HADAMARD PRODUCT FOR LOW-RANK BILINEAR POOLING

Jin-Hwa Kim
Interdisciplinary Program in Cognitive Science
Seoul National University
Seoul 08826, Republic of Korea
jhkim@bi.snu.ac.kr

Kyoung-Woon On
School of Computer Science and Engineering
Seoul National University
Seoul 08826, Republic of Korea
kwon@bi.snu.ac.kr

Woosang Lim
School of Computing, KAIST
Daejeon 34141, Republic of Korea
quasar17@kaist.ac.kr

Jeonghee Kim & Jung-Woo Ha
NAVER LABS Corp. & NAVER Corp.
Gyeonggi-do 13561, Republic of Korea
{jeonghee.kim,jungwoo.ha}@navercorp.com

Byoung-Tak Zhang
School of Computer Science and Engineering & Interdisciplinary Program in Cognitive Science
Seoul National University & Surromind Robotics
Seoul 08826, Republic of Korea
btzhang@bi.snu.ac.kr

ABSTRACT

Bilinear models provide rich representations compared with linear models. They have been applied in various visual tasks, such as object recognition, segmentation, and visual question-answering, to get state-of-the-art performances taking advantage of the expanded representations. However, bilinear representations tend to be high-dimensional, limiting the applicability to computationally complex tasks. We propose low-rank bilinear pooling using Hadamard product for an efficient attention mechanism of multimodal learning. We show that our model outperforms compact bilinear pooling in visual question-answering tasks with the state-of-the-art results on the VQA dataset, having a better parsimonious property.

1 INTRODUCTION

Bilinear models (Tenenbaum & Freeman, 2000) provide richer representations than linear models. To exploit this advantage, fully-connected layers in neural networks can be replaced with bilinear pooling. The outer product of two vectors (or Kroneker product for matrices) is involved in bilinear pooling, as a result of this, all pairwise interactions among given features are considered. Recently, a successful application of this technique is used for fine-grained visual recognition (Lin et al., 2015).

However, bilinear pooling produces a high-dimensional feature of quadratic expansion, which may constrain a model structure and computational resources. For example, an outer product of two feature vectors, both of which have 1K-dimensionality, produces a million-dimensional feature vector. Therefore, for classification problems, the choice of the number of target classes is severely constrained, because the number of parameters for a standard linear classifier is determined by multiplication of the size of the high-dimensional feature vector and the number of target classes.

Compact bilinear pooling (Gao et al., 2016) reduces the quadratic expansion of dimensionality by two orders of magnitude, retaining the performance of the full bilinear pooling. This approximation uses sampling-based computation, Tensor Sketch Projection (Charikar et al., 2002; Pham & Pagh, 2013), which utilizes an useful property that $\Psi(x \otimes y, h, s) = \Psi(x, h, s) * \Psi(y, h, s)$, which means the projection of outer product of two vectors is the convolution of two projected vectors. Here, Ψ is the proposed projection function, and, h and s are randomly sampled parameters by the algorithm.

Nevertheless, compact bilinear pooling embraces two shortcomings. One comes from the sampling approach. Compact bilinear pooling relies on a favorable property, $E[\langle \Psi(x,h,s), \Psi(y,h,s) \rangle] = \langle x, y \rangle$, which provides a basis to use projected features instead of original features. Yet, calculating the exact expectation is computationally intractable, so, the random parameters, h and s are fixed during training and evaluation. This practical choice leads to the second. The projected dimension of compact bilinear pooling should be large enough to minimize the bias from the fixed parameters. Practical choices are 10K and 16K for 512 and 4096-dimensional inputs, respectively (Gao et al., 2016; Fukui et al., 2016). Though, these *compacted* dimensions are reduced ones by two orders of magnitude compared with full bilinear pooling, such high-dimensional features could be a bottleneck for computationally complex models.

We propose low-rank bilinear pooling using Hadamard product (element-wise multiplication), which is commonly used in various scientific computing frameworks as one of tensor operations. The proposed method factors a three-dimensional weight tensor for bilinear pooling into three two-dimensional weight matrices, which enforces the rank of the weight tensor to be low-rank. As a result, two input feature vectors linearly projected by two weight matrices, respectively, are computed by Hadamard product, then, followed by a linear projection using the third weight matrix. For example, the projected vector $\mathbf{z}$ is represented by $\mathbf{W}_z^T(\mathbf{W}_\mathbf{x}^T\mathbf{x} \circ \mathbf{W}_\mathbf{y}^T\mathbf{y})$, where $\circ$ denotes Hadamard product.

We also explore to add non-linearity using non-linear activation functions into the low-rank bilinear pooling, and shortcut connections inspired by deep residual learning (He et al., 2016). Then, we show that it becomes a simple baseline model (Antol et al., 2015) or one-learning block of Multimodal Residual Networks (Kim et al., 2016b) as a low-rank bilinear model, yet, this interpretation has not be done.

Our contributions are as follows: First, we propose low-rank bilinear pooling to approximate full bilinear pooling to substitute compact bilinear pooling. Second, Multimodal Low-rank Bilinear Attention Networks (MLB) having an efficient attention mechanism using low-rank bilinear pooling is proposed for visual question-answering tasks. MLB achieves a new state-of-the-art performance, and has a better parsimonious property. Finally, ablation studies to explore alternative choices, *e.g.* network depth, non-linear functions, and shortcut connections, are conducted.

2 Low-rank Bilinear Model

Bilinear models use a quadratic expansion of linear transformation considering every pair of features.

$$f_i = \sum_{j=1}^{N}\sum_{k=1}^{M} w_{ijk} x_j y_k + b_i = \mathbf{x}^T \mathbf{W}_i \mathbf{y} + b_i \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are input vectors, $\mathbf{W}_i \in \mathbb{R}^{N \times M}$ is a weight matrix for the output f_i, and b_i is a bias for the output f_i. Notice that the number of parameters is $L \times (N \times M + 1)$ including a bias vector $\mathbf{b}$, where L is the number of output features.

Pirsiavash et al. (2009) suggest a low-rank bilinear method to reduce the rank of the weight matrix $\mathbf{W}_i$ to have less number of parameters for regularization. They rewrite the weight matrix as $\mathbf{W}_i = \mathbf{U}_i\mathbf{V}_i^T$ where $\mathbf{U}_i \in \mathbb{R}^{N \times d}$ and $\mathbf{V}_i \in \mathbb{R}^{M \times d}$, which imposes a restriction on the rank of $\mathbf{W}_i$ to be at most $d \leq \min(N, M)$.

Based on this idea, f_i can be rewritten as follows:

$$f_i = \mathbf{x}^T \mathbf{W}_i \mathbf{y} + b_i = \mathbf{x}^T \mathbf{U}_i \mathbf{V}_i^T \mathbf{y} + b_i = \mathbb{1}^T (\mathbf{U}_i^T \mathbf{x} \circ \mathbf{V}_i^T \mathbf{y}) + b_i \tag{2}$$

where $\mathbb{1} \in \mathbb{R}^d$ denotes a column vector of ones, and $\circ$ denotes Hadamard product. Still, we need two third-order tensors, $\mathbf{U}$ and $\mathbf{V}$, for a feature vector $\mathbf{f}$, whose elements are $\{f_i\}$. To reduce the order of the weight tensors by one, we replace $\mathbb{1}$ with $\mathbf{P} \in \mathbb{R}^{d \times c}$ and b_i with $\mathbf{b} \in \mathbb{R}^c$, then, redefine as $\mathbf{U} \in \mathbb{R}^{N \times d}$ and $\mathbf{V} \in \mathbb{R}^{M \times d}$ to get a projected feature vector $\mathbf{f} \in \mathbb{R}^c$. Then, we get:

$$\mathbf{f} = \mathbf{P}^T (\mathbf{U}^T \mathbf{x} \circ \mathbf{V}^T \mathbf{y}) + \mathbf{b} \tag{3}$$

where d and c are hyperparameters to decide the dimension of joint embeddings and the output dimension of low-rank bilinear models, respectively.

3 LOW-RANK BILINEAR POOLING

A low-rank bilinear model in Equation 3 can be implemented using two linear mappings without biases for embedding two input vectors, Hadamard product to learn joint representations in a multiplicative way, and a linear mapping with a bias to project the joint representations into an output vector for a given output dimension. Then, we use this structure as a pooling method for deep neural networks. Now, we discuss possible variations of low-rank bilinear pooling based on this model inspired by studies of neural networks.

3.1 FULL MODEL

In Equation 3, linear projections, U and V, can have their own bias vectors. As a result, linear models for each input vectors, $\mathbf{x}$ and $\mathbf{y}$, are integrated in an additive form, called as *full model* for linear regression in statistics:

$$\begin{aligned}\mathbf{f} &= \mathbf{P}^T\big((\mathbf{U}^T\mathbf{x} + \mathbf{b}_x) \circ (\mathbf{V}^T\mathbf{y} + \mathbf{b}_y)\big) + \mathbf{b} \\ &= \mathbf{P}^T(\mathbf{U}^T\mathbf{x} \circ \mathbf{V}^T\mathbf{y} + \mathbf{U}'^T\mathbf{x} + \mathbf{V}'^T\mathbf{y}) + \mathbf{b}'. \end{aligned} \tag{4}$$

Here, $\mathbf{U}'^T = \text{diag}(\mathbf{b}_y) \cdot \mathbf{U}^T$, $\mathbf{V}'^T = \text{diag}(\mathbf{b}_x) \cdot \mathbf{V}^T$, and $\mathbf{b}' = \mathbf{b} + \mathbf{P}^T(\mathbf{b}_x \circ \mathbf{b}_y)$.

3.2 NONLINEAR ACTIVATION

Applying non-linear activation functions may help to increase representative capacity of model. The first candidate is to apply non-linear activation functions right after linear mappings for input vectors.

$$\mathbf{f} = \mathbf{P}^T\big(\sigma(\mathbf{U}^T\mathbf{x}) \circ \sigma(\mathbf{V}^T\mathbf{y})\big) + \mathbf{b} \tag{5}$$

where σ denotes an arbitrary non-linear activation function, which maps any real values into a finite interval, *e.g.* sigmoid or $\tanh$. If two inputs come from different modalities, statistics of two inputs may be quite different from each other, which may result an interference. Since the gradient with respect to each input is directly dependent on the other input in Hadamard product of two inputs.

Additional applying an activation function after the Hadamard product is not appropriate, since activation functions doubly appear in calculating gradients. However, applying the activation function only after the Hadamard product would be alternative choice (We explore this option in Section 5) as follows:

$$\mathbf{f} = \mathbf{P}^T\sigma\big(\mathbf{U}^T\mathbf{x} \circ \mathbf{V}^T\mathbf{y}\big) + \mathbf{b}. \tag{6}$$

Note that using the activation function in low-rank bilinear pooling can be found in an implementation of simple baseline for the VQA dataset (Antol et al., 2015) without an interpretation of low-rank bilinear pooling. However, notably, Wu et al. (2016c) studied learning behavior of multiplicative integration in RNNs with discussions and empirical evidences.

3.3 SHORTCUT CONNECTION

When we apply two previous techniques, full model and non-linear activation, linear models of two inputs are nested by the non-linear activation functions. To avoid this unfortunate situation, we add shortcut connections as explored in residual learning (He et al., 2016).

$$\mathbf{f} = \mathbf{P}^T\big(\sigma(\mathbf{U}^T\mathbf{x}) \circ \sigma(\mathbf{V}^T\mathbf{y})\big) + h_x(\mathbf{x}) + h_y(\mathbf{y}) + \mathbf{b} \tag{7}$$

where h_x and h_y are shortcut mappings. For linear projection, the shortcut mappings are linear mappings. Notice that this formulation is a generalized form of the one-block layered MRN (Kim et al., 2016b). Though, the shortcut connections are not used in our proposed model, as explained in Section 6.

4 MULTIMODAL LOW-RANK BILINEAR ATTENTION NETWORKS

In this section, we apply low-rank bilinear pooling to propose an efficient attention mechanism for visual question-answering tasks, based on the interpretation of previous section. We assumed that inputs are a question embedding vector $\mathbf{q}$ and a set of visual feature vectors $\mathbf{F}$ over $S \times S$ lattice space.

4.1 LOW-RANK BILINEAR POOLING IN ATTENTION MECHANISM

Attention mechanism uses an attention probability distribution α over $S \times S$ lattice space. Here, using low-rank bilinear pooling, α is defined as

$$\alpha = \text{softmax}\Big(\mathbf{P}_\alpha^T\big(\sigma(\mathbf{U}_\mathbf{q}^T\mathbf{q} \cdot \mathbb{1}^T) \circ \sigma(\mathbf{V}_\mathbf{F}^T\mathbf{F}^T)\big)\Big) \tag{8}$$

where $\alpha \in \mathbb{R}^{G \times S^2}$, $\mathbf{P}_\alpha \in \mathbb{R}^{d \times G}$, σ is a hyperbolic tangent function, $\mathbf{U}_\mathbf{q} \in \mathbb{R}^{N \times d}$, $\mathbf{q} \in \mathbb{R}^N$, $\mathbb{1} \in \mathbb{R}^{S^2}$, $\mathbf{V}_\mathbf{F} \in \mathbb{R}^{M \times d}$, and $\mathbf{F} \in \mathbb{R}^{S^2 \times M}$. If $G > 1$, multiple glimpses are explicitly expressed as in Fukui et al. (2016), conceptually similar to Jaderberg et al. (2015). And, the softmax function applies to each row vector of α. The bias terms are omitted for simplicity.

4.2 MULTIMODAL LOW-RANK BILINEAR ATTENTION NETWORKS

Attended visual feature $\hat{\mathbf{v}}$ is a linear combination of $\mathbf{F}_i$ with coefficients $\alpha_{g,i}$. Each attention probability distribution α_g is for a glimpse g. For $G > 1$, $\hat{\mathbf{v}}$ is the concatenation of resulting vectors $\hat{\mathbf{v}}_g$ as

$$\hat{\mathbf{v}} = \Big\|_{g=1}^{G} \sum_{s=1}^{S^2} \alpha_{g,s}\mathbf{F}_s \tag{9}$$

where $\|$ denotes concatenation of vectors. The posterior probability distribution is an output of a softmax function, whose input is the result of another low-rank bilinear pooling of $\mathbf{q}$ and $\hat{\mathbf{v}}$ as

$$p(a|\mathbf{q}, \mathbf{F}; \Theta) = \text{softmax}\Big(\mathbf{P}_o^T\big(\sigma(\mathbf{W}_\mathbf{q}^T\mathbf{q}) \circ \sigma(\mathbf{V}_{\hat{\mathbf{v}}}^T\hat{\mathbf{v}})\big)\Big) \tag{10}$$

$$\hat{a} = \arg\max_{a \in \Omega} p(a|\mathbf{q}, \mathbf{F}; \Theta) \tag{11}$$

where $\hat{a}$ denotes a predicted answer, Ω is a set of candidate answers and Θ is an aggregation of entire model parameters.

5 EXPERIMENTS

In this section, we conduct six experiments to select the proposed model, Multimodal Low-rank Bilinear Attention Networks (MLB). Each experiment controls other factors except one factor to assess the effect on accuracies. Based on MRN (Kim et al., 2016b), we start our assessments with an initial option of $G = 1$ and shortcut connections of MRN, called as Multimodal Attention Residual Networks (MARN). Notice that we use one embeddings for each visual feature for better performance, based on our preliminary experiment (not shown). We attribute this choice to the attention mechanism for visual features, which provides more capacity to learn visual features. We use the same hyper-parameters of MRN (Kim et al., 2016b), without any explicit mention of this.

The VQA dataset (Antol et al., 2015) is used as a primary dataset, and, for data augmentation, question-answering annotations of Visual Genome (Krishna et al., 2016) are used. Validation is performed on the VQA *test-dev* split, and model comparison is based on the results of the VQA *test-standard* split. For the comprehensive reviews of VQA tasks, please refer to Wu et al. (2016a) and Kafle & Kanan (2016a). The details about preprocessing, question and vision embedding, and hyperparameters used in our experiments are described in Appendix A. The source code for the experiments is available in Github repository[1].

Number of Learning Blocks Kim et al. (2016b) argue that three-block layered MRN shows the best performance among one to four-block layered models, taking advantage of residual learning. However, we speculate that an introduction of attention mechanism makes deep networks hard to optimize. Therefore, we explore the number of learning blocks of MARN, which have an attention mechanism using low-rank bilinear pooling.

Number of Glimpses Fukui et al. (2016) show that the attention mechanism of two glimpses was an optimal choice. In a similar way, we assess one, two, and four-glimpse models.

[1] https://github.com/jnhwkim/MulLowBiVQA

Table 1: The accuracies of our experimental model, Multimodal Attention Residual Networks (MARN), with respect to the number of learning blocks (L#), the number of glimpse (G#), the position of activation functions (tanh), answer sampling, shortcut connections, and data augmentation using Visual Genome dataset, for VQA *test-dev* split and Open-Ended task. Note that our proposed model, Multimodal Low-rank Bilinear Attention Networks (MLB) have no shortcut connections, compared with MARN. **MODEL**: model name, **SIZE**: number of parameters, **ALL**: overall accuracy in percentage, **Y/N**: yes/no, **NUM**: numbers, and **ETC**: others. Since Fukui et al. (2016) only report the accuracy of the ensemble model on the *test-standard*, the *test-dev* results of their single models are included in the last sector. Some figures have different precisions which are rounded. $*$ indicates the selected model for each experiment.

MODEL	SIZE	ALL	Y/N	NUM	ETC
MRN-L3	65.0M	61.68	82.28	38.82	49.25
MARN-L3	65.5M	62.37	82.31	38.06	50.83
MARN-L2	56.3M	63.92	82.88	37.98	53.59
* MARN-L1	47.0M	63.79	82.73	37.92	53.46
MARN-L1-G1	47.0M	63.79	82.73	37.92	53.46
* MARN-L1-G2	57.7M	64.53	83.41	37.82	54.43
MARN-L1-G4	78.9M	64.61	83.72	37.86	54.33
No Tanh	57.7M	63.58	83.18	37.23	52.79
* Before-Product	57.7M	64.53	83.41	37.82	54.43
After-Product	57.7M	64.53	83.53	37.06	54.50
Mode Answer	57.7M	64.53	83.41	37.82	54.43
* Sampled Answer	57.7M	64.80	83.59	38.38	54.73
Shortcut	57.7M	64.80	83.59	38.38	54.73
* No Shortcut	51.9M	65.08	84.14	38.21	54.87
MLB	51.9M	65.08	84.14	38.21	54.87
MLB+VG	51.9M	65.84	83.87	37.87	56.76
MCB+Att (Fukui et al., 2016)	69.2M	64.2	82.2	37.7	54.8
MCB+Att+GloVe (Fukui et al., 2016)	70.5M	64.7	82.5	37.6	55.6
MCB+Att+Glove+VG (Fukui et al., 2016)	70.5M	65.4	82.3	37.2	57.4

Non-Linearity We assess three options applying non-linearity on low-rank bilinear pooling, vanilla, before Hadamard product as in Equation 5, and after Hadamard product as in Equation 6.

Answer Sampling VQA (Antol et al., 2015) dataset has ten answers from unique persons for each question, while Visual Genome (Krishna et al., 2016) dataset has a single answer for each question. Since difficult or ambiguous questions may have divided answers, the probabilistic sampling from the distribution of answers can be utilized to optimize for the multiple answers. An instance [2] can be found in Fukui et al. (2016). We simplify the procedure as follows:

$$p(a_1) = \begin{cases} |a_1|/\Sigma_i |a_i|, & \text{if } |a_1| \geq 3 \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

$$p(a_0) = 1 - p(a_1) \tag{13}$$

where $|a_i|$ denotes the number of unique answer a_i in a set of multiple answers, a_0 denotes a mode, which is the most frequent answer, and a_1 denotes the secondly most frequent answer. We define the divided answers as having at least three answers which are the secondly frequent one, for the evaluation metric of VQA (Antol et al., 2015),

$$\text{accuracy}(a_k) = \min\left(|a_k|/3, 1\right). \tag{14}$$

[2] https://github.com/akirafukui/vqa-mcb/blob/5fea8/train/multi_att_2_glove/vqa_data_provider_layer.py#L130

Table 2: The VQA *test-standard* results to compare with state-of-the-art. Notice that these results are trained by provided VQA train and validation splits, without any data augmentation.

	Open-Ended				MC
MODEL	**ALL**	**Y/N**	**NUM**	**ETC**	**ALL**
iBOWIMG (Zhou et al., 2015)	55.89	76.76	34.98	42.62	61.97
DPPnet (Noh et al., 2016)	57.36	80.28	36.92	42.24	62.69
Deeper LSTM+Normalized CNN (Antol et al., 2015)	58.16	80.56	36.53	43.73	63.09
SMem (Xu & Saenko, 2016)	58.24	80.80	37.53	43.48	-
Ask Your Neurons (Malinowski et al., 2016)	58.43	78.24	36.27	46.32	-
SAN (Yang et al., 2016)	58.85	79.11	36.41	46.42	-
D-NMN (Andreas et al., 2016)	59.44	80.98	37.48	45.81	-
ACK (Wu et al., 2016b)	59.44	81.07	37.12	45.83	-
FDA (Ilievski et al., 2016)	59.54	81.34	35.67	46.10	64.18
HYBRID (Kafle & Kanan, 2016b)	60.06	80.34	37.82	47.56	-
DMN+ (Xiong et al., 2016)	60.36	80.43	36.82	48.33	-
MRN (Kim et al., 2016b)	61.84	82.39	**38.23**	49.41	66.33
HieCoAtt (Lu et al., 2016)	62.06	79.95	38.22	51.95	66.07
RAU (Noh & Han, 2016)	63.2	81.7	38.2	52.8	67.3
MLB (ours)	**65.07**	**84.02**	37.90	**54.77**	**68.89**

The rate of the divided answers is approximately 16.40%, and only 0.23% of questions have more than two divided answers in VQA dataset. We assume that it eases the difficulty of convergence without severe degradation of performance.

Shortcut Connection The contribution of shortcut connections for residual learning is explored based on the observation of the competitive performance of single-block layered model. Since the usefulness of shortcut connections is linked to the network depth (He et al., 2016).

Data Augmentation The data augmentation with Visual Genome (Krishna et al., 2016) question answer annotations is explored. Visual Genome (Krishna et al., 2016) originally provides 1.7 Million visual question answer annotations. After aligning to VQA, the valid number of question-answering pairs for training is 837,298, which is for distinct 99,280 images.

6 RESULTS

The six experiments are conducted sequentially. Each experiment determines experimental variables one by one. Refer to Table 1, which has six sectors divided by mid-rules.

6.1 SIX EXPERIMENT RESULTS

Number of Learning Blocks Though, MRN (Kim et al., 2016b) has the three-block layered architecture, MARN shows the best performance with two-block layered models (63.92%). For the multiple glimpse models in the next experiment, we choose one-block layered model for its simplicity to extend, and competitive performance (63.79%).

Number of Glimpses Compared with the results of Fukui et al. (2016), four-glimpse MARN (64.61%) is better than other comparative models. However, for a parsimonious choice, two-glimpse MARN (64.53%) is chosen for later experiments. We speculate that multiple glimpses are one of key factors for the competitive performance of MCB (Fukui et al., 2016), based on a large margin in accuracy, compared with one-glimpse MARN (63.79%).

Non-Linearity The results confirm that activation functions are useful to improve performances. Surprisingly, there is no empirical difference between two options, before-Hadamard product and

after-Hadamard product. This result may build a bridge to relate with studies on multiplicative integration with recurrent neural networks (Wu et al., 2016c).

Answer Sampling Sampled answers (64.80%) result better performance than mode answers (64.53%). It confirms that the distribution of answers from annotators can be used to improve the performance. However, the number of multiple answers is usually limited due to the cost of data collection.

Shortcut Connection Though, MRN (Kim et al., 2016b) effectively uses shortcut connections to improve model performance, one-block layered MARN shows better performance without the shortcut connection. In other words, the residual learning is not used in our proposed model, MLB. It seems that there is a trade-off between introducing attention mechanism and residual learning. We leave a careful study on this trade-off for future work.

Data Augmentation Data augmentation using Visual Genome (Krishna et al., 2016) question answer annotations significantly improves the performance by 0.76% in accuracy for VQA *test-dev* split. Especially, the accuracy of *others* (ETC)-type answers is notably improved from the data augmentation.

6.2 Comparison with State-of-the-Art

The comparison with other single models on VQA *test-standard* is shown in Table 2. The overall accuracy of our model is approximately 1.9% above the next best model (Noh & Han, 2016) on the *Open-Ended* task of VQA. The major improvements are from *yes-or-no* (Y/N) and *others* (ETC)-type answers. In Table 3, we also report the accuracy of our ensemble model to compare with other ensemble models on VQA *test-standard*, which won 1st to 5th places in VQA Challenge 2016[3]. We beat the previous state-of-the-art with a margin of 0.42%.

Table 3: The VQA *test-standard* results for ensemble models to compare with state-of-the-art. For unpublished entries, their team names are used instead of their model names. Some of their figures are updated after the challenge.

	Open-Ended				MC
MODEL	**ALL**	**Y/N**	**NUM**	**ETC**	**ALL**
RAU (Noh & Han, 2016)	64.12	83.33	38.02	53.37	67.34
MRN (Kim et al., 2016b)	63.18	83.16	39.14	51.33	67.54
DLAIT (not published)	64.83	83.23	**40.80**	54.32	68.30
Naver Labs (not published)	64.79	83.31	38.70	54.79	69.26
MCB (Fukui et al., 2016)	66.47	83.24	39.47	**58.00**	70.10
MLB (ours)	**66.89**	**84.61**	39.07	57.79	**70.29**
Human (Antol et al., 2015)	83.30	95.77	83.39	72.67	91.54

7 Related Works

MRN (Kim et al., 2016b) proposes multimodal residual learning with Hadamard product of low-rank bilinear pooling. However, their utilization of low-rank bilinear pooling is limited to joint residual mapping function for multimodal residual learning. Higher-order Boltzmann Machines (Memisevic & Hinton, 2007; 2010) use Hadamard product to capture the interactions of input, output, and hidden representations for energy function. Wu et al. (2016c) propose the recurrent neural networks using Hadamard product to integrate multiplicative interactions among hidden representations in the model. For details of these related works, please refer to Appendix D.

[3]http://visualqa.org/challenge.html

Yet, compact bilinear pooling or multimodal compact bilinear pooling (Gao et al., 2016; Fukui et al., 2016) is worth to discuss and carefully compare with our method.

7.1 COMPACT BILINEAR POOLING

Compact bilinear pooling (Gao et al., 2016) approximates full bilinear pooling using a sampling-based computation, Tensor Sketch Projection (Charikar et al., 2002; Pham & Pagh, 2013):

$$\Psi(x \otimes y, h, s) = \Psi(x, h, s) * \Psi(y, h, s) \tag{15}$$

$$= \text{FFT}^{-1}(\text{FFT}(\Psi(x, h, s) \circ \text{FFT}(\Psi(y, h, s)) \tag{16}$$

where $\otimes$ denotes outer product, $*$ denotes convolution, $\Psi(v, h, s)_i := \sum_{j:h_j=i} s_j \cdot v_j$, FFT denotes Fast Fourier Transform, d denotes an output dimension, $x, y, h, s \in \mathbb{R}^n$, x and y are inputs, and h and s are random variables. h_i is sampled from $\{1, ..., d\}$, and s_i is sampled from $\{-1, 1\}$, then, both random variables are fixed for further usage. Even if the dimensions of x and y are different from each other, it can be used for multimodal learning (Fukui et al., 2016).

Similarly to Equation 1, compact bilinear pooling can be described as follows:

$$f_i = \mathbf{x}^T \mathcal{W}_i \mathbf{y} \tag{17}$$

where $\mathcal{W}_{ijk} = s_{ijk} w_{ijk}$ if s_{ijk} is sampled from $\{-1, 1\}$, w_{ijk} is sampled from $\{\mathbf{P}_{i1}, \mathbf{P}_{i2}, \ldots, \mathbf{P}_{id}\}$, and the compact bilinear pooling is followed by a fully connected layer $\mathbf{P} \in \mathbb{R}^{|\Omega| \times d}$. Then, this method can be formulated as a hashing trick (Weinberger et al., 2009; Chen et al., 2015) to share randomly chosen bilinear weights using d parameters for a output value, in a way that a single parameter is shared by NM/d bilinear terms in expectation, with the variance of $NM(d-1)/d^2$ (See Appendix B).

In comparison with our method, their method approximates a three-dimensional weight tensor in bilinear pooling with a two-dimensional matrix $\mathbf{P}$, which is larger than the concatenation of three two-dimensional matrices for low-rank bilinear pooling. The ratio of the number of parameters for a single output to the total number of parameters for $|\Omega|$ outputs is $d/d|\Omega| = 1/|\Omega|$ (Fukui et al., 2016), vs. $d(N + M + 1)/d(N + M + |\Omega|) = (N + M + 1)/(N + M + |\Omega|) \approx 2/3$ (ours), since our method uses a three-way factorization. Hence, more parameters are allocated to each bilinear approximation than compact bilinear pooling does, effectively managing overall parameters guided by back-propagation algorithm.

MCB (Fukui et al., 2016), which uses compact bilinear pooling for multimodal tasks, needs to set the dimension of output d to 16K, to reduce the bias induced by the fixed random variables h and s. As a result, the majority of model parameters (16K $\times$ 3K = 48M) are concentrated on the last fully connected layer, which makes a *fan-out* structure. So, the total number of parameters of MCB is highly sensitive to the number of classes, which is approximately 69.2M for *MCB+att*, and 70.5M for *MCB+att+GloVe*. Yet, the total number of parameters of our proposed model (MLB) is 51.9M, which is more robust to the number of classes having d = 1.2K, which has a similar role in model architecture.

8 CONCLUSIONS

We suggest a low-rank bilinear pooling method to replace compact bilinear pooling, which has a *fan-out* structure, and needs complex computations. Low-rank bilinear pooling has a flexible structure using linear mapping and Hadamard product, and a better parsimonious property, compared with compact bilinear pooling. We achieve new state-of-the-art results on the VQA dataset using a similar architecture of Fukui et al. (2016), replacing compact bilinear pooling with low-rank bilinear pooling. We believe our method could be applicable to other bilinear learning tasks.

ACKNOWLEDGMENTS

The authors would like to thank Patrick Emaase for helpful comments and editing. Also, we are thankful to anonymous reviewers who provided comments to improve this paper. This work was supported by NAVER LABS Corp. & NAVER Corp. and partly by the Korea government (IITP-R0126-16-1072-SW.StarLab, KEIT-10044009-HRI.MESSI, KEIT-10060086-RISF, ADD-UD130070ID-BMRR). The part of computing resources used in this study was generously shared by Standigm Inc.

REFERENCES

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to Compose Neural Networks for Question Answering. *arXiv preprint arXiv:1601.01705*, 2016.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. *IEEE International Conference on Computer Vision*, 2015.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pp. 693–703. Springer, 2002.

Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing Neural Networks with the Hashing Trick. In *32nd International Conference on Machine Learning*, pp. 2285–2294, 2015.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding. *arXiv preprint arXiv:1606.01847*, 2016.

Yarin Gal. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. *arXiv preprint arXiv:1512.05287*, 2015.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact Bilinear Pooling. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

Ilija Ilievski, Shuicheng Yan, and Jiashi Feng. A Focused Dynamic Attention Model for Visual Question Answering. *arXiv preprint arXiv:1604.01485*, 2016.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems 28*, pp. 2008–2016, 2015.

Kushal Kafle and Christopher Kanan. Visual Question Answering: Datasets, Algorithms, and Future Challenges. *arXiv preprint arXiv:1610.01465*, 2016a.

Kushal Kafle and Christopher Kanan. Answer-Type Prediction for Visual Question Answering. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4976–4984, 2016b.

Jin-Hwa Kim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. TrimZero: A Torch Recurrent Module for Efficient Natural Language Processing. In *KIIS Spring Conference*, volume 26, pp. 165–166, 2016a.

Jin-Hwa Kim, Sang-Woo Lee, Dong-Hyun Kwak, Min-Oh Heo, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Multimodal Residual Learning for Visual QA. *arXiv preprint arXiv:1606.01455*, 2016b.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems 28*, pp. 3294–3302, 2015.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprint arXiv:1602.07332*, 2016.

Nicholas Léonard, Sagar Waghmare, Yang Wang, and Jin-Hwa Kim. rnn : Recurrent Library for Torch. *arXiv preprint arXiv:1511.07889*, 2015.

Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear CNN Models for Fine-grained Visual Recognition. In *IEEE International Conference on Computer Vision*, pp. 1449–1457, 2015.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical Question-Image Co-Attention for Visual Question Answering. *arXiv preprint arXiv:1606.00061*, 2016.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask Your Neurons: A Deep Learning Approach to Visual Question Answering. *arXiv preprint arXiv:1605.02697*, 2016.

Roland Memisevic and Geoffrey E Hinton. Unsupervised learning of image transformations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

Roland Memisevic and Geoffrey E Hinton. Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural computation*, 22(6):1473–1492, 2010.

Hyeonwoo Noh and Bohyung Han. Training Recurrent Answering Units with Joint Loss Minimization for VQA. *arXiv preprint arXiv:1606.03647*, 2016.

Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. Image Question Answering using Convolutional Neural Network with Dynamic Parameter Prediction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 239–247. ACM, 2013.

Hamed Pirsiavash, Deva Ramanan, and Charless C. Fowlkes. Bilinear classifiers for visual recognition. In *Advances in Neural Information Processing Systems 22*, pp. 1482–1490, 2009.

Joshua B Tenenbaum and William T Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *26th International Conference on Machine Learning*, pp. 1113–1120, 2009.

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. Visual Question Answering: A Survey of Methods and Datasets. *arXiv preprint arXiv:1607.05910*, 2016a.

Qi Wu, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. Ask Me Anything: Free-form Visual Question Answering Based on Knowledge from External Sources. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016b.

Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan Salakhutdinov. On Multiplicative Integration with Recurrent Neural Networks. *arXiv preprint arXiv:1606.06630*, 2016c.

Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic Memory Networks for Visual and Textual Question Answering. In *33rd International Conference on Machine Learning*, 2016.

Huijuan Xu and Kate Saenko. Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering. In *European Conference on Computer Vision*, 2016.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked Attention Networks for Image Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Simple Baseline for Visual Question Answering. *arXiv preprint arXiv:1512.02167*, 2015.

Appendix

A Experiment Details

A.1 Preprocessing

We follow the preprocessing procedure of Kim et al. (2016b). Here, we remark some details of it, and changes.

A.1.1 Question Embedding

The 90.45% of questions for the 2K-most frequent answers are used. The vocabulary size of questions is 15,031. GRU (Cho et al., 2014) is used for question embedding. Based on earlier studies (Noh et al., 2016; Kim et al., 2016b), a word embedding matrix and a GRU are initialized with Skip-thought Vector pre-trained model (Kiros et al., 2015). As a result, question vectors have 2,400 dimensions.

For efficient computation of variable-length questions, Kim et al. (2016a) is used for the GRU. Moreover, for regularization, Bayesian Dropout (Gal, 2015) which is implemented in Léonard et al. (2015) is applied while training.

A.2 Vision Embedding

ResNet-152 networks (He et al., 2016) are used for feature extraction. The dimensionality of an input image is $3 \times 448 \times 448$. The outputs of the last convolution layer is used, which have $2,048 \times 14 \times 14$ dimensions.

A.3 Hyperparameters

The hyperparameters used in MLB of Table 2 are described in Table 4. The batch size is 100, and the number of iterations is fixed to 250K. For data augmented models, a simplified early stopping is used, starting from 250K to 350K-iteration for every 25K iterations (250K, 275K, 300K, 325K, and 350K; at most five points) to avoid exhaustive submissions to VQA *test-dev* evaluation server. RMSProp (Tieleman & Hinton, 2012) is used for optimization.

Though, the size of joint embedding size d is borrowed from Kim et al. (2016b), a grid search on d confirms this choice in our model as shown in Table 5.

Table 4: Hyperparameters used in MLB (single model in Table 2).

SYMBOL	VALUE	DESCRIPTION
S	14	attention lattice size
N	2,400	question embedding size
M	2,048	channel size of extracted visual features
d	1,200	joint embedding size
G	2	number of glimpses
$\|\Omega\|$	2,000	number of candidate answers
η	3e-4	learning rate
λ	0.99997592083	learning rate decay factor at every iteration
p	0.5	dropout rate
θ	± 10	gradient clipping threshold

A.4 Model Schema

Figure 1 shows a schematic diagram of MLB, where $\circ$ denotes Hadamard product, and Σ denotes a linear combination of visual feature vectors using coefficients, which is the output of softmax function. If $G > 1$, the softmax function is applied to each row vectors of an output matrix (Equation 8), and we concatenate the resulting vectors of the G linear combinations (Equation 9).

A.5 Ensemble of Seven Models

The *test-dev* results for individual models consisting of our ensemble model is presented in Table 6.

Table 5: The effect of joint embedding size d.

		Open-Ended			
d	**SIZE**	**ALL**	**Y/N**	**NUM**	**ETC**
800	45.0M	64.89	84.08	38.15	54.55
1000	48.4M	65.06	**84.18**	38.01	54.85
1200	51.9M	**65.08**	84.14	**38.21**	**54.87**
1400	55.4M	64.94	84.13	38.00	54.64
1600	58.8M	65.02	84.15	37.79	54.85

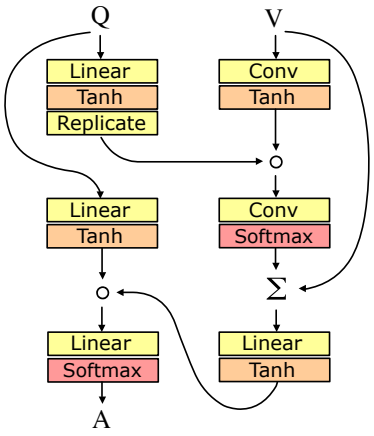

Figure 1: A schematic diagram of MLB. *Replicate* module copies an question embedding vector to match with S^2 visual feature vectors. *Conv* modules indicate 1×1 convolution to transform a given channel space, which is computationally equivalent to linear projection for channels.

Table 6: The individual models used in our ensemble model in Table 3.

		Open-Ended			
MODEL	**GLIMPSE**	**ALL**	**Y/N**	**NUM**	**ETC**
MLB	2	64.89	84.13	37.85	54.57
MLB	2	65.08	**84.14**	38.21	54.87
MLB	4	65.01	84.09	37.66	54.88
MLB-VG	2	65.76	83.64	37.57	56.86
MLB-VG	2	65.84	83.87	37.87	56.76
MLB-VG	3	66.05	83.88	38.13	57.13
MLB-VG	4	**66.09**	83.59	**38.32**	**57.42**
Ensemble	-	66.77	84.54	39.21	57.81

B UNDERSTANDING OF MULTIMODAL COMPACT BILINEAR POOLING

In this section, the algorithm of multimodal compact bilinear pooling (MCB) (Gao et al., 2016; Fukui et al., 2016) is described as a kind of hashing tick (Chen et al., 2015).

$\mathbf{x} \in \mathbb{R}^{n_x}$ and $\mathbf{y} \in \mathbb{R}^{n_y}$ are the given inputs, $\Phi(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^d$ is the output. Random variables $\mathbf{h}_x \in \mathbb{N}^{n_x}$ and $\mathbf{h}_y \in \mathbb{N}^{n_y}$ are uniformly sampled from $\{1, \dots, d\}$, and $\mathbf{s}_x \in \mathbb{Z}^{n_x}$ and $\mathbf{s}_y \in \mathbb{Z}^{n_y}$ are uniformly sampled from $\{-1, 1\}$. Then, Count Sketch projection function Ψ (Charikar et al., 2002) projects $\mathbf{x}$ and $\mathbf{y}$ to intermediate representations $\Psi(\mathbf{x}, \mathbf{h}_x, \mathbf{s}_x) \in \mathbb{R}^d$ and $\Psi(\mathbf{y}, \mathbf{h}_y, \mathbf{s}_y) \in \mathbb{R}^d$, which is defined as:

$$\Psi(\mathbf{v}, \mathbf{h}, \mathbf{s})_i := \sum_{j:h_j=i} s_j \cdot v_j \tag{18}$$

Notice that both $\mathbf{h}$ and $\mathbf{s}$ remain as constants after initialization (Fukui et al., 2016).

The probability of $h_{xj} = i$ and $h_{yj} = i$ for the given j is $1/d^2$. Hence, the expected number of bilinear terms in $\Psi(\mathbf{x}, \mathbf{h}_x, \mathbf{s}_x)_i \Psi(\mathbf{y}, \mathbf{h}_y, \mathbf{s}_y)_i$ is $(n_x n_y)/d^2$. Since, the output $\Phi(\mathbf{x}, \mathbf{y})$ is a result of circular convolution of $\Psi(\mathbf{x}, \mathbf{h}_x, \mathbf{s}_x)$ and $\Psi(\mathbf{y}, \mathbf{h}_y, \mathbf{s}_y)$, the expected number of bilinear terms in $\Phi(\mathbf{x}, \mathbf{y})_i$ is $(n_x n_y)/d$. Likewise, the probability of that a bilinear term is allocated in $\Phi(\mathbf{x}, \mathbf{y})_i$ is $1/d$. The probability distribution of the number of bilinear terms in $\Phi(\mathbf{x}, \mathbf{y})_i$ follows a multinomial distribution, whose mean is $(n_x n_y)/d$ and variance is $(n_x n_y)(d-1)/d^2$.

Linear projection after the multimodal compact bilinear pooling provides weights on the bilinear terms, in a way that a shared weight is assigned to $\Phi(\mathbf{x}, \mathbf{y})_i$, which has $(n_x n_y)/d$ bilinear terms in expectation, though each bilinear term can have a different sign induced by both $\mathbf{s}_x$ and $\mathbf{s}_y$.

HashedNets (Chen et al., 2015) propose a method to compress neural networks using a low-cost hashing function (Weinberger et al., 2009), which is the same function of $\Psi(\mathbf{v}, \mathbf{h}, \mathbf{s})$. They randomly group a portion of connections in neural networks to share a single weight. We speculate that multimodal compact bilinear pooling uses the hashing tick to reduce the number of full bilinear weights with the rate of $d/(n_x n_y)$. However, this approximation is limited to two-way interaction, compared with three-way factorization in our method.

C REPLACEMENT OF LOW-RANK BILINEAR POOLING

For the explicit comparison with compact bilinear pooling, we explicitly substitute compact bilinear pooling for low-rank bilinear pooling to control everything else, which means that the rest of the model architecture is exactly the same.

According to Fukui et al. (2016), we use MCB followed by Signed Square Root, L2-Normalization, Dropout (p=0.1), and linear projection from 16,000-dimension to the target dimension. Also, Dropout (p=0.3) for a question embedding vector. Note that an overall architecture for multimodal learning of both is the same. Experimental details are referenced from the implementation [4] of Fukui et al. (2016).

For test-dev split, our version of MCB gets 61.48% for overall accuracy (yes/no: 82.48%, number: 37.06%, and other: 49.07%) vs. 65.08% (ours, MLB in Table 1). Additionally, if the nonlinearity in getting attention distributions is increased as the original MCB does using ReLU, we get 62.11% for overall accuracy (yes/no: 82.55%, number: 37.18%, and other: 50.30%), which is still the below of our performance [5].

We do not see it as a decisive evidence of the better performance of MLB, but as a reference (the comparison of test-dev results may be also unfair.), since an optimal architecture and hyperparameters may be required for each method.

[4] https://github.com/akirafukui/vqa-mcb

[5] Our version of MCB definition can be found in https://github.com/jnhwkim/MulLowBiVQA/blob/master/netdef/MCB.lua

D RELATED WORKS

D.1 MULTIMODAL RESIDUAL NETWORKS

MRN (Kim et al., 2016b) is an implicit attentional model using multimodal residual learning with Hadamard product which does not have any explicit attention mechanism.

$$\mathcal{F}^{(k)}(\mathbf{q}, \mathbf{v}) = \sigma(\mathbf{W}_{\mathbf{q}}^{(k)}\mathbf{q}) \circ \sigma(\mathbf{W}_2^{(k)}\sigma(\mathbf{W}_1^{(k)}\mathbf{v})) \tag{19}$$

$$H_L(\mathbf{q}, \mathbf{v}) = \mathbf{W}_{\mathbf{q}'}\mathbf{q} + \sum_{l=1}^{L} \mathbf{W}_{\mathcal{F}^{(l)}}\mathcal{F}^{(l)}(H_{l-1}, \mathbf{v}) \tag{20}$$

where $\mathbf{W}_*$ are parameter matrices, L is the number of learning blocks, $H_0 = \mathbf{q}$, $\mathbf{W}_{\mathbf{q}'} = \Pi_{l=1}^{L}\mathbf{W}_{\mathbf{q}'}^{(l)}$, and $\mathbf{W}_{\mathcal{F}^{(l)}} = \Pi_{m=l+1}^{L}\mathbf{W}_{\mathbf{q}'}^{(m)}$. Notice that these equations can be generalized by Equation 7.

However, an explicit attention mechanism allows the use of lower-level visual features than fully-connected layers, and, more importantly, spatially selective learning. Recent state-of-the-art methods use a variant of an explicit attention mechanism in their models (Lu et al., 2016; Noh & Han, 2016; Fukui et al., 2016). Note that shortcut connections of MRN are not used in the proposed Multimodal Low-rank Bilinear (MLB) model. Since, it does not have any performance gain due to not stacking multiple layers in MLB. We leave the study of residual learning for MLB for future work, which may leverage the excellency of bilinear models as suggested in Wu et al. (2016a).

D.2 HIGHER-ORDER BOLTZMANN MACHINES

A similar model can be found in a study of Higher-Order Boltzmann Machines (Memisevic & Hinton, 2007; 2010). They suggest a factoring method for the three-way energy function to capture correlations among input, output, and hidden representations.

$$\begin{aligned} -E(\mathbf{y}, \mathbf{h}; \mathbf{x}) &= \sum_f \big(\sum_i x_i w_{if}^x\big)\big(\sum_j y_j w_{jf}^y\big)\big(\sum_k h_k w_{kf}^h\big) + \sum_k w_k^h h_k + \sum_j w_j^y y_j \\ &= \big(\mathbf{x}^T\mathbf{W}^x \circ \mathbf{y}^T\mathbf{W}^y \circ \mathbf{h}^T\mathbf{W}^h\big)\mathbb{1} + \mathbf{h}^T\mathbf{w}^h + \mathbf{y}^T\mathbf{w}^y \end{aligned} \tag{21}$$

Setting aside of bias terms, the $I \times J \times K$ parameter tensor of unfactored Higher-Order Boltzmann Machines is replaced with three matrices, $\mathbf{W}^x \in \mathbb{R}^{I \times F}$, $\mathbf{W}^y \in \mathbb{R}^{J \times F}$, and $\mathbf{W}^h \in \mathbb{R}^{K \times F}$.

D.3 MULTIPLICATIVE INTEGRATION WITH RECURRENT NEURAL NETWORKS

Most of recurrent neural networks, including vanilla RNNs, Long Short Term Memory networks (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Units (Cho et al., 2014), share a common expression as follows:

$$\phi(\mathbf{W}\mathbf{x} + \mathbf{U}\mathbf{h} + \mathbf{b}) \tag{22}$$

where ϕ is a non-linear function, $\mathbf{W} \in \mathbb{R}^{d \times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{U} \in \mathbb{R}^{d \times m}$, $\mathbf{h} \in \mathbb{R}^m$, and $\mathbf{b} \in \mathbb{R}^d$ is a bias vector. Note that, usually, $\mathbf{x}$ is an input state vector and $\mathbf{h}$ is an hidden state vector in recurrent neural networks.

Wu et al. (2016c) propose a new design to replace the additive expression with a multiplicative expression using Hadamard product as

$$\phi(\mathbf{W}\mathbf{x} \circ \mathbf{U}\mathbf{h} + \mathbf{b}). \tag{23}$$

Moreover, a general formulation of this multiplicative integration can be described as

$$\phi(\boldsymbol{\alpha} \circ \mathbf{W}\mathbf{x} \circ \mathbf{U}\mathbf{h} + \mathbf{W}\mathbf{x} \circ \boldsymbol{\beta}_1 + \mathbf{U}\mathbf{h} \circ \boldsymbol{\beta}_2 + \mathbf{b}) \tag{24}$$

which is reminiscent of *full model* in Section 3.1.