MUTAN: Multimodal Tucker Fusion for Visual Question Answering

Hedi Ben-Younes, Remi Cadene, Matthieu Cord, Nicolas Thome

▶ **To cite this version:**

Hedi Ben-Younes, Remi Cadene, Matthieu Cord, Nicolas Thome. MUTAN: Multimodal Tucker Fusion for Visual Question Answering. 2017 IEEE International Conference on Computer Vision (ICCV), Oct 2017, Venice, Italy. pp.2631-2639, 10.1109/ICCV.2017.285 . hal-02073637

HAL Id: hal-02073637

https://hal.sorbonne-universite.fr/hal-02073637

Submitted on 26 Mar 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

arXiv:1705.06676v1 [cs.CV] 18 May 2017

MUTAN: Multimodal Tucker Fusion for Visual Question Answering

Hedi Ben-younes [1,2] * Rémi Cadene [1]* Matthieu Cord [1] Nicolas Thome [3]
[1] Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu, 75005 Paris
[2] Heuritech, 248 rue du Faubourg Saint-Antoine, 75012 Paris
[3] Conservatoire National des Arts et Métiers
hedi.ben-younes@lip6.fr, remi.cadene@lip6.fr, matthieu.cord@lip6.fr, nicolas.thome@cnam.fr

Abstract

Bilinear models provide an appealing framework for mixing and merging information in Visual Question Answering (VQA) tasks. They help to learn high level associations between question meaning and visual concepts in the image, but they suffer from huge dimensionality issues.

We introduce MUTAN, a multimodal tensor-based Tucker decomposition to efficiently parametrize bilinear interactions between visual and textual representations. Additionally to the Tucker framework, we design a low-rank matrix-based decomposition to explicitly constrain the interaction rank. With MUTAN, we control the complexity of the merging scheme while keeping nice interpretable fusion relations. We show how our MUTAN model generalizes some of the latest VQA architectures, providing state-of-the-art results.

1. Introduction

Multimodal representation learning for text and image has been extensively studied in recent years. Currently, the most popular task is certainly Visual Question Answering (VQA) [19, 2]. VQA is a complex multimodal task which aims at answering a question about an image. A specific benchmark has been first proposed [19], and large scale datasets have been recently collected [21, 2, 31], enabeling the development of more powerful models.

To solve this problem, precise image and text models are required and, most importantly, high level interactions between these two modalities have to be carefully encoded into the model in order to provide the correct answer. This projection from the unimodal spaces to a multimodal one is supposed to extract and model the relevant correlations between the two spaces. Besides, the model must have the ability to understand the full scene, focus its attention on the relevant visual regions and discard the useless information regarding the question.

*Equal contribution

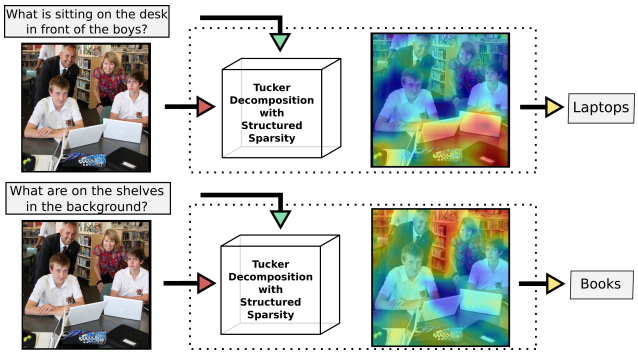


Figure 1: The proposed MUTAN model uses a Tucker decomposition of the image/question correlation tensor, which enables modeling rich and accurate multi-modal interactions. For the same input image, we show the result of the MUTAN fusion process when integrated into an attention mechanism [28]: we can see that the regions with larger attention scores (in red) indicate a very fine understanding of the image and question contents, enabling MUTAN to properly answer the question (see detailed maps in experiments section).

Bilinear models are powerful approaches for the fusion problem in VQA because they encode full second-order interactions. They currently hold state-of the-art performances [5, 8]. The main issue with these bilinear models is related to the number of parameters, which quickly becomes intractable with respect to the input and output dimensions. Therefore, current bilinear approaches must be simplified or approximated by reducing the model complexity: in [5], the authors sacrifice trainability by using a handcrafted multi-modal projection, while a global tensor rank constraint is applied in [8], reducing correlations to a simple element-wise product.

In this work, we introduce a new architecture called MUTAN (Figure 2), which focuses on modeling fine and rich interactions between image and textual modalities. Our approach is based on a Tucker decomposition [24] of the cor-

relation tensor, which is able to represent full bilinear interactions, while maintaining the size of the model tractable. The resulting scheme allows us to explicitly control the model complexity, and to choose an accurate and interpretable repartition of the learnable parameters.

In the next section, we provide more details on related VQA works and highlight our contributions. The MUTAN fusion model, based on a Tucker decomposition, is presented in section 3, and successful experiments are reported in section 4.

2. Related work

The main task in multimodal visual and textual analysis aims at learning an alignment between feature spaces [29, 23, 18]. Thus, the recent task of image captioning aims at generating linguistic descriptions of images [25, 10, 28]. Instead of explicitly learning an alignment between two spaces, the goal of VQA [2, 19] is to merge both modalities in order to decide which answer is correct. This problem requires modeling very precise correlations between the image and the question representations.

Attention. Attention mechanisms [28] have been a real breakthrough in multimodal systems, and are fundamental for VQA models to obtain the best possible results. [30] propose to stack multiple question-guided attention mechanisms, each one looking at different regions of the image.

[22] and [14] extract bounding boxes in the image and score each one of them according to the textual features. In [17], word features are aggregated with an attention mechanism guided by the image regions and, equivalently, the region visual features are aggregated into one global image embedding. This co-attentional framework uses concatenations and sum pooling to merge all the components. On the contrary, [5] and [8] developed their own fusion methods that they use for global and attention-based strategies.

In this paper, we use the attentional modeling, proposed in [5], as a tool that we integrate in our new fusion strategy for both the global fusion and the attentional modeling.

Fusion strategies. Early works have modeled interactions between multiple modalities with first order interactions. The IMG+BOW model in [21] is the first to use a concatenation to merge a global image representation with a question embedding, obtained by summing all the learnt word embeddings from the question. In [22], (image, question, answer) triplets are scored in an attentional framework. Each local feature is given a score corresponding to its similarity with textual features. These scores are used to weight region multimodal embeddings, obtained from a concatenation between the region's visual features and the textual embeddings. The hierarchical co-attention network [17], after extracting multiple textual and visual features, merges them with concatenations and sums.

Second order models are a more powerful way to model interactions between two embedding spaces. Bilinear interactions have shown great success in deep learning for fine-grained classification [16], and Multimodal language modeling [10]. In VQA, a simple element-wise product between the two vectors is performed in [2]. [7] also uses an element-wise product in a more complex iterative global merging scheme. In [14], they use the element-wise product aggregation in an attentional framework. To go deeper in bilinear interactions, Multimodal Compact Bilinear pooling (MCB) [5] uses an outer product $\mathbf{q}\otimes\mathbf{v}$ between visual $\mathbf{v}$ and textual $\mathbf{q}$ embeddings. The count-sketch projection [3] Ψ is used to project $\mathbf{q}\otimes\mathbf{v}$ on a lower dimensional space. Interestingly, nice count-sketch properties are capitalized to compute the projection without having to explicitly compute the outer product. However, interaction parameters in MCB are fixed by the count-sketch projection (randomly chosen in $\{0; -1; 1\}$), limiting its expressive power for modeling complex interactions between image and questions. In contrast, our approach is able to model rich second order interaction with learned parameters.

In the recent Multimodal Low-rank Bilinear (MLB) pooling work [8], full bilinear interactions between image and question spaces are parametrized by a tensor. Again, to limit the number of free parameters, this tensor is constrained to be of low rank r. The MLB strategy reaches state-of-the-art performances on the well-known VQA database [2]. Despite these impressive results, the low rank tensor structure is equivalent to a projection of both visual and question representations into a common r-dimensional space, and to compute simple element-wise product interactions in this space. MLB is thus essentially designed to learn a powerful mono-modal embedding for text and image modalities, but relies on a simple fusion scheme in this space.

In this work, we introduce MUTAN, a multimodal fusion scheme based on bilinear interactions between modalities. To control the number of model parameters, MUTAN reduces the size of the mono-modal embeddings, while modeling their interaction as accurately as possible with a full bilinear fusion scheme. Our submission therefore encompasses the following contributions:

– New fusion scheme for VQA relying on a Tucker tensor-based decomposition, consisting in a factorization into three matrices and a core tensor. We show that the MUTAN fusion scheme generalizes the latest bilinear models, *i.e.* MCB [5] and MLB [8], while having more expressive power.

– Additional structured sparsity constraint the core tensor to further control the number of model parameters. This acts as a regularizer during training and prevents overfitting, giving us more flexibility to adjust the input/output projections.

– State-of-the-art results on the most widely used dataset

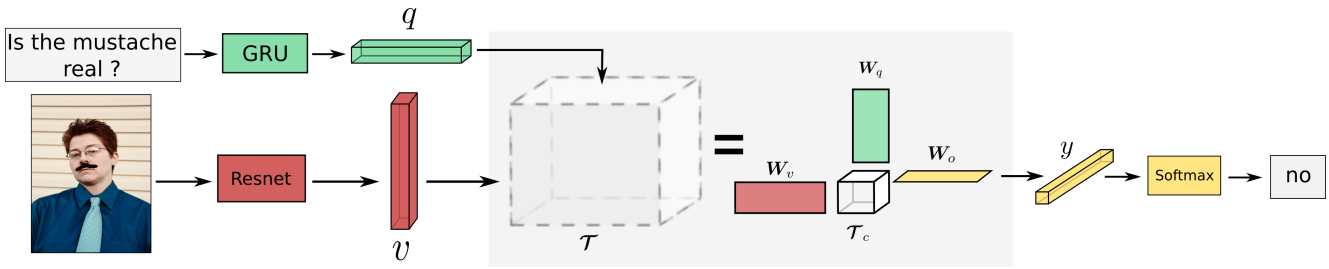


Figure 2: MUTAN fusion scheme for global Visual QA. The prediction is modeled as a bilinear interaction between visual and linguistic features, parametrized by the tensor $\mathcal{T}$. In MUTAN, we factorise the tensor $\mathcal{T}$ using a Tucker decomposition, resulting in an architecture with three intra-modal matrices $\boldsymbol{W}_q$, $\boldsymbol{W}_v$ and $\boldsymbol{W}_o$, and a smaller tensor $\mathcal{T}_c$. The complexity of $\mathcal{T}_c$ is controlled *via* a structured sparsity constraint on the slice matrices of the tensor.

for Visual QA [2]. We also show that MUTAN outperforms MCB [5] and MLB [8] in the same setting, and that performances can be further improved when combined with MLB, validating the complementarity potential between the two approaches.

3. MUTAN Model

Our method deals with the problem of Visual Question Answering (VQA). In VQA, one is given a question $q \in \mathcal{Q}$ about an image $v \in \mathcal{I}$, and the goal is to provide a meaningful answer. During training, we aim at learning a model such that the predicted answer $\hat{\mathbf{a}}$ matches the correct one $\mathbf{a}^\star$. More formally, denoting as Θ the whole set of parameters of the model, the predicted output $\hat{\mathbf{a}}$ can be written as:

$$\hat{a} = \arg\max_{a \in \mathcal{A}} p_\Theta\left(a|v,q\right) \tag{1}$$

The general architecture of the proposed approach is shown in Figure 2. As commonly done in VQA, images v and questions q are firstly embedded into vectors and the output is represented as a classification vector $\mathbf{y}$. In this work, we use a fully convolutional neural network [6] (ResNet-152) to describe the image content, and a GRU recurrent network [11, 4] for the question, yielding representations $\mathbf{v} \in \mathbb{R}^{d_v}$ for the image and $\mathbf{q} \in \mathbb{R}^{d_q}$ for the question. Vision and language representations $\mathbf{v}$ and $\mathbf{q}$ are then fused using the operator $\mathcal{T}$ (explained below) to produce a vector $\mathbf{y}$, providing (through a softmax function) the final answer in Eq. (1). This global merging scheme is also embedded into a visual attention-based mechanism [8] to provide our final MUTAN architecture.

Fusion and Bilinear models The issue of merging visual and linguistic information is crucial in VQA. Complex and high-level interactions between textual meaning in the question and visual concepts in the image have to be extracted to provide a meaningful answer.

Bilinear models [5, 8] are recent powerful solutions to the fusion problem, since they encode fully-parametrized bilinear interactions between the vectors $\mathbf{q}$ and $\mathbf{v}$:

$$\mathbf{y} = \left(\mathcal{T} \times_1 \mathbf{q}\right) \times_2 \mathbf{v} \tag{2}$$

with the full tensor $\mathcal{T} \in \mathbb{R}^{d_q \times d_v \times |\mathcal{A}|}$, and the operator $\times_i$ designing the *i-mode* product between a tensor and a matrix (here a vector).

Despite their appealing modeling power, fully-parametrized bilinear interactions quickly become intractable in VQA, because the size of the full tensor is prohibitive using common dimensions for textual, visual and output spaces. For example, with $d_v \approx d_q \approx 2048$ and $|\mathcal{A}| \approx 2000$, the number of free parameters in the tensor $\mathcal{T}$ is $\sim 10^{10}$. Such a huge number of free parameters is a problem both for learning and for GPU memory consumption[1].

In MUTAN, we factorize the full tensor $\mathcal{T}$ using a Tucker decomposition. We also propose to complete our decomposition by structuring the second tensor $\mathcal{T}_c$ (see gray box in Fig. 2) in order to keep flexibility over the input/output dimensions while keeping the number of parameters tractable.

3.1. Tucker decomposition

The Tucker decomposition [24] of a 3-way tensor $\mathcal{T} \in \mathbb{R}^{d_q \times d_v \times |\mathcal{A}|}$ expresses $\mathcal{T}$ as a tensor product between *factor matrices* $\boldsymbol{W}_q, \boldsymbol{W}_v$ and $\boldsymbol{W}_o$, and a *core tensor* $\mathcal{T}_c$ in such a way that:

$$\mathcal{T} = \left(\left(\mathcal{T}_c \times_1 \boldsymbol{W}_q\right) \times_2 \boldsymbol{W}_v\right) \times_3 \boldsymbol{W}_o \tag{3}$$

with $\boldsymbol{W}_q \in \mathbb{R}^{d_q \times t_q}$, $\boldsymbol{W}_v \in \mathbb{R}^{d_v \times t_v}$ and $\boldsymbol{W}_o \in \mathbb{R}^{|\mathcal{A}| \times t_o}$, and $\mathcal{T}_c \in \mathbb{R}^{t_q \times t_v \times t_o}$. Interestingly, Eq. (3) states that the

[1] A tensor with 8 billion float32 scalars approximately needs 32Go to be stored, while top-grade GPUs hold about 24Go each.

weights in $\mathcal{T}$ are functions of a restricted number of parameters $\forall i \in [1, d_q], j \in [1, d_v], k \in [1, d_o]$:

$$\mathcal{T}[i,j,k] = \sum_{l \in [1,t_q], m \in [1,t_v], n \in [1,t_o]} \mathcal{T}_c[l,m,n] \boldsymbol{W}_q[i,l] \boldsymbol{W}_v[j,m] \boldsymbol{W}_o[k,n]$$

$\mathcal{T}$ is usually summarized as $\mathcal{T} = [\![\mathcal{T}_c; \boldsymbol{W}_q, \boldsymbol{W}_v, \boldsymbol{W}_o]\!]$. A comprehensive discussion on Tucker decomposition and tensor analysis may be found in [12].

3.2. Multimodal Tucker Fusion

As we parametrize the weights of the tensor $\mathcal{T}$ with its Tucker decomposition of the Eq. (3), we can rewrite Eq. (2) as follows:

$$y = \left(\left(\mathcal{T}_c \times_1 \left(\mathbf{q}^\top \boldsymbol{W}_q \right) \right) \times_2 \left(\mathbf{v}^\top \boldsymbol{W}_v \right) \right) \times_3 \boldsymbol{W}_o \quad (4)$$

This is strictly equivalent to encode a full bilinear interaction of projections of q and v into a latent pair representation $\mathbf{z}$, and to use this latent code to predict the correct answer. If we define $\tilde{\mathbf{q}} = \mathbf{q}^\top \boldsymbol{W}_q \in \mathbb{R}^{t_q}$ and $\tilde{\mathbf{v}} = \mathbf{v}^\top \boldsymbol{W}_v \in \mathbb{R}^{t_v}$, we have:

$$\mathbf{z} = (\mathcal{T}_c \times_1 \tilde{\mathbf{q}}) \times_2 \tilde{\mathbf{v}} \in \mathbb{R}^{t_o} \quad (5)$$

$\mathbf{z}$ is projected into the prediction space $\mathbf{y} = \mathbf{z}^\top \boldsymbol{W}_o \in \mathbb{R}^{|\mathcal{A}|}$ and $\mathbf{p} = \mathrm{softmax}(\mathbf{y})$. In our experiments, we use non-linearities $\tilde{\mathbf{q}} = \tanh(\mathbf{q}^\top \boldsymbol{W}_q)$ and $\tilde{\mathbf{v}} = \tanh(\mathbf{v}^\top \boldsymbol{W}_v)$ in the fusion, as in [8], providing slightly better results. The multimodal Tucker fusion is depicted in Figure 2.

Interpretation Using the Tucker decomposition, we have separated $\mathcal{T}$ into four components, each having a specific role in the modeling. Matrices $\boldsymbol{W}_q$ and $\boldsymbol{W}_v$ project the question and the image vectors into spaces of respective dimensions t_q and t_v. These dimensions directly impact the modeling complexity that will be allowed for each modality. The higher t_q (resp. t_v) will be, the more complex the question (resp. image) modeling will be. Tensor $\mathcal{T}_c$ is used to model interactions between $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{v}}$. It learns a projection from all the correlations $\tilde{\mathbf{q}}[i]\tilde{\mathbf{v}}[j]$ to a vector $\mathbf{z}$ of size t_o. This dimension controls the complexity allowed for the *interactions* between modalities. Finally, the matrix $\boldsymbol{W}_o$ scores this pair embedding $\mathbf{z}$ for each class in $\mathcal{A}$.

3.3. Tensor sparsity

To further balance between expressivity and complexity of the interactions modeling, we introduce a structured sparsity constraint based on the rank of the slice matrices in $\mathcal{T}_c$. When we perform the t_o bilinear combinations between $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{v}}$ of Eq. (5), each dimension $k \in [\![1, t_o]\!]$ in $\mathbf{z}$ can be written as:

$$\mathbf{z}[k] = \tilde{\mathbf{q}}^\top \mathcal{T}_c[:,:,k] \tilde{\mathbf{v}} \quad (6)$$

The correlations between elements of $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{v}}$ are weighted by the parameters of $\mathcal{T}_c[:,:,k]$. We might benefit from the introduction of a structure in each of these slices. This structure can be expressed in terms of rank constraints on the slices of $\mathcal{T}_c$. We impose the rank of each slice to be equal to a constant R. Thus we express each slice $\mathcal{T}_c[:,:,k]$ as a sum of R rank one matrices:

$$\mathcal{T}_c[:,:,k] = \sum_{r=1}^{R} \mathbf{m}_r^k \otimes \mathbf{n}_r^{k\top} \quad (7)$$

with $\mathbf{m}_r^k \in \mathbb{R}^{t_q}$ and $\mathbf{n}_r^k \in \mathbb{R}^{t_v}$ Eq. (6) becomes:

$$z[k] = \sum_{r=1}^{R} \left(\tilde{\mathbf{q}}^\top \mathbf{m}_r^k \right) \left(\tilde{\mathbf{v}}^\top \mathbf{n}_r^k \right) \quad (8)$$

We can define R matrices $\boldsymbol{M}_r \in \mathbb{R}^{t_q \times t_o}$ (resp. $\boldsymbol{N}_r \in \mathbb{R}^{t_v \times t_o}$) such as $\forall k \in [\![1, d_o]\!], M_r[:,k] = \mathbf{m}_r^k$ (resp. $N_r[:,k] = \mathbf{n}_r^k$). The structured sparsity on $\mathcal{T}_c$ can then be written as:

$$\mathbf{z} = \sum_{r=1}^{R} \mathbf{z}_r \quad (9)$$

$$\mathbf{z}_r = (\tilde{\mathbf{q}}^\top \boldsymbol{M}_r) * (\tilde{\mathbf{v}}^\top \boldsymbol{N}_r) \quad (10)$$

Interpretation Adding this rank constraint on $\mathcal{T}_c$ leads to expressing the output vector $\mathbf{z}$ as a sum over R vectors $\mathbf{z}_r$. To obtain each of these vectors, we project $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{v}}$ into a common space and merge them with an elementwise product. Thus, we can interpret $\mathbf{z}$ as modeling an OR interaction over multiple AND gates (R in MUTAN) between projections of $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{v}}$. $\mathbf{z}[k]$ can described in terms of logical operators as:

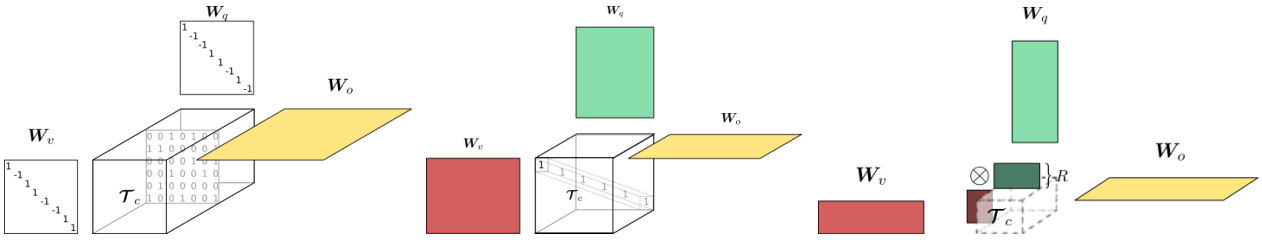
$$\mathbf{z}_r[k] = \left(\tilde{\mathbf{q}} \text{ similar to } \mathbf{m}_r^k \right) \text{ AND } \left(\tilde{\mathbf{v}} \text{ similar to } \mathbf{n}_r^k \right) \quad (11)$$

$$\mathbf{z}[k] = \mathbf{z}_1[k] \text{ OR ... OR } \mathbf{z}_R[k] \quad (12)$$

This decomposition gives a very clear insight of how the fusion is carried out in our MUTAN model. In our experiments, we will show how different r's in $[\![1, R]\!]$ behave, depending on the type of question. We will exhibit some cases where some r's specialize over specific question types.

3.4. Model Unification and Discussion

In this subsection, we show how two state of the art models, namely Multimodal Low-rank bilinear pooling [8] (MLB) and Multimodal Compact Bilinear pooling [5] (MCB), can be seen as special cases of our Multimodal Tucker Fusion. Each of these models use a different type of bilinear interaction between $\mathbf{q}$ and $\mathbf{v}$, hence instantiating a specific parametrization of the weight tensor $\mathcal{T}$. These parameterizations actually consist in a Tucker decomposition with specific constraints on the elements $\mathcal{T}_c, \boldsymbol{W}_q, \boldsymbol{W}_v$ and $\boldsymbol{W}_o$. More importantly, when we cast MCB and MLB



(a) MCB (b) MLB (c) MUTAN

Figure 3: Tensor design strategies. (a) MCB: $\boldsymbol{W}_q$ and $\boldsymbol{W}_v$ are fixed diagonal matrices, $\boldsymbol{\mathcal{T}}^c$ is a sparse fixed tensor, only the output factor matrix $\boldsymbol{W}_o$ is learnt; (b) MLB: the 3 factor matrices are learnt but the core tensor is $\boldsymbol{\mathcal{T}}^c$ set to identity; (c) MUTAN: $\boldsymbol{W}_q$, $\boldsymbol{W}_v$, $\boldsymbol{W}_o$ and $\boldsymbol{\mathcal{T}}^c$ are learnt. The full bilinear interaction $\boldsymbol{\mathcal{T}}^c$ is structured with a low-rank (R) decomposition.

into the framework of Tucker decompositions, we show that the structural constraints imposed by these two models state that some parameters are fixed, while they are free to be learnt in our full Tucker fusion. This is illustrated in Figure 3. We show in color the learnt parameters.

3.4.1 Multimodal Compact Bilinear (MCB)

We can show that the Multimodal Compact Bilinear pooling [5] can be written as a bilinear model where the weight tensor $\boldsymbol{\mathcal{T}}^{mcb}$ is decomposed into its Tucker decomposition, with specific structures on the decompositions' elements. The intramodal projection matrices $\boldsymbol{W}_q^{mcb}$ and $\boldsymbol{W}_v^{mcb}$ are diagonal matrices where the non-zero coefficients take their values in $\{-1; 1\}$: $\boldsymbol{W}_q^{mcb} = \mathbf{Diag}(\mathbf{s}_q)$ and $\boldsymbol{W}_v^{mcb} = \mathbf{Diag}(\mathbf{s}_v)$, where $\mathbf{s}_q \in \mathbb{R}^{d_q}$ and $\mathbf{s}_v \in \mathbb{R}^{d_v}$ are random vectors sampled at the instanciation of the model but kept fixed afterwards. The core tensor $\boldsymbol{\mathcal{T}}_c$ is sparse and its values follow the rule: $\boldsymbol{\mathcal{T}}_c^{mcb}[i, j, k] = 1$ if $h(i, j) = k$ (and 0 else), where $h : [\![1, d_q]\!] \times [\![1, d_v]\!] \rightarrow [\![1, d_o]\!]$ is randomly sampled at the beginning of training and no longer changed.

As was noticed in [8], all the learnt parameters in MCB are located *after* the fusion. The combinations of dimensions from $\mathbf{q}$ and from $\mathbf{v}$ that are supposed to interact with each other are randomly sampled beforehand (through h). To compensate for the fact of fixing the parameters $\mathbf{s}_q, \mathbf{s}_v$ and h, they must set a very high t_o dimension (typically 16,000). This set of combinations is taken as a feature vector for classification.

3.4.2 Multimodal Low-rank Bilinear (MLB)

The low-rank bilinear interaction corresponds to a canonical decomposition of the tensor $\boldsymbol{\mathcal{T}}$ such as its rank is equal to R. It is well-known that the low-rank decomposition of a tensor is a special case of the Tucker decomposition, such as $\boldsymbol{\mathcal{T}}^{mlb} = [\![\boldsymbol{\mathcal{I}}_R; \boldsymbol{W}_q, \boldsymbol{W}_v, \boldsymbol{W}_o]\!]$ where $t_q = t_v = t_o = R$. Two major constraints are imposed when reducing Tucker decomposition to low-rank decomposition. First, the three dimensions t_q, t_v and t_o are structurally set to be equal. The dimension of the space in which a modality is projected (t_q and t_v) quantifies the model's complexity. Our intuition is that since the image and language spaces are different, they may require to be modeled with different levels of complexity, hence different projection dimensions. The second constraint is on the core tensor, which is set to be the identity. A dimension k of $\tilde{\mathbf{q}}^{mlb}$ is only allowed to interact with the same dimension of $\tilde{\mathbf{v}}^{mlb}$, which might be restrictive. We will experimentally show the beneficial effect of removing these constraints.

We would like to point out the differences between MLB and the structured sparsity per slice presented in 3.3. There are two main differences between the two approaches. First, our rank reduction is made on the core tensor of the Tucker decomposition $\boldsymbol{\mathcal{T}}_c$, while in MLB they constrain the rank of the global tensor $\boldsymbol{\mathcal{T}}$. This lets us keep different dimensionalities for the projected vectors $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{v}}$. The second difference is we do not reduce the tensor on the third mode, but only on the first two modes corresponding to the image and question modalities. The implicit parameters in $\boldsymbol{\mathcal{T}}_c$ are correlated *inside* a mode-3 slice but independent *between* the slices.

4. Experiments

VQA Dataset The VQA dataset [2] is built over images of MSCOCO [15], where each image was manually annotated with 3 questions. Each one of these questions is then answered by 10 annotators, yielding a list of 10 ground-truth answers. The dataset is composed of 248,349 pairs (image, question) for the training set, 121,512 for validation and 244,302 for testing. The ground truth answers are given for the *train* and *val* splits, and one must submit their predictions to an evaluation server to get the scores on *test-std* split. Note that the evaluation server makes it possible to submit multiple models per day on *test-dev*, which is a subsample of *test-std*. The whole submission on *test-std* can only be done *five* times. We focus on the open-ended task, where the ground truth answers are given in free natural language phrases. This dataset comes with its evaluation

metric, presented in [2]. When the model predicts an answer for a visual question, the VQA accuracy is given by:

$$\min\left(1, \frac{\#\text{ humans that provided that answer}}{3}\right) \quad (13)$$

If the predicted answer appears at least 3 times in the ground truth answers, the accuracy for this example is considered to be 1. Intuitively, this metrics takes into account the consensus between annotators.

MUTAN Setup We first resize our images to be of size $(448, 448)$. We use ResNet152 [6] as our visual feature extractor, which produces feature maps of size $14 \times 14 \times 2048$. We keep the 14×14 tiling when attention models are used (section 4.2). Otherwise, the image is represented as the average of 14×14 vectors at the output of the CNN (section 4.1). To represent questions, we use a GRU [4] initialized with the parameters of a pretrained Skip-thoughts model [11]. Each model is trained to predict the most common answer in the 10 annotated responses. $|\mathcal{A}|$ is fixed to the 2000 most frequent answers as in [8], and we train our model using ADAM [9] (see details in supplementary material).

4.1. Fusion Scheme Comparison

To point out the performance variation due to the fusion modules, we first compare MUTAN to state-of-the-art bilinear models, under the same experimental framework. We do not use attention models here. Several merging scheme results are presented in Table 1: Concat denotes a baseline where $\mathbf{v}$ and $\mathbf{q}$ are merged by simply concatenating them. For MCB [5] and MLB [8], we use the available code [2] [3] to train models on the same visual and linguistic features. We choose an output dimension of 16,000 for MCB and 1,200 for MLB, as indicated in the respective articles. MUTAN_noR designates the MUTAN model without the rank sparsity constraint. We choose all the projection dimensions to be equal to each other: $t_q = t_v = t_o = 160$. These parameters are chosen considering the results on *val* split. Finally, our MUTAN [4] designates the full Tucker decomposition with rank sparsity strategy. We choose all the projection dimensions to be equal to each other: $t_q = t_v = t_o = 360$, and a rank $R = 10$. These parameters were chosen so that MUTAN and MUTAN_noR have the same number of parameters. As we can see in Table 1, MUTAN_noR performs better than MLB, which validates the fact that modeling full bilinear interactions between low dimensional projections yields a more powerful representation than having strong mono-modal transformations with a simple fusion scheme (element-wise product). With the structured sparsity, MUTAN obtains the best results, validating our intuition of having a nice tradeoff between the projection dimensions and a reasonable number of useful bilinear interaction parameters in the core tensor $\mathcal{T}_c$. Finally, a naive late fusion MUTAN+MLB further improves performances (about +1pt on *test-dev*). It validates the complementarity between the two types of tensor decomposition.

[2] https://github.com/jnhwkim/cbp

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

[4] https://github.com/cadene/vqa.pytorch

		test-dev				*val*
Model	Θ	Y/N	No.	Other	All	All
Concat	8.9	79.25	36.18	46.69	58.91	56.92
MCB	32	80.81	35.91	46.43	59.40	57.39
MLB	7.7	**82.02**	36.61	46.65	60.08	57.91
MUTAN_noR	4.9	81.44	36.42	46.86	59.92	57.94
MUTAN	4.9	81.45	**37.32**	**47.17**	**60.17**	**58.16**
MUTAN+MLB	17.5	82.29	37.27	48.23	61.02	58.76

Table 1: Comparison between different fusion under the same setup on the *test-dev* split. Θ indicates the number of learnable parameters (in million).

4.2. State-of-the-art comparison

To compare the performance of the proposed approach to state-of-the-art works, we associate the MUTAN fusion with recently introduced techniques for VQA, which are described below.

Attention mechanism We use the same kind of multi-glimpse attention mechanisms as the ones presented in [5] and [8]. We use MUTAN to score the region embeddings according to the question vector, and compute a global visual vector as a sum pooling weighted by these scores.

Answer sampling (Ans. Sampl.) Each (image,question) pair in the VQA dataset is annotated with 10 ground truth answers, corresponding to the different annotators. In those 10, we keep only the answers occuring more than 3 times, and randomly choose the one we ask our model to predict.

Data augmentation (DAVG) We use Visual Genome [13] as a data augmentation to train our model, keeping only the example whose answer is in our vocabulary. This triples the size of our training set.

Ensembling MUTAN (5) consist in an ensemble of five models trained on *train+val* splits. We use 3 attentional MUTAN architectures with one trained with additional Visual Genome data. The 2 other models are instances of MLB, which can be seen as a special case of MUTAN. Details about the ensembling will be provided in the supplementary material.

Results State-of-the-art comparison results are gathered in Table 2. Firstly, we can notice that bilinear models, *i.e.*

	test-dev				*test-std*
	Y/N	No.	Other	All	All
SMem 2-hop [27]	80.87	37.32	43.12	57.99	58.24
Ask Your Neur. [20]	78.39	36.45	46.28	58.39	58.43
SAN [30]	79.3	36.6	46.1	58.7	58.9
D-NMN [1]	81.1	38.6	45.5	59.4	59.4
ACK [26]	81.01	38.42	45.23	59.17	59.44
MRN [7]	82.28	38.82	49.25	61.68	61.84
HieCoAtt [17]	79.7	38.7	51.7	61.8	62.1
MCB (7) [5]	83.4	39.8	58.5	66.7	66.5
MLB (7) [8]	84.57	39.21	57.81	66.77	66.89
MUTAN (3)	84.54	39.32	57.36	67.03	66.96
MUTAN (5)	**85.14**	**39.81**	**58.52**	**67.42**	**67.36**

Table 2: MUTAN performance comparison on the test-dev and test-standard splits VQA dataset; (n) for an ensemble of n models.

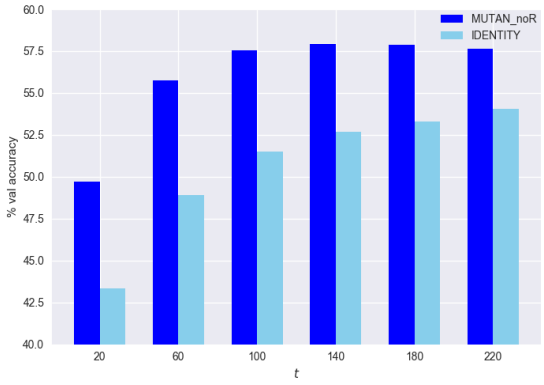


Figure 4: The improvements given by MUTAN_noR over a model trained with the identity tensor as a fusion operator between $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{v}}$.

MCB [5] and MLB [8] have a strong edge over other methods with a less powerful fusion scheme.

MUTAN outperforms all the previous methods with a large margin on *test-dev* and *test-std*. This validates the relevance of the proposed fusion scheme, which models precise interactions between modalities. The good performances of MUTAN (5) also confirms its complementarity with MLB, already seen in section 4.1 without attention mecanism: MLB learns informative mono-modal projections, wheras MUTAN is explicitly devoted to accurately models bilinear interactions. Finally, we can notice that the performance improvement of MUTAN in this enhanced setup is conform to the performance gap reported in section 4.1, showing that the benefit of the fusion scheme directly translates for the whole VQA task.

Finally, we also evaluated an ensemble of 3 models based on the MUTAN fusion scheme (without MLB), that we denote as MUTAN (3). This ensemble also outperforms state-of-the-art results. We can point out that this improvement is reached with is an ensembling of 3 models, which is smaller than the previous state-of-the-art MLB results containing an ensembling of 7 models.

4.3. Further analysis

Experimental setup In this section, we study the behavior of MUTAN under different conditions. Here, we examine under different aspects the fusion between $\mathbf{q}$ and $\mathbf{v}$ with the Tucker decomposition of tensor $\mathcal{T}$. As we did previously, we don't use the attention mechanism in this section. We only consider a global visual vector, computed as the average of the 14×14 region vectors given by our CNN. We also don't use the answer sampling, asking our model to always predict the most frequent answer of the 10 ground truth responses. All the models are trained on the VQA *train* split, and the scores are reported on *val*.

Impact of a plain tensor The goal is to see how important are all the parameters in the core tensor $\mathcal{T}_c$, which model the correlations between projections of $\mathbf{q}$ and $\mathbf{v}$. We train multiple MUTAN_noR, where we fix all projection dimensions to be equal $t_q = t_v = t_o = t$ and t ranges from 20 to 220. In Figure 4, we compare these MUTAN_noR with a model trained with the same projection dimension, but where $\mathcal{T}_c$ is replaced by the identity tensor[5]. One can see that MUTAN_noR gives much better results than identity tensor, even for very small core tensor dimensions. This shows that MUTAN_noR is able to learn powerful correlations between modalities[6].

Impact of rank sparsity We want to study the impact of introducing the rank constraint in the core tensor $\mathcal{T}_c$. We fix the input dimensions $t_q = 210$ and $t_v = 210$, and vary the output dimension t_o for multiple rank constraints R. As we can see in Figure 5, controlling the rank of slices in $\mathcal{T}_c$ allows to better model the interactions between the unimodal spaces. The different colored lines show the behavior of MUTAN for different values of R. Comparing $R = 60$ (blue line) and $R = 20$ (green line), we see that a lower rank allows to reach higher values of t_o without overfitting. The number of parameters in the fusion is lower, and the accuracy on the *val* split is higher.

[5]This is strictly equivalent to MLB [8] without attention. However, we are fully aware that it takes between 1000 and 2000 dimensions of projection to be around the operating point of MLB. With our experimental setup, we just focus on the effect of adding parameters to our fusion scheme.

[6]Notice that for each t, MUTAN_noR has t^3 parameters. For instance, for $t = 220$, MUTAN adds 10.6M parameters over identity.

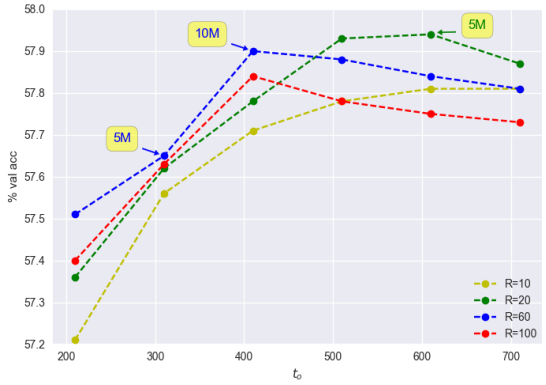
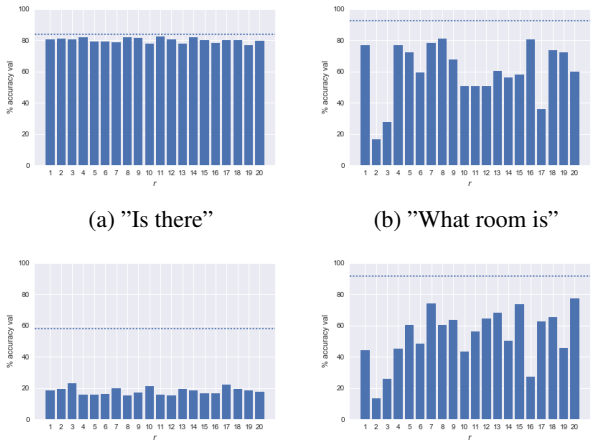


Figure 5: Accuracy on VQA *val* in function of t_o. Each colored dot shows the score of a MUTAN model trained on *train*. The yellow labels indicate the number of parameters in the fusion.

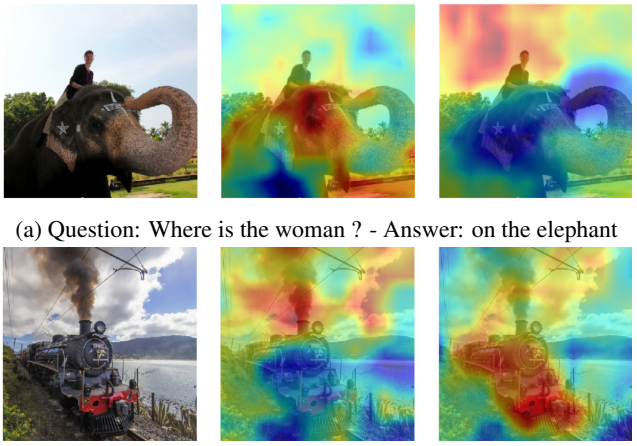


(a) "Is there" (b) "What room is"

(c) "What is the man" (d) "What sport is"

Figure 6: Visualizing the performances of ablated systems according to the R variables. Full system performance is denoted in dotted line.

Qualitative observations In MUTAN, the vector $\mathbf{z}$ that encodes the (image,question) pair is expressed as a sum over R vectors $\mathbf{z}_r$. We want to study the R different latent projections that have been learnt during training, and assess whether the representations have captured different semantic properties of inputs. We quantify the differences between each of the R spaces using the VQA question types. We first train a model on the *train* split, with $R = 20$, and measure its performance on the *val* set. Then, we set to 0 all of the $\mathbf{z}_r$ vectors except one, and evaluate this ablated system on the validation set. In Figure 6, we compare the full system to the R ablated systems for 4 different question types. The dotted line shows the accuracy of the full system, while the different bars show the accuracy of the ablated system for each R. Depending on the question type, we observe 3 different behaviors of the ranks. When the question type's answer support is small, we observe that each rank has learnt enough to reach almost the same accuracy as the global system. This is the case for questions starting by "Is there", whose answer is almost always "yes" or "no". Other question types require information from all the latent projections, as in the case of "What is the man". This leads to cases where all projections perform equally and significantly worst when taken individually than when combined to get the full model. At last, we observe that specific projections contribute more than others depending on the question type. For example, latent variable 16 performs well on "what room is", and is less informative to answer questions starting by "what sport is". The opposite behavior is observed for latent variable 17.



(a) Question: Where is the woman ? - Answer: on the elephant

(b) Question: Where is the smoke coming from ? - Answer: train

Figure 7: The original image is shown on the left. The center and right images show heatmaps obtained when turning off all the projections but one, for two different projections. Each projection focuses on a specific concept needed to answer the question.

We run the same kind of analysis for the MUTAN fusion in the attention mechanism. In Figure 7, we show for two images the different attentions that we obtain when turning off all the projections but one. For the first image, we can see that a projection focuses on the elephant, while another focuses on the woman. Both these visual informations are necessary to answer the question "Where is the woman ?". The same behavior is observed for the second image, where a projection focuses on the smoke while another gives high attention to the train.

5. Conclusion

In this paper, we introduced our MUTAN strategy for the VQA task. Our main contribution is a multimodal fusion between visual and textual information using a bilinear framework. Our model combines a Tucker decomposition with a low-rank matrix constraint. It is designed to control the full bilinear interaction's complexity. MUTAN factorizes the interaction tensor into interpretable elements, and allows an easy control of the model's expressiveness. We also show how the Tucker decomposition framework generalizes the most competitive VQA architectures. MUTAN is evaluated on the most recent VQA dataset, reaching state-of-the-art.

References

[1] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Learning to compose neural networks for question answering. In *NAACL HLT 2016, San Diego California, USA, June 12-17, 2016*, pages 1545–1554, 2016.

[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual Question Answering. In *ICCV*, 2015.

[3] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages and Programming*, pages 693–703, 2002.

[4] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, EMNLP 2014*, pages 103–111, 2014.

[5] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv:1606.01847*, 2016.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[7] J.-H. Kim, S.-W. Lee, D. Kwak, M.-O. Heo, J. Kim, J.-W. Ha, and B.-T. Zhang. Multimodal Residual Learning for Visual QA. In *NIPS*, pages 361–369, 2016.

[8] J.-H. Kim, K.-W. On, J. Kim, J.-W. Ha, and B.-T. Zhang. Hadamard Product for Low-rank Bilinear Pooling. In *5th International Conference on Learning Representations*, 2017.

[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[10] R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal neural language models. In *ICML*, pages 595–603, 2014.

[11] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-thought vectors. In *NIPS*, pages 3294–3302, 2015.

[12] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, Aug. 2009.

[13] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016.

[14] R. Li and J. Jia. Visual question answering with question representation update (qru). In *NIPS*, pages 4655–4663. 2016.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[16] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 2015.

[17] J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, pages 289–297, 2016.

[18] L. Ma, Z. Lu, L. Shang, and H. Li. Multimodal convolutional neural networks for matching image and sentence. In *ICCV*, pages 2623–2631, 2015.

[19] M. Malinowski and M. Fritz. Towards a visual turing challenge. In *Learning Semantics (NIPS workshop)*, December 2014.

[20] M. Malinowski, M. Rohrbach, and M. Fritz. Ask your neurons: A deep learning approach to visual question answering. *arXiv:1605.02697*, 2016.

[21] M. Ren, R. Kiros, and R. S. Zemel. Exploring models and data for image question answering. In *NIPS*, pages 2953–2961, 2015.

[22] K. J. Shih, S. Singh, and D. Hoiem. Where to look: Focus regions for visual question answering. In *CVPR*, 2016.

[23] R. Socher, A. Karpathy, Q. Le, C. Manning, and A. Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218, 2014.

[24] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[25] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, pages 3156–3164, 2015.

[26] Q. Wu, P. Wang, C. Shen, A. Dick, and A. van den Hengel. Ask me anything: free-form visual question answering based on knowledge from external sources. In *CVPR*, 2016.

[27] H. Xu and K. Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *ECCV*, pages 451–466, 2016.

[28] 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*, pages 2048–2057, 2015.

[29] F. Yan and K. Mikolajczyk. Deep correlation for matching images and text. In *CVPR*, June 2015.

[30] Z. Yang, X. He, J. Gao, L. Deng, and A. J. Smola. Stacked attention networks for image question answering. In *CVPR*, pages 21–29, 2016.

[31] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei. Visual7W: Grounded Question Answering in Images. In *CVPR*, 2016.

Supplementary material

Preprocessing details

Image As in [5] (MCB) or [8] (MLB), we preprocess the images before training our VQA models as follow. We load and rescal the image to 448. It is important to notice that we keep the proportion. Thus, 448 will be the size of the smaller edge. Then, we crop the image at the center to have a region of size 448×448. We normalize the image using the ImageNet normalization. Finally, we feed the image to a pretrained ResNet-152 and extract the features before the last Rectified Linear Unit (ReLU).

Question We use almost the same preprocessing as [5] or [8] for the questions. We keep the questions which are associated to the 2000 most occuring answers. We convert the questions characters to lower case and remove all the ponctuations. We use the space character to split the question into a sequence of words. Then, we replace all the words which are not in the vocabulary of our pretrained Skip-thoughts model by a special "unknown" word ("UNK"). Finally, we pad all the sequences of words with zero-padding to match the maximum sequence length of 26 words. We use TrimZero as in [8] to avoid the zero values from the padding.

Optimization details

Algorithm It is important to notice that we use the classical implementation of Adam [1] with a learning rate of 10^{-4} unlike in [5] or [8]. In fact, we tried RMSPROP, SGD Nesterov and Adam with or without learning rate decay. We found that Adam without learning rate decay was more convenient and lead to the same accuracy.

Batch size During the optimization process, we use a batch size of 512 for the models without an attention modeling. For the others, we use a batch size of 100, because the models are more memory consuming.

Early stopping As in [5] and [8], we use early stopping as a regularizer. During our training process, we save the model parameters after each epoch. To evaluate our model on the evaluation server, we chose the best epoch according to the Open Ended validation accuracy computed on the *val* split when available.

As in [5] and [8], for the models trained on the *trainval* split, we use the *test-dev* split as a validation set and are obliged to submit several times on the evaluation server. Note that we are limited to 10 submissions per day. In practice, we submit 3 to 4 times per models for epochs associated to training accuracies between 63% to 70%.

Ensemble details

In table 3, we report several single models which compose our two ensembles. MUTAN(3) is made of a MUTAN trained on the *trainval* split with 2 glimpses, an other MUTAN with 3 glimpses and a third MUTAN with 2 glimpses trained on the *trainval* split with the visual genome data augmentation. All three have been trained with the same hyper-parameters besides the number of glimpses.

MUTAN(5) is made of the three same MUTAN models of MUTAN(3) and two MLB models which can be viewed as a special case of our Multimodal Tucker Fusion. The first MLB has 2 glimpses and was trained on the *trainval* split. It has been made available by the authors of [8] [2]. The second MLB has 4 glimpses and was trained by ourself on the *trainval* split with the visual genome data augmentation.

The final results of both ensembles are obtained by averaging the features extracted before the final Softmax layer of all their models.

Scores details

In table 3, we provide the scores for each answer type processed on the *val* and *test-dev* splits. In table 4, we provide the same scores for *test-dev* and *test-standard*.

*Equal contribution

[1] https://github.com/torch/optim/blob/master/adam.lua

[2] https://github.com/jnhwkim/MulLowBiVQA/tree/master/model

		test-dev				*val*			
Model	Θ	Y/N	No.	Other	All	Y/N	No.	Other	All
Concat	8.9	79.25	36.18	46.69	58.91	80.01	33.72	45.46	56.92
MCB	32	80.81	35.91	46.43	59.40	81.61	33.94	45.14	57.39
MLB	7.7	**82.02**	36.61	46.65	60.08	**82.36**	34.35	45.54	57.91
MUTAN_noR	4.9	81.44	36.42	46.86	59.92	82.28	35.07	45.48	57.94
MUTAN	4.9	81.45	**37.32**	**47.17**	**60.17**	82.07	**35.16**	**46.03**	**58.16**
MUTAN+MLB	17.5	82.29	37.27	48.23	61.02	82.59	35.21	46.84	58.76

Table 3: Comparison between different fusion under the same setup on the *test-dev* split. Θ indicates the number of learnable parameters (in million).

	test-dev				*test-std*			
	Y/N	No.	Other	All	Y/N	No.	Other	All
SMem 2-hop [27]	80.87	37.32	43.12	57.99	80.0	37.53	43.48	58.24
Ask Your Neur. [20]	78.39	36.45	46.28	58.39	78.24	36.27	46.32	58.43
SAN [30]	79.3	36.6	46.1	58.7	-	-	-	58.9
D-NMN [1]	81.1	38.6	45.5	59.4	-	-	-	59.4
ACK [26]	81.01	38.42	45.23	59.17	81.07	37.12	45.83	59.44
MRN [7]	82.28	38.82	49.25	61.68	82.39	38.23	49.41	61.84
HieCoAtt [17]	79.7	38.7	51.7	61.8	-	-	-	62.1
MCB (7) [5]	83.4	39.8	58.5	66.7	83.2	39.5	58.0	66.5
MLB (7) [8]	84.54	39.21	57.81	66.77	84.61	39.07	57.79	66.89
MUTAN (3)	84.57	39.32	57.36	67.03	84.39	38.70	58.20	66.96
MUTAN (5)	**85.14**	**39.81**	**58.52**	**67.42**	**84.91**	**39.79**	**58.35**	**67.36**

Table 4: MUTAN performance comparison on the *test-dev* and *test-standard* splits VQA dataset; (n) for an ensemble of n models.