Visual question answering with a hybrid convolution recurrent model

Philipp Harzig, Christian Eggert, Rainer Lienhart

Angaben zur Veröffentlichung / Publication details:

Harzig, Philipp, Christian Eggert, and Rainer Lienhart. 2018. "Visual question answering with a hybrid convolution recurrent model." In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval - ICMR '18, June 11 - 14, 2018, Yokohama, Japan*, 318–25. New York, USA: ACM Press. https://doi.org/10.1145/3206025.3206054.

Visual Question Answering With a Hybrid Convolution Recurrent Model

Philipp Harzig
University of Augsburg
Augsburg, Germany
philipp.harzig@informatik.uni-augsburg.de

Christian Eggert
University of Augsburg
Augsburg, Germany
christian.eggert@informatik.uni-augsburg.de

Rainer Lienhart
University of Augsburg
Augsburg, Germany
rainer.lienhart@informatik.uni-augsburg.de

ABSTRACT

Visual Question Answering (VQA) is a relatively new task, which tries to infer answer sentences for an input image coupled with a corresponding question. Instead of dynamically generating answers, they are usually inferred by finding the most probable answer from a fixed set of possible answers. Previous work did not address the problem of finding all possible answers, but only modeled the answering part of VQA as a classification task. To tackle this problem, we infer answer sentences by using a Long Short-Term Memory (LSTM) network that allows us to dynamically generate answers for (image, question) pairs. In a series of experiments, we discover an end-to-end Deep Neural Network structure, which allows us to dynamically answer questions referring to a given input image by using an LSTM decoder network. With this approach, we are able to generate both less common answers, which are not considered by classification models, and more complex answers with the appearance of datasets containing answers that consist of more than three words.

CCS CONCEPTS

• **Information systems** → **Multimedia and multimodal retrieval**; • **Computing methodologies** → **Natural language generation**; **Speech recognition**; **Scene understanding**; **Neural networks**; *Image representations*;

KEYWORDS

VQA, Visual Question Answering, multimodal retrieval, natural language generation, LSTM, multimodal fusion

ACM Reference Format:
Philipp Harzig, Christian Eggert, and Rainer Lienhart. 2018. Visual Question Answering With a Hybrid Convolution Recurrent Model. In *ICMR '18: 2018 International Conference on Multimedia Retrieval, June 11–14, 2018, Yokohama, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3206025.3206054

1 INTRODUCTION

In recent years, the machine learning community rapidly adapted neural networks. Especially, Deep Convolutional Neural Networks (DCNNs) developed into the main method for extracting representations from images. Over time, the models got deeper, more complex [10, 22, 23] and achieved higher accuracies in the ImageNet Challenge [21]. Because these models are trained on a very large dataset of images, they learn to extract a good feature representation of images and, thus, are also used for other problems than image classification. Additionally, methods for interpreting language in form of sentences were also not unaffected by the advances in neural networks. Recurrent Neural Networks (RNNs) have become the standard in extracting information from text or creating sentences. Particularly, a special form of RNNs called Long Short-Term Memory (LSTM) [12] networks are nowadays used widely in natural language processing.

A new task called Visual Question Answering (VQA) emerged lately. In VQA, an image and a question regarding that image are the inputs, and a sentence answering the question is the desired output. In other words, the VQA task is a model that answers questions asked for an input image. This task is challenging, because a good understanding both of images and natural language is essential to be able to predict the answer of a question asked regarding the contents of the image. Its difficulty lies in fusing feature representations of the image and question into a joint representation, which then makes it possible to generate an answer for that very image-question pair. Unlike most other approaches, we use an LSTM network to generate the answers for the VQA task. The general practice is to extract the most common answers from the training set and predict the answers by choosing the most likely answer out of a fixed predefined set. Teney et al. [24] treat VQA as a multi-label classification task, i.e., for a given image and question the answer is predicted from a fixed set of answers. This strategy delivers good scores as the official evaluation metric is the accuracy. Hence, a fixed set of the most common answers has a greater influence on the score than less common answers. We tackle the serious drawback of leaving out the less common answers altogether. We deal with this problem by constructing the answer with an LSTM network, which can produce answers out of the entire vocabulary corpus.

Our main contributions are as follows: (1) We use a CNN network to extract features from the input question, which has been shown to work better than an LSTM encoding in our case. (2) We compare different approaches, which embed the question and image features into a joint semantic space. (3) We do not see the VQA task as a classification problem but rather we see it as a more complex problem that can not be approximated by some 3000 possible answers.

Natural language is more complex than a set of predefined answers, thus, we replace the fully-connected classification layer with an LSTM network, that can produce arbitrary sentences. Finally, (4) we find out that our architecture is able to produce new answers not contained in the training set at all, some of which we show in a qualitative analysis.

2 RELATED WORK

When it comes to VQA, we face the challenge to process image data properly, process natural language and correctly fuse them in order to answer the questions regarding the image accurately. In this section, we give an overview of related work in all of those research areas.

Natural language processing. Natural language processing is a long-studied topic in computer vision. Barnard et al. [3] present a statistical model that learns relationships between visual information from images and text information coming with the images. Farhadi et al. [6] infer triplets consisting of object, action and scene from handcrafted image features and transfer them to text afterwards. Similarly, Li et al. [17] use image features to compose sentences from scratch by using a text corpus comprised of n-grams gathered from the web. Lately, Recurrent Neural Networks (RNNs) in form of Long Short-Term Memory (LSTM) [12] networks became very popular in both understanding text and generating text.
The encoder/decoder structure used by current image captioning approaches as well as our VQA model is heavily influenced by machine translation networks that transform a sentence from one language into another. For example, Google [27] uses LSTM-driven neural networks for their translation system. Facebook [8] goes into another direction and eliminates the LSTM layers by replacing them through ordinary convolutional layers resulting in better performance both in speed and accuracy.

Image Captioning. The image captioning task is closely related to the VQA task. The goal of image captioning is to create a natural language sentence that describes a given input image. VQA can be seen as a special case of image captioning, where the input question modality is added to the model and answers instead of descriptions should be generated. Different approaches appeared over time which try to convert the content of images into text fragments. State-of-the-art approaches still use the idea of extracting features from an image and converting them into a natural language sentence. They use modern image classification DCNNs like [23] as a feature extractor. One such architecture is the Show and Tell model by Vinyals et al. [25, 26], which makes use of recurrent neural networks for sequence modeling to convert image features into human-like sentences. Also Karpathy et al. [15] utilize neural networks for sequence modeling in form of a Bidirectional Recurrent Neural Network to generate captions for images. They refine their model to additionally describe parts of the image instead of captioning the image as a whole and call this method dense captioning [13, 14]. Different approaches and their performances are reported on the website for the MSCOCO [4, 18] captioning challenge.

Joint multimodal embeddings. There are a lot of approaches that deal with combining multiple modalities in a joint embedding space. Zhou et al. [30] propose a simple VQA architecture, which concatenates image and question modalities. Other models use simple element-wise multiplication of question and image features [2, 24]. Fukui et al. [7] state that the outer product between question and image modalities are more expressive than a vector concatenation or an element-wise product and introduce a technique called Multimodal Compact Linear pooling. However, the outer product is infeasible due to its high dimensionality, hence they project the outer product into a lower dimensional space. Yu et al. [28] also use a bilinear pooling model to project the image and question into a joint embedding space (see Section 5.3).

VQA architectures. For the original VQA [2] task, many architectures have been presented. For example, Zhou et al. [30] proposed a simple architecture, that uses a simple bag-of-words model as question features and image feature vectors extracted by a CNN. The creators of the original VQA dataset themselves introduced a model that trains a classification model on the 1000 most frequent answers. They use an LSTM model for the question embedding and the VGGNet [22] as the image feature extractor DCNN. Finally, Fukui et al. [7] won the VQA challenge by using Multimodal Compact Linear pooling.
The VQA v2 [9] dataset, which is a balanced version of the VQA dataset (see Section 3 for details), makes it harder to achieve high scores. In their analysis, Goyal et al. found out that many VQA models take advantage of language priors to predict the target answers. Teney et al. [24] present an architecture that uses soft scores as ground truth targets instead of using one-hot targets as used in traditional classification. They also implement a bottom-up attention to attend to specific regions of the images. With those and other findings they won the 2017 VQA challenge. After the challenge has ended, Yu et al. [28] even surpassed Teney et al. on the VQA-v2 open ended challenge leaderboard by using the Multimodal Factorized Bilinear pooling (MFB) approach.

3 DATASET

We train and test our model on the VQA-v2 [9] dataset, which is a balanced version of the original VQA [2] dataset. The VQA dataset is built upon the MSCOCO [18] dataset and contains (question, answer) pairs for a subset of the MSCOCO images. Goyal et al. [9] balanced the VQA dataset in such a way, that for every question there are two similar images that have differing answers to that question and, therefore, containing about twice the number of questions. They also found out that models trained on the first version of the dataset exploit language priors, thus, performing worse on the VQA-v2 dataset. Their dataset contains 443K, 214K and 453K questions in the train split, validation split and test split, respectively. The test split is divided into four subsplits: test-dev, test-standard, test-challenge and test-reserve. There is an online evaluation server[1] for generated answers on the test-dev and test-standard split known as the *VQA Real Image Challenge 2017 (Open-Ended)*. On average, the VQA-v2 dataset contains 5.34 questions per image and 53.4 answers per image, i.e., every question was answered by 10 different annotators. Every question has a ground-truth of 10 answers, whereas every answer is flagged, whether the annotator was confident to

[1] https://evalai.cloudcv.org/web/challenges/challenge-page/1/overview, test-dev allows 10 submissions per day, while test-standard is limited to 5 submissions in total.

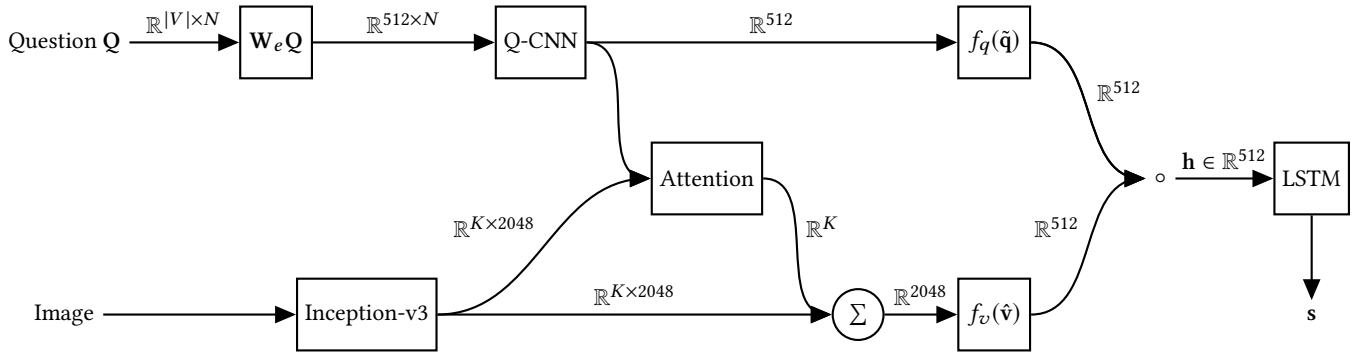

Figure 1: Architecture of our VQA model. We embed the question Q via the word embedding matrix $\mathbf{W}_e$ and extract the question features with a residual CNN. We attend to the image feature maps produced by the *Inception-v3* network with an attention mechanism producing a single weighted feature vector. We feed the image and question features through non-linear layers f and extract a joint multimodal feature vector h by element-wise multiplication.

answer the question correctly. We only use confident answers, i.e., every question has 8.11 confident answers on average.
For our studies we rebalance the train and validation splits, so we use the train split and 90% of the validation split as training set and the remaining 10% of the validation split as the validation set. We do this to have a larger training corpus available. For evaluating the performance of our models, we use the standard VQA script by Antol et al. [2]. This script calculates the overall accuracy and the accuracies over three different answer categories. These are yes/no answers (Y/N), number answers and other answers.
For some of our experiments, we extend the dataset by the popular Visual Genome [16] dataset, which contains an additional 1.7 million Visual Question Answers on 108,077 images.

4 MODEL

Our VQA model is composed of multiple components. We embed the question into a feature vector via a CNN and extract a feature vector from the corresponding image via a DCNN (*Inception-v3* [23] network in our case). We attend to the features of the input image via the question and combine both feature vectors via multimodal fusion into a single representation of the image-question pair. Instead of modeling the answering part as a classification task, we generate the answer with an LSTM network. We depict our architecture with the decoder LSTM in Figure 1. In the following, we describe each part of our architecture.

4.1 Question embedding

We add a start word <S> at the beginning and an end word </S> to the end of each question. Additionally, we tokenize each question by splitting it up into single words. We allow questions to be of any length and construct our vocabulary V from all words contained in the questions and answers (we tokenize answers the same way as we do with the questions). Each word in the question $\mathbf{Q} = (\mathbf{q}_0, \ldots, \mathbf{q}_N)$ of length N is represented by a one-hot vector $\mathbf{q}_t$, where t is the time-step or the index of the current word in the input question. We embed each word of the question into vectors

$$\hat{\mathbf{q}}_t = \mathbf{W}_e\mathbf{q}_t, t \in 0 \ldots N-1 \tag{1}$$

of dimenstion $m = 512$, where $\mathbf{W}_e \in \mathbb{R}^{512\times|V|}$.

For getting a feature representation of the whole sentence, we feed the sentence through a couple of convolutional layers. We use gated convolutions in a residual architecture like Dauphin et al. [5]. In particular, they state that CNNs do not suffer from vanishing and exploding gradients like RNNs do and, therefore, a forget gate used in RNNs like LSTMs is not required in CNNs, which process language. However, they find that allowing the network to control what information should be propagated through the layers via output gates is useful for language modeling and introduce the gated linear unit (GLU), which learns which information should be forwarded to the next layer. For each GLU we learn parameters for two convolution operations ($*$) with the same number of output feature maps m and calculate its result as

$$GLU(\mathbf{X}) = (\mathbf{X} * \mathbf{W}_{c1} + \mathbf{b}_{c1}) \circ \sigma(\mathbf{X} * \mathbf{W}_{c2} + \mathbf{b}_{c2}) \tag{2}$$

where $\mathbf{X} \in \mathbb{R}^{N\times m}$ is the input of the GLU and $\mathbf{W}_{c1} \in \mathbb{R}^{k\times m\times n}$, $\mathbf{b}_{c1} \in \mathbb{R}^n$, $\mathbf{W}_{c2} \in \mathbb{R}^{k\times m\times n}$, $\mathbf{b}_{c2} \in \mathbb{R}^n$ are learned parameters. Note that k is the kernel size of the convolution, $n = 512$ are the number of outputs and $\circ$ is the elementwise multiplication, so the output of the GLU is of dimensionality $\mathbb{R}^{N\times n}$. We visualize the architecture of a GLU in Figure 2.

We feed our embedded question through a GLU with $n = 512, k = 5$ followed by two residual blocks, each consisting out of two GLUs with a skip connection from the input to the output. We depict our question embedding architecture in Figure 3. Finally, we use global average pooling across the question length N to reduce the dimensionality of each question feature representation from $\mathbb{R}^{N\times n}$ to $\tilde{\mathbf{q}} \in \mathbb{R}^n$.

4.2 Image embedding

We also extract a feature representation from each input image with a DCNN, namely, we use the *Inception-v3* [23] architecture. We obtain a feature vector of size $K \times 2048$, where $K = 8 \times 8 = 64$

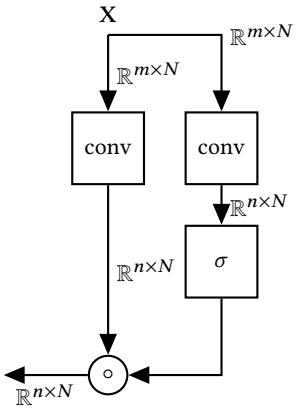

Figure 2: A gated linear unit (GLU). Inputs X are fed through a convolution layer twice with different parameter sets. One convolution layer has a sigmoid activation function while the other has no activation function. Their outputs are multiplied pointwise.

from the *Mixed_7c* layer in the case of the *Inception-v3* network. For our experiments we use (a) a global average pooled feature representation of size 2048 or (b) the feature vector of size $K \times 2048$ in the case of image attention (see Section 4.3).

4.3 Image attention

Similarly to Teney et al. [24], we use a simple form of image attention that attends to certain areas of the image. We do not use features from image regions extracted by a object detection pipeline like Faster-RCNN [20], but only attend to spatial locations on the feature map of the image. The feature map has a size of 8×8 with 2048 dimensions each, which corresponds to $K = 64$ image locations. We denote each location $i = 1 \dots K$ by $\mathbf{v}_i$ and concatenate it with the question representation $\tilde{\mathbf{q}}$ and feed them through a non-linear layer f_a (see Section 4.4) to learn a scalar attention weight $\boldsymbol{\alpha}_i$ for each image location.

$$\mathbf{a}_i = \mathbf{W}_a f_a([\mathbf{v}_i, \tilde{\mathbf{q}}]) \quad (3)$$

$$\boldsymbol{\alpha} = \text{softmax}(\mathbf{a}) \quad (4)$$

$$\hat{\mathbf{v}} = \sum_{i=1}^{K} \boldsymbol{\alpha}_i \mathbf{v}_i, \quad (5)$$

where $\mathbf{W}_a \in \mathbb{R}^{K\times 2560}$ is a learned parameter matrix. In the case (b) of no image attention, we use global average pooled feature map as described in Section 4.2 ($\hat{\mathbf{v}} = \text{avg\_pool}(\mathbf{v})$).

4.4 Multimodal fusion

We need to project the question embedding and image embedding into a joint semantic space to be able to produce answers dependent both on the input image and the question. To do this, we first reduce the dimensionality of the image embedding to match the question embedding size of $n = 512$. Again, we use the non-linear layer f from [24], which is defined as a function $f_\theta : \mathbf{x} \in \mathbb{R}^\mu \rightarrow \mathbf{y} \in \mathbb{R}^\nu$

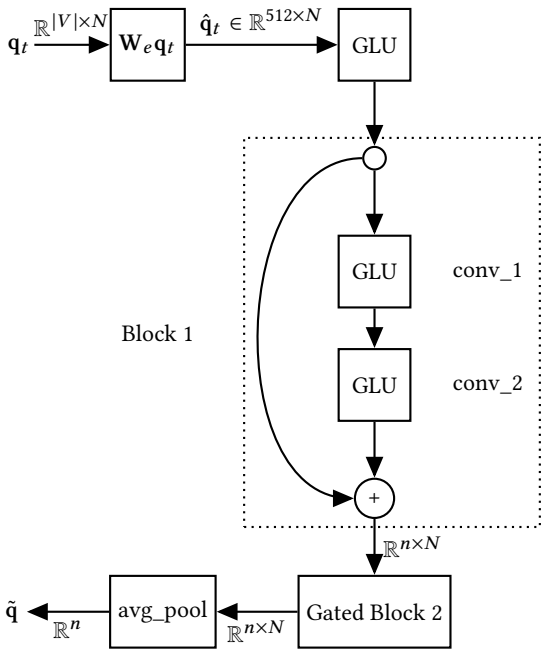

Figure 3: We extract a feature representation out of the input questions with a convolutional architecture. First, we embed each word of the question $\mathbf{q}_t$ into a lower-dimensional representation, which we feed through a GLU followed by two post-activation residual blocks. After those blocks, we use global average pooling to get one vector describing the question.

$$\tilde{\mathbf{y}} = \tanh(\mathbf{W}_\theta \mathbf{x} + \mathbf{b}_\theta) \quad (6)$$

$$\mathbf{g} = \sigma(\mathbf{W}_{\theta'} \mathbf{x} + \mathbf{b}_{\theta'}) \quad (7)$$

$$\mathbf{y} = \tilde{\mathbf{y}} \circ \mathbf{g}, \quad (8)$$

where $\mathbf{W}_\theta, \mathbf{W}_{\theta'} \in \mathbb{R}^{\nu\times\mu}$ are learned weight matrices and $\mathbf{b}_\theta, \mathbf{b}_{\theta'} \in \mathbb{R}^\nu$ are learned biases.
In particular, we use $f_v(\hat{\mathbf{v}})$ with $\nu = 512$ to reduce the dimensionality to the dimensionality of the question embedding. As common practice in the VQA community [2, 24], we then use the element-wise product to fuse image and question features with

$$\mathbf{h} = f_q(\tilde{\mathbf{q}}) \circ f_v(\hat{\mathbf{v}}). \quad (9)$$

Note, that we also apply the non-linearity function to the question embedding, where we set $\nu = \mu = 512$. We depict our model including the multimodal fusion in Figure 1.

4.5 Output LSTM

As we pointed out, most other VQA models have the drawback of leaving out the less common answers and model the problem of predicting the answer to an (image, question) pair as a classification task. To do otherwise, we use a classical approach coming from image captioning models [25] and machine translation networks. Generally speaking, those kind of models use an encoder-decoder

structure, which encodes inputs (e.g. source language sentence or input image) into a fixed-length vector representation and decodes this feature vector into the target sentence.

In our case the input is the combination of question and input image and is encoded in the vector $\mathbf{h}$ and the target is the answer described by a sentence $\mathrm{S} = (\mathbf{s}_0, \ldots, \mathbf{s}_M)$, where $\mathbf{s}_t$ is the word at time step t and M the length of the answer. Our model then maximizes the probability p of the correct answer S, given $\mathbf{h}$

$$\Theta^* = \arg\max_{\theta} \sum_{(\mathbf{h},\mathrm{S})} \log p(\mathrm{S}|\mathbf{h};\Theta), \tag{10}$$

where Θ are all variables of our model and Θ^* is the optimal parameter set. Of course, each answer S consists of a variable number of words and the model is optimized for each (question, image, answer) triple. Therefore, for each training triple, the joint log probability

$$\log(\mathrm{S}|\mathbf{h}) = \sum_{t=0}^{M} \log p(\mathbf{s}_t|\mathbf{h}, \mathbf{s}_0, \ldots, \mathbf{s}_{t-1}) \tag{11}$$

is maximized.

We can model the log probability $p_t = \log p(\mathbf{s}_t|\mathbf{h}, \mathbf{s}_0, \ldots, \mathbf{s}_{t-1})$ by using an RNN, where the probability p_t is the output of the RNN at time step t. In particular, we use an LSTM network as the decoder network, where we initialize the hidden state at time step $t = -1$ with the multimodal feature representation $\mathbf{h}$ of the question and image. Note, that we only feed $\mathbf{h}$ once into the LSTM network. The loss that defines the error signal is then given by the sum over the log likelihoods of the correct word at each position, where log is the natural logarithm applied element-wise to the vector p_t:

$$L(\mathbf{h}, \mathrm{S}) = -\sum_{t=1}^{M} [\log p_t] \cdot \mathbf{s}_t. \tag{12}$$

4.6 Training

In all experiments, we use a stochastic gradient descent (SGD) solver with a base learning rate of $\eta_b = 2.0$. The base learning rate applies to the decoder LSTM network, while we use a reduced learning rate for other parts of our model. For the question encoder CNN, the image attention and the multimodal fusion, we use a learning rate of $\eta_{\text{Q-CNN}} = 0.005 \cdot \eta_b$, $\eta_{\text{att}} = 0.05 \cdot \eta_b$ and $\eta_{\text{fusion}} = 0.05 \cdot \eta_b$, respectively. At a varying number ϵ of epochs, we halve the learning rate. When finetuning the *Inception-v3* network in an end-to-end fashion we set all learning rates to $\eta = 0.0005$. We use a batch-size of $\beta = 128$ in most of our experiments and reduce the batch-size to $\beta = 32$ when finetuning the *Inception-v3* network. All experiments were conducted on a single NVIDIA Titan X Pascal GPU.

For training we use each input (image, question, answer) triple as one training example. Note, that we sample multiple training examples from one (image, question) pair with multiple possible answers to model the uncertainty between the annotators. Thus, we get a slightly richer training signal because the model learns that multiple answers could be correct.

Depending on the model, we halve the learning rate 2 to 4 times and train for $\epsilon = 2$ to $\epsilon = 4$ epochs per learning rate. When using a lower learning rate, we increase the number of epochs per learning rate to 6. In the finetuning stage, we train until convergence for a total of up to $\epsilon = 35$ epochs. On our GPU, training in the first stage with precomputed features takes 2 to 4 days. The finetuning stage takes 8 to 10 additional days.

4.7 Implementation

We implemented our model in Tensorflow [1] and used the slim module for a prebuilt *Inception-v3* DCNN. Our model shares code with the Show and Tell [25] model, which is publicly available as a Tensorflow implementation.

5 EVALUATION

We conducted a series of experiments to conclude an architecture producing best results for our scenario (see Table 1). In the following we list some of our ablation experiments in order to justify our hyperparameter selection. All models build upon an *Inception-v3* DCNN pretrained on the ILSVRC 2012 corpus. We precomputed the *Inception-v3* features of the input images, since every image is used multiple times and, thus, training is sped up. Except stated otherwise, we report accuracies on our validation split of the VQA-v2 dataset.

5.1 Question feature extraction

As we described in Section 4.1, we used a convolutional architecture to extract a single feature vector out of a given question. In addition to the post-activation layers we also conducted experiments with pre-activation layers (similar to ResNet-v2 [11]), which performed a little worse in our case. We also compared classification accuracies for different number of residual blocks (see column # blocks). Here we see that a larger number of residual blocks (4) does not improve performance in comparison to our choice of using only 2 blocks. Furthermore, in the first column of the feature extraction columns, we show our different choices for the kernel size k of the question convolutions. As a comparison, we also extract the question features with a simple LSTM network (one LSTM cell with 512 units), which decreases the validation accuracy about an absolute of 10 %. Note, that we used GLUs for every model except for model 6. Therefore, for the question feature extraction part of our model, we chose to use 2 post-activation residual blocks with a kernel size of $k = 5$.

5.2 Image attention

We then added a simple image attention model (see Section 4.3) to our model. As we still precompute the feature maps of the *Inception-v3* network, we cannot use the global average pooled feature maps anymore, but precompute the direct output of the *Mixed_7c* layer. In the table we mark models, that use attention with a checkmark in the second column. In the results, we notice that our model also profits by adding a simple form of attention. Attention improves the accuracy by about 1% and applying the non-linear layer f_q (see Section 4.4) to the question features ($\tilde{\mathbf{q}}$) also has a small positive effect on the validation accuracy. If we also optimize parameters of the *Inception-v3* DCNN, we get an additional improvement of about 3%.

5.3 Multimodal Factorized Bilinear pooling

In addition of fusing the question and image features simply by calculating the element-wise product (see Section 4.4), we also

Table 1: Studies on our test and validation splits of the VQA-v2 dataset. We analyze different combinations of hyperparamters in our model. The first column states the kind of model, Att stands for Attention, FT for finetuning of all parameters (including the *Inception-v3* image feature extraction DCNN), MM-Fuse describes the kind of multimodal fusion used and decoder specifies whether we used an LSTM or a fully-connected (FC) layer for generating the answers. The columns for validation performance give the accuracies on our validation split (10 % of the VQA-v2 validation split).

				Feature extraction Q				MM-Fuse	Decoder	Validation Performance			
Name	Att	FT	VG	k	# blocks	act	$f_q(\tilde{\mathbf{q}})$			All	Y/N	Num	Other
1				4	4	pre		eltwise	LSTM	52.02	71.69	34.82	41.38
2				4	4	post		eltwise	LSTM	52.42	71.57	35.98	42.05
3				4	1	pre		eltwise	LSTM	50.65	69.07	31.34	41.62
4				4	2	post		eltwise	LSTM	52.44	71.52	35.48	42.28
5		✓		4	2	post		eltwise	LSTM	54.67	72.63	37.04	45.55
6				-	LSTM	-		eltwise	LSTM	41.48	63.69	28.18	27.95
7	✓			5	2	post		eltwise	LSTM	53.54	73.33	34.81	43.34
8	✓			5	2	post	✓	eltwise	LSTM	53.71	73.62	35.12	43.38
9	✓		✓	5	2	post	✓	eltwise	LSTM	53.97	73.69	36.83	43.42
10	✓	✓		5	2	post	✓	eltwise	LSTM	56.62	75.45	39.33	46.79
11	✓			5	2	post	✓	eltwise	FC	49.16	72.82	33.41	35.26
12	✓			5	2	post		MFB	LSTM	51.94	72.44	33.54	41.13
13	✓		✓	5	2	post		MFB	LSTM	52.20	71.45	35.40	41.90

implemented the Multimodal Factorized Bilinear Pooling (MFB) approach by Yu et al. [28, 29], which is currently ranked first in the VQA-v2 open-ended challenge. They project the image features and question features into a larger embedding space, i.e., our embedding space is $\in \mathbb{R}^{5 \cdot 512}$. These larger embedding spaces are combined via an element-wise product and then reduced to $\mathbb{R}^{512}$ via sum-pooling, i.e., blocks of 5 values are summed up. This vector is then power normalized ($z \leftarrow sign(z)|z|^{0.5}$) and L_2 normalized ($z \leftarrow z/||z||_2$). Yu et al. propose that MFB supports the exploitation of more complex correlations between multimodal features. As we can see in the *MM-Fuse* column of Table 1, using the MFB approach (Models 12 and 13) does not increase overall accuracy in our case and we stick with using the simple fusing approach described in Section 4.4.

5.4 Classical VQA approach

As we use an LSTM network as the decoder instead of a fully connected layer (FC) with a softmax cross-entropy loss function, we still want to compare our architecture to the classical VQA classification approach. For this reason, we extended our model to be able to classify the 3000 most common answers of the train split. When we evaluate this approach on our test set, accuracies drastically decrease compared to the LSTM decoder (53.71 vs. 49.61). This may be an indicator that the classification approach performs worse than an LSTM decoder network.

5.5 Dataset extension

We also examine, whether the validation accuracy can be improved by extending the dataset with the Visual Genome (VG) dataset. Because we do not evaluate on the VG dataset, we include the whole dataset into our train split. Models trained with the extended dataset are indicated by the VG column. In contrast to models that use the classification approach, we use the whole VG instead of only using the part, which contains the same top answers as the VQA-v2 dataset. In both experiments, in which we extended our dataset by VG, we notice a slight improvement in the scores. This is due to the fact that the VG dataset contains longer answers than the VQA-v2 dataset, which mainly consists of one to two word answers and we measure the performance on the validation split of the VQA-v2 dataset. We depict the distribution of answer lengths in Figure 4.

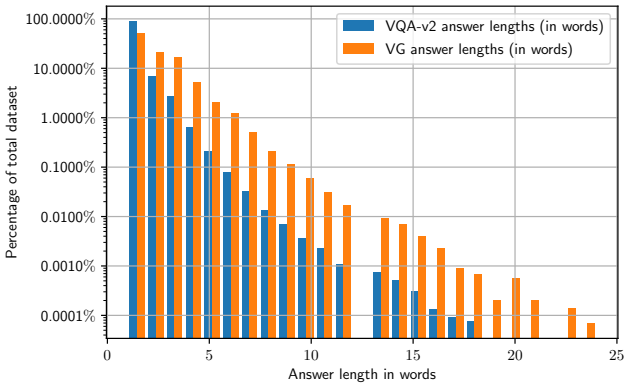

Figure 4: The VQA-v2 dataset contains many short answers (89.41% and 6.92% of all answers are of length 1 and 2 words, respectively), while the the VG dataset contains more answers with more words (52.47%, 21.86% and 16.70% of all answers are of length 1, 2 and 3 words, respectively).

Table 2: Fractions of unique answers generated by our models. LCA are less common answers left out altogether by classification models, MCA are most common answers, which are the only answers generateable by classification models and new answers are newly generated sentences by our models not contained in the train split. The LCA accuracy describes the percentage of correctly generated answers out of the LCA set, i.e., these are correct answers given by our model, which classification models are not able to produce.

Name	LCA	LCA accuracy	MCA	new answers
9	14.97 %	11.91 %	62.73 %	22.30%
13	14.70 %	9.47 %	63.62 %	21.68 %
10	19.49 %	8.10 %	69.74 %	10.77 %
7	24.52 %	9.91 %	64.22 %	11.26 %

5.6 Less common answers

We use an LSTM network to generate answers, thus, in contrast to classification models, we are able to generate answers which are not part of the 3000 most common answers. Our train set contains 190,677 unique answers and the 3000 most common answers (MCA) comprise 90.5% of all answers in the dataset. Thus, classification models with 3000 logits are unable to produce 9.5% of the less common answers (LCA) of the train set. In contrast, our LSTM model is able to produce every answer and even new answers. In an experiment, we determined the fraction of MCA, LCA and new answers of all uniquely generated answers of some of our models. Our models generate between 14.70% and 24.52% LCA and between 11.26% and 22.30% new answers. With the official evaluation tool, we also find that the accuracy of LCA is up to 11.91%. We visualize these findings in Table 2. In qualitative analysis we depict some of the new answers generated by our model in Figure 5. The first row shows correct answers not detected by the official evaluation script. The second row depicts incorrect answers, e.g. (e) and (f) show answers being too short (due to the short answer bias of the dataset) and (g) and (h) show wrong answers, even though the answers are plausible given the scene.

5.7 Performance on the test set

In Table 3, we compare our model to other models on the test-dev as well as on the test-std dataset split. We compare our model to the MCB [7] approach as reported in [9] and the single network model by Teney et al. [24]. We did not train an ensemble of networks and did not implement attention on image regions by a object detection model. Our models surpass the d-LSTM+n approach but are worse than the best scoring approaches, since we focus more on generating less common answers. Furthermore, our model can produce more complex answers due to the LSTM decoder network and could be seen as an approach for more complex VQA datasets to come.

6 CONCLUSION AND FUTURE WORK

We presented an end-to-end architecture for the VQA task. In contrast to other approaches, we use an LSTM network to generate the answers for the (image, question) pairs. We conducted a series of experiments to find a good combination of hyperparameters for our architecture. Competing approaches model the VQA task as a classification task, where a fixed number of most common answers are the possible classes, hence, they are not able to produce less common answers. With our approach, we were able to generate such less common answers with an accuracy of 11.91% and also showed that our model generates answers not contained in the training set. Some of these produced sentences answer questions correctly. We also find out that the ground-truth answers of current VQA datasets are biased towards a short number of words, i.e., more than 90% of the answers have a maximum length of two words. For datasets containing more complex answers, our model is more suitable than models using the classification approach. Hence, we want to focus our future research on finding a performance measure to evaluate longer and more complex answers and extending the datasets by more complex sentences.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. 265–283.

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*. 2425–2433.

[3] Kobus Barnard and David Forsyth. 2001. Learning the semantics of words and pictures. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, Vol. 2. IEEE, 408–415.

[4] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325* (2015).

[5] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083* (2016).

[6] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *European conference on computer vision*. Springer, 15–29.

[7] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847* (2016).

[8] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. *arXiv preprint arXiv:1705.03122* (2017).

[9] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. *arXiv preprint arXiv:1612.00837* (2016).

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European Conference on Computer Vision*. Springer, 630–645.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[13] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4565–4574.

[14] Andrej Karpathy. 2016. *CONNECTING IMAGES AND NATURAL LANGUAGE*. Ph.D. Dissertation. STANFORD UNIVERSITY.

[15] Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3128–3137.

[16] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision* 123, 1 (2017), 32–73.

[17] Siming Li, Girish Kulkarni, Tamara L Berg, Alexander C Berg, and Yejin Choi. 2011. Composing simple image descriptions using web-scale n-grams. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 220–228.

(a) Q: Where is the man?
A: in a restaurant

(b) Q: Where are the trees?
A: behind the elephants

(c) Q: Where is the bus?
A: on the street

(d) Q: Where is the red carpet?
A: on the ground

(e) Q: Who wears a cowboy hat?
A: the man on the

(f) Q: What is the woman doing?
A: feeding the

(g) Q: Where is the photo taken?
A: in a kitchen

(h) Q: Where is the boy looking?
A: at the beach

Figure 5: Images associated with question and generated answers by our model. All answers shown are new ones not contained in the training set. Figures (a) - (d) show correct answers not detected by the official evaluation script. The second row shows wrong answers. Especially, (e) and (f) show sentences, where the end of sentence token was generated to early (dataset bias of short answers). (g) and (h) show wrong answers.

Table 3: One of our models (model 10) compared to state-of-the-art models on the test-dev and test-std dataset splits.

	VQA-v2 test-dev				VQA-v2 test-std			
	All	Y/N	Num	Other	All	Y/N	Num	Other
MCB [7, 9]	-	-	-	-	62.27	78.82	38.28	53.36
d-LSTM+n [9, 19]	-	-	-	-	54.22	73.46	35.18	41.83
Teney et al. Single Network [24]	62.07	79.20	39.46	52.62	62.27	79.32	39.77	52.59
10 (Ours)	56.32	75.64	36.21	44.36	56.68	76.02	36.56	44.46

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*. Springer, 740–755.

[19] Jiasen Lu, Xiao Lin, Dhruv Batra, and Devi Parikh. 2015. Deeper LSTM and normalized CNN Visual Question Answering model. https://github.com/VT-vision-lab/VQA_LSTM_CNN. (2015).

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.

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

[23] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2818–2826.

[24] Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. 2017. Tips and Tricks for Visual Question Answering: Learnings from the 2017 Challenge. *arXiv preprint arXiv:1708.02711* (2017).

[25] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3156–3164.

[26] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2016. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE transactions on pattern analysis and machine intelligence* (2016).

[27] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* (2016).

[28] Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. 2017. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *Proc. IEEE Int. Conf. Comp. Vis*, Vol. 3.

[29] Zhou Yu, Jun Yu, Chenchao Xiang, Jianping Fan, and Dacheng Tao. 2017. Beyond Bilinear: Generalized Multi-modal Factorized High-order Pooling for Visual Question Answering. *arXiv preprint arXiv:1708.03619* (2017).

[30] Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2015. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167* (2015).