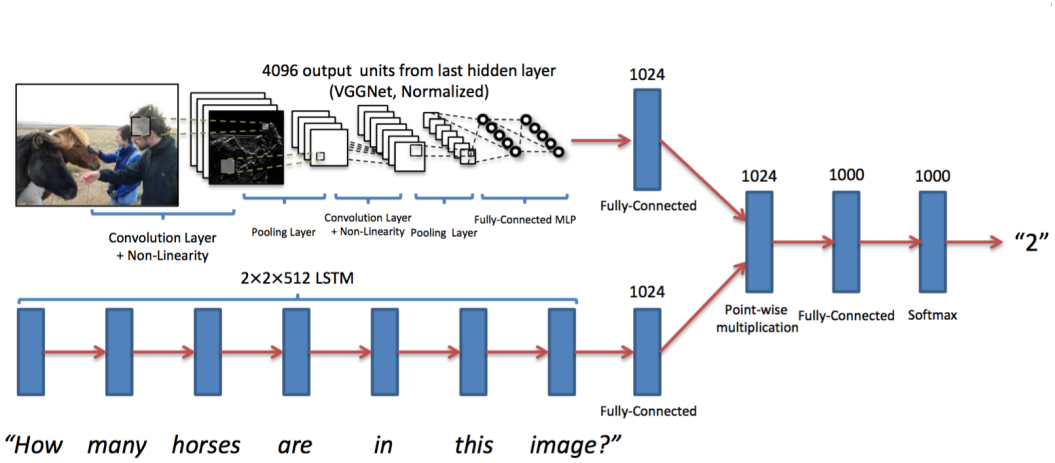

Figure 1: Baseline Model [3]

3 Methods

We experiment with both the vision (image) and language (questions) portions of our model, retaining the overall structure of a model that merges both branches with point-wise multiplication before culminating in softmax confidence over the 1000 most likely outputs. We describe the different components of our model below:

3.1 Image Channel

This channel provides an embedding for the image. We experiment with two embeddings:

1. VGGNet[6]: The activations from the last hidden layer of the VGGNet trained on imagenet are used as 4096-dim image embedding, and this output is passed into a fully connected layer having 1024 units. We obtained pre-computed features from the baseline implementation.

2. ResNet[5]: The activations from the last hidden layer are used as 2048-dim image embedding. This output is passed into a fully connected layer having 1024 units. In the interest of a direct comparison between vision models, we resized all images to a fixed size (224x224) and used the pre-trained ResNet model without any additional training. The features from the final convolutional layer were saved as image descriptors.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2: A tabular description of ResNet [5], which is too large to display graphically. We use the 152-layer version of ResNet, which obtained the best results on the network's original vision tasks, and take the output of the final 'average pool' as our descriptor.

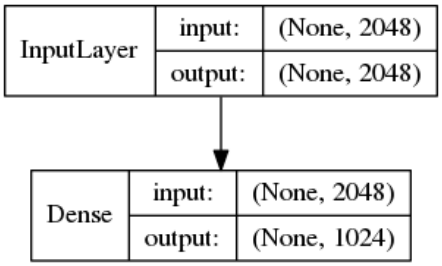

Figure 3: The trained component of our Image Model

3.2 Question Channel

This channel provides an embedding for the question. We experiment with three different embeddings:

1. LSTM with dropout(LSTM-Q): We use GloVe embeddings for questions. This is passed through 2 LSTMs having 512 units each followed a dropout layer. The output is passed through a dense layer with 1024 units and *tanh* activation

2. LSTM Birectional(BLSTM-Q):We use GloVe embeddings for questions. This is passed through 2 LSTMs having 512. The output is passed through a dense layer with 1024 units and *tanh* activation.

3. 1D Convolution for Questions(1DCNN-Q): We use GloVe embeddings for questions. This is passed through 1D convolution layer with 512 filters and stride of 2. This is followed by 1D MaxPooling and flattening. The output is passed through a dense layer with 1024 units and *tanh* activation.

3.3 Multi-Layer Perceptron (MLP)

The image and question embeddings are combined via point-wise multiplication to obtain a single embedding. These combined embeddings are passed through as dense layer with 1024 units and *tanh* activation. This is followed by a dropout and a softmax layer with 1000 nodes. The entire model is learned with cross-entropy loss with RMSProp optimizer. For training, we use 215359 training samples and report the performance using the answer that has the highest activation from potential multiple choice answers in the validation set which has 121512 samples.

1. **VGGNet + LSTM-Q**: This is the baseline model. The model achieves an accuracy of 59% when trained for 20 epochs. Each epoch takes around 65 seconds since the model has 2 LSTM layers.

2. **VGGNet +BLSTM-Q**: The model achieves slight increase in performance using a Bi-directional LSTM. However, the model training becomes very slow. We get a performance of 59.31% after training for 20 epochs.

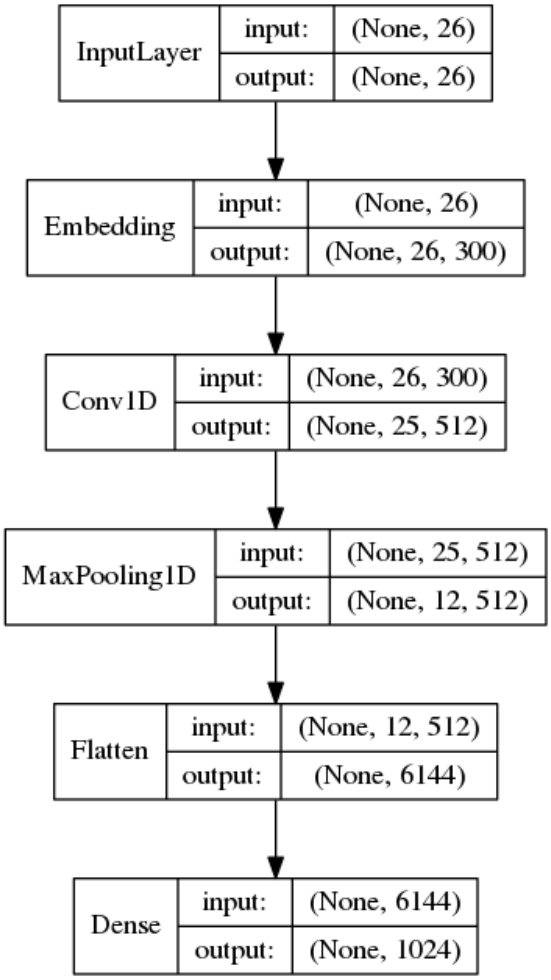

Figure 4: Questions Model

3. **VGGNet + 1DCNN-Q**: This model is extremely fast during training and outperforms the LSTM based models. This model achieves a performance of 59.5% after 20 epochs.

4. **ResNet + LSTM-Q**: In this model we replace VGGnet features with ones obtained from Resnet Model. The Resnet model performs better than the VGGNet.

5. **ResNet + BLSTM-Q**: The Bidirectional LSTM helps to improve the performance slightly. But we see that the training time increases by a lot.

6. **ResNet + 1DCNN-Q**: This is our best performing model. It combines the power of Resnet with a 1D-Convolutional Neural Network for Language. This 1D-CNN works at a bigram level by making strides of 2. We achieve an accuracy of 61% and a training time of 12 seconds per epoch.

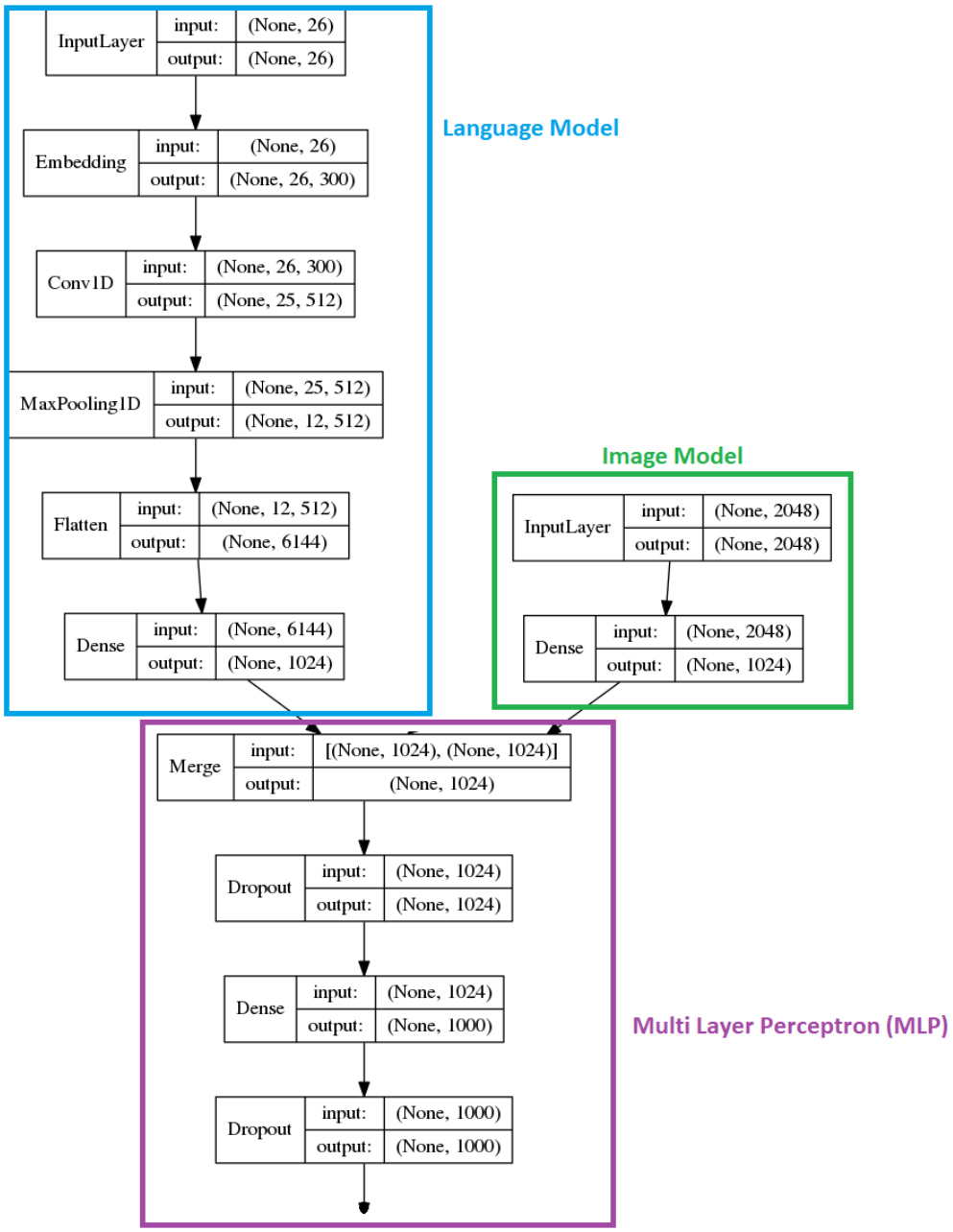

Figure 5: Final Model

Table 1: Results

Model	**Observation**	
	Accuracy(%)	**Training Time:1 Epoc (Seconds)**
VGGNet+LSTM-Q	58.9	65
VGGNet+BLSTM-Q	59.31	125
VGGNet+1DCCN-Q	59.5	10
ResNet+BLSTM-Q	60.55	150
ResNet+LSTM-Q	60.06	85
ResNet+1DCNN-Q	61.41	12

4 Hardware Used

CPU: Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz, 28 Cores, 126GB Memory (OSC)[1]
GPU: Nvidia Tesla P100-PCIE @ 1328Mhz, 16GB Memory (OSC)[1]

5 Conclusion

The VQA model we modified was clearly a simple setup with room for improvement. Even more sophisticated techniques could have been applied, such as end-to-end training or increasing the amount of processing performed by the model after merging verbal and visual information. Far better performance is likely possible with more sophisticated models (such as Kelvin Xu's "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention"[2], which integrate visual information during recurrent processing). However, we successfully experimented with replacing VGG with Resnet and using a Convolutional Neural Network for Language.

References

[1] Ohio Supercomputer Center. 1987. Ohio Supercomputer Center. Columbus OH: Ohio Supercomputer Center. `http://osc.edu/ark:/19495/f5s1ph73`

[2] Kazemi, Vahid, and Ali Elqursh. "Show, Ask, Attend, and Answer: A Strong Baseline For Visual Question Answering." *arXiv preprint arXiv:1704.03162* (2017).

[3] Antol, Stanislaw, et al. "Vqa: Visual question answering." *Proceedings of the IEEE International Conference on Computer Vision.* 2015.

[4] Anant Gupta, VQA-Keras-Visual-Question-Answering, (2016), GitHub repository, `https://github.com/anantzoid/VQA-Keras-Visual-Question-Answering`

[5] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016.

[6] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[7] Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai and Fei-Fei, Li. "ImageNet: A Large-Scale Hierarchical Image Database" In *CVPR* 2009.