Pointer-CNN for Visual Question Answering

Jakob Svidt
Aalborg University
jsvidt13@student.aau.dk

Jens Søholm Jepsen
Aalborg University
jjepse12@student.aau

June 14, 2018

Abstract

Visual Question Answering(VQA) is an interesting problem from a research perspective, as it is an intersection of the Computer Vision and Natural Language Processing (NLP) domains. Many recent methods focus on improving features, attention mechanisms and hyper-parameter tuning. Most approaches model the problem with a fixed-sized classifier over the answers. We propose a Pointer-CNN classifier for multiple choice in VQA, which achives state of the art performance on both the VQA v1.0 and reasonable performance on the Visual7W data set. We provide an analysis and discussion of performance of the model on different question categories of VQA v1.0, to identify the shortcomings of our architecture.

1. Introduction

In recent years neural networks have been successfully applied to problems across a vast variety of domains, such as Natural Language Processing (NLP), Computer Vision(CV) and many other domains. In Natural Language Processing, models using variations of Recurrent Neural Networks (RNNs), attention mechanisms and pre-trained word-embeddings have achieved human performance on tasks such as Question Answering [1]. The aim of this task is to correctly answer questions given a Wikipedia article. In the field of Computer Vision, Object Classification gained a lot of momentum following the introduction of Convolutional Neural Networks, and the ImageNet Large Scale Visual Recognition Competition (ILSVRC) [2], [3]. Several models, such as ResNet, VGG and GoogleNet have since achieved super-human performance on the task [4], [5], [6]. Convolutional neural networks have likewise been used successfully on the task of Object Detection, with varieties of R-CNN models and lately YOLO [7], [8].

Visual Question Answering is an intersection of all of the domains mentioned above. The task is specified as follows: given an image, and a question posed in natural language, provide a natural language answer to the question with the image as context. Two tasks

can be specified within this problem: multiple choice, where the task is to choose between a number of question specific answer candidates. Open-ended, where the model has to produce an answer only given the image and question. Although the sub-domains of CV and NLP have achieved impressive performance, VQA seems to be a more difficult problem to solve. [9], [10]

There might be several reasons why this problem is difficult. Answering questions about images requires high-level knowledge about real-word concepts, and how they are represented visually and in language. Since much work has been done in both the image and natural language domain, one of the main challenges of VQA is finding common ground between these two modalities, to be able to reason about them in concert. State of the art methods tend to use features transferred from these two domains. In the image domain these features are usually from image classification and object detection models[11], and in the language domain they are usually pre-trained word-embeddings[12]. Since these features come from different sources, there is no connection between the visual and linguistic features representing the same concept, which means there is a gap between features transferred from different domains. The fact that the problem consists of three input sources of which two are text, namely question and image, means there is a predominance of features from the language domain. This makes it easy for models to rely more on the linguistic features from the question, and ignore the visual features from the image[9]. Using transferred features might introduce other issues. The object classification and detection domains only consider the objects in the image, although the image may contain information about the background or there might even be a need for external knowledge not appearing on the image. Our main contributions are:

1. A Pointer-CNN which achieves state of the art performance on the multiple choice task on VQA v1.0.

2. A way of modeling the problem, such that we can have a variable number of answer candidates instead of a fixed set.

The rest of the paper is structured as follows: *Related work*, where we review the field Visual Question Answering in terms of datasets and model and model components, showing that not much work has been done investigating the effects of different classifiers. *Problem Formulation*, where we briefly state the problem that we are trying to solve. In *Proposed Model*, we describe the different model components used to extract features from the input, combine them and produce an answer. *Experiments*, where we introduce the datasets, evaluation metrics and our results. The *Conclusion* section summarises our findings. In *Future work*, we discuss our ideas for further improvements, given our findings. In addition, we provide an optional *Appendix*, that briefly covers the main concepts needed to understand the work presented in this paper.

2. Related Work

Recently, the field of Visual Question Answering has gained a lot of traction. Especially following the introduction of the VQA challenges and datasets [13], [14], [15]. Other

(a) Q: *What sport is this?* A: *Tennis*

(b) Q: *Where is the cow standing?* A: *Field*

(c) Q: *Who flies the plane?* A: *Pilot*

(d) Q: *What time of day is it?* A: *Sunset*

Figure 1: The task of Visual Question Answering results in an answer, given an image and a question.

datasets have also contributed to the interest in VQA, such as: Visual7W [16], DAQUAR dataset [17], Visual Madlibs Q&A [18], COCO-QA [19] and Visual Genome [20].

Most models built for the tasks in these datasets focus on improving image features, language features, attention mechanisms, or finding a better architectures and hyper-parameter tuning [21], [22], [23], [10]. While these approaches have pushed the state of the art, not much work has been done to improve the part of the model that provides the answer. Jabri *et al.* argue that current approaches can be divided into generative and discriminative models[9].

Generative Models produce an answer by encoding information about the image and question, and passing that encoding to a decoder represented by an RNN. The RNN generates the answer word by word. While this approach seems attractive, since the solution space is only limited by the vocabulary. It has proven difficult to jointly learn suitable features along with a decoding model [24], [25].

Discriminative Models choose an answer by classification. In the multiple choice task the classes are the number of answers associated with a given question-image pair. Many current approaches can only classify over a fixed number of answer choices[15], requiring the number of answer candidates for all questions to be the same, therefore requiring changes to the model architecture to train it on datasets with a different number of answer choices. Jabri *et al.* [9] model the problem as a binary-classification problem over triples of image, question and answer for each answer candidate, which allows them to train the same model on datasets with a different number of classes, without any parameter or architectural changes. The simplicity of their model, however, reduces the model performance on the VQA v1.0 multiple choice task.

In open-ended, a common approach is to classify over the top-k most common answers or answers that appear more than n-times in the training data. The classifier is typically modeled by a feed forward neural network that outputs a probability distribution over the top-k answers. This limits the number of answers the model can produce to these k answers, effectively limiting the solution space. Teney *et al.* [10] found that choosing an n of 8, resulting in 3129 answer candidates, for the open-ended task, made their model perform the best.

Inspired by pointer networks [26], that are discriminative models where the set of output classes depend on the input, we try to combine the strengths of the different discriminative models described above.

3. Problem Formulation

The problem of visual question answering can be defined as: A model $M(I, Q, A_{I,Q})$ takes an image I as a tensor of dimensions ($width \times height \times channels$), a question Q as a sequence of word indices and a set of answer candidates $A_{I,Q}$ where each member c_n is a sequence of word indices and produces the probability of each c_n being the correct answer. In the multiple choice task $A_{I,Q}$ varies depending on the image and question. In the open-ended task this can simply be considered the set of all possible answers.

4. Proposed Model

In this section we propose the Pointer-CNN neural network model for the multiple choice VQA task. In the following sections W_{name} refers to a learnable matrix of weights, and b_{name} refers to a learnable bias vector and the subscript $name$ identifies in which component they occur. The constant H_{dim} refers to the common hidden size, which we use throughout the model. σ denotes the activation function(A.1.1), which is the Exponential Linear Unit (ELU), unless otherwise stated. ELU works as in equation 1.

$$\sigma_i(z_i) = \begin{cases} z_i, \text{ if } z_i > 0 \\ e^{z_i} - 1, \text{ otherwise} \end{cases} \tag{1}$$

Figure 2 shows an overview of our proposed model for the multiple choice task. In the following section we describe the components of that model.

4.1. Image embedding

Computer vision tasks are complex, and require large datasets to specify them, and lots of processing power to train. Since the amount of labelled images in visual question answering tasks are relatively small, compared to other computer vision tasks, we use image features transferred from other models trained on these larger datasets.

The input image is represented as a tensor of $W \times H \times C$. We follow the approach of Anderson, He, Buehler, Teney, Johnson, Gould, and Zhang [11] and embed the image using features extracted from a pre-trained R-CNN object detection model. This model is trained on RGB images of size 448 × 448. The R-CNN works in two stages, object localization and classification. One part of the model proposes a number of bounding boxes containing objects, with associated confidence scores. The other part of the model classifies the sub-images contained in these bounding boxes. The activations of the last layer of the classification model is used as the features for each of the regions. The top 36 object proposals are used, sorted by proposal confidence. This gives us a set of 36 image features of dimension 2048, where each vector represents the features of a detected

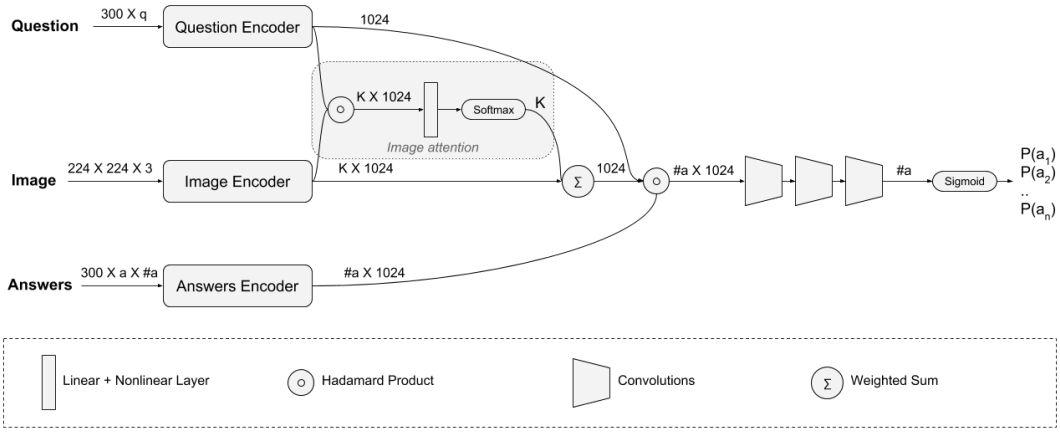

Figure 2: Overview of the proposed model for the multiple choice task. Where numbers above lines indicate the size of the features passed trough to the next operation. K is the number of regions from the CNNs. q is the length of the question. a is the length of the answer. $\#a$ is the number of answer candidates. P is the probability that a given a is the correct answer

object, just before classification. Each of these vectors are transformed through a single layer, to the common hidden size, as shown in equation 2. This is done in order to have a common feature size for both language and image features, so they can be fused later in the network.

$$\mathsf{r}_n = \sigma(\mathsf{i}_n \mathsf{W} + \mathsf{b}) \tag{2}$$

Where $\mathsf{r}_n \quad R$ is the image region embedding corresponding to the feature vector i_n, and the dimensions of W are $2048 \times H_{dim}$. This step gives us the set $R = \{\mathsf{r}_0..\mathsf{r}_{36}\}$ of region embeddings.

4.1.1. Object Classification

As an alternative to the image features from object detection, we also experiment with using features from variations of Resnet[27] (see A.2), trained on the object classification task. To use the features from a pre-trained CNN, we take the activations of the CNN, just before pooling and classification, resulting in a spatial feature map. Using an image of size $224 \times 224 \times 3$, we get 7×7 feature maps for the convolution operations in the CNN, which we use as image region embeddings. Each of the k feature maps are passed trough a neural network, mapping the embeddings to H_{dim}, represented as 1024 on figure 2.

4.2. Question and Answers Embedding

The language specified by the visual question answering datasets, is quite limited, in that many words only occur a single or a few times. This means that learning a language model from scratch, would result in a very crude model, that would have very limited knowledge of the semantic and contextual information of words. To circumvent this, we use word embeddings pre-trained on a large language corpora. These word embeddings supply the model with semantic information not directly present in the datasets, allowing the language model to generalize.

The question and answers are fed to the model as sequences of word indicies. The question is a single sequence $Q = [q_1, \ldots, q_n]$ and the answer candidates is a set of sequences $A_{Q,I} = \{[c_{1,1}, \ldots, c_{1,m}], \ldots, [c_{k,1}, \ldots, c_{k,m}]\}$, one for each answer candidate k. These are embedded by first converting each word index to its corresponding pre-trained word embedding. We use the publicly available embeddings from GloVe [12], providing us with a word vector of size 300. Each of word embeddings in the sentence are then fed through a Recurrent Neural Network(RNN)(see section A.4), to capture the contextual information of the sentence. We are using a Gated Recurrent Unit (GRU) RNN, as this solves the vanishing gradient problem associated with standard RNNs [28]. This gives us a sentence vector with the size of the last layer in the RNN. As we have multiple answer candidates, the answers encoder returns a feature tensor with the number of answer candidates $|A_{I,Q}| \times H_{dim}$. The RNNs for both question and answers, have the same architecture and shared weights, to allow the model to learn a common language model for both text types. Two individual layers, one for questions and one for answers, are used to transform the last hidden state of the RNN to H_{dim}.

We also experiment with using a simple BoW embedding of sentences, where the RNN above is replaced by a simple sum over the word embeddings to produce a sentence embedding, before it is passed through the transformation to H_{dim}. This step produces a question embeddign q and a set of answer embeddings $\{\mathsf{a}_0 \ldots \mathsf{a}_n\}$.

4.3. Multi-modal fusion

To allow the model to jointly reason over both question, answer and image, we fuse the information contained in the three embeddings. To fuse the text and image embeddings, we experiment with two methods, the hadamard product and concatenation. The choice of fusion determines how the model is constrained, when learning the embeddings of the different modalities. Concatenating the embeddings of the modalities places no constraint on the learned embedding, whereas using the hadamard product forces the model to learn a common embedding space for the modalities, because of the multiplicative interactions between the embeddings. We find that hadamard fusion is superior to concatenation, as is reported in the literature [10].

We define a function $fuse(\mathsf{x}, \mathsf{y})$ that fuses two embeddings by hadamard product ( ):

$$fuse(\mathsf{x}, \mathsf{y}) = \mathsf{x} \quad \mathsf{y} \tag{3}$$

where x and y are the two embeddings.

4.4. Image attention

By using an attention module the model can focus on the relevant regions of the image, given a particular question. This module first fuses each of the image region embeddings with the question, and then transforms these fused embeddings to a number of scalar values, representing the relevance of each of the image regions given the question embedding. The unnormalized region weights, $\mathbf{u}$ are calculated as in equation 4.

$$u_n = fuse(\mathbf{r}_n, \mathbf{q})\mathbf{W}_{att} + \mathbf{b}_{att} \tag{4}$$

The scalar values are normalized by passing them through a softmax function in equation 5.

$$w_j = \frac{e^{u_j}}{\sum_{k=1}^{K} e^{u_k}} \quad for\ j = 1..k \tag{5}$$

Each of the image region embeddings are scaled by their corresponding scalar weights, and then summed, to obtain a fixed size vector of the common hidden size in equation 6.

$$\mathbf{i} = \sum_{\mathbf{r}_n \in R, w_k \in \mathbf{w}} \mathbf{r}_n \cdot w_n \tag{6}$$

The vector $\mathbf{i}$ is the weighted sum over the image region embeddings of the whole image, and referred to as the image embedding. We also experiment with implicit attention[29], that simply averages the region embeddings after fusing them with the question embedding, as in 7.

$$\mathbf{i} = \frac{\sum_{\mathbf{r}_n \in R} fuse(\mathbf{r}_n, \mathbf{q})}{|R|} \tag{7}$$

4.5. Answer Fusion

Each of the multiple choice answers are fused the answer embeddings with both the question and image embedding, to obtain a set of multi-modal answer embeddings, one for each answer, that contain information about all of the three input types, before they are passed along to the classifier.

$$\mathbf{f}_n = fuse(fuse(\mathbf{i}, \mathbf{q}), \mathbf{a}_n) \tag{8}$$

4.6. Classification

We approach the problem as either a multi-class or multi-label classification problem, where the classes are determined by the multiple choices provided with the question. Considering the problem as a multi-class classification task, means that we assume a question can have only one answer. This assumption does not take into account language ambiguities and questions with subjective answers, as are present in the VQA dataset. For instance, the question *"is the tea hot?"* can be difficult to determine only based on

an image. If we model the problem as a multi-label classification task, we allow for the possibility of a question having multiple correct answers, thus alleviating the problems of the multi-class formulation, as Teney *et al.* [10]. This is closely related to the approach used by Jabri *et al.* [9] who formulate the problem as a binary classification over a tuple of question, image and a single answer candidate, where we consider all answer candidates at the same time. When modelling the problem as a multi-class classification, the outputs of the network are normalized by a softmax, and in the case of multi-label the outputs are normalized by a sigmoid. The multi-class classification output of the model is defined in equation 9.

$$softmax(\mathbf{x}) = P(A_{Q,I}|Q, I) \tag{9}$$

where $A_{Q,I}$ is the set of answer choices for image Q, I, and $\mathbf{x}$ is the log-probabilities given by the model. The multi-label classification problem is defined as in equation 10.

$$sigmoid(x_n) = P(c_n = g|Q, I) \tag{10}$$

where g is the ground truth answer, $P(c_n = g|Q, I)$ is the probability of answer candidate c_n $A_{Q,I}$ being the correct answer and $\mathbf{x}$ is the log-probabilities given by the model. To perform hard classification given the two types of output above, we simply select the answer candidate with the highest probability.

4.6.1. Pointer-CNN

We propose a Pointer-CNN (P-CNN), inspired by the work of Vinyals, Fortunato, and Jaitly [26] as well as Wang and Jiang [30]. The purpose of the P-CNN is to enable the model to output a probability distribution over answers that is dependent on the input.

This allows us to combine the positive effects of having the answers as inputs, as observed by Jabri *et al.* [9], and still model the problem directly as multi-class or multi-label, instead of as a binary classification problem or classification over a fixed set of answer candidates as in many of the open-ended solutions [10], [23], [22]. We show that this formulation of the problem, allows the model to achieve higher accuracy on the multiple choice task. A central part of the P-CNN architecture is treating the answers as inputs, which allows the model to utilize semantic answer information from the pre-trained word-embeddings. The input to the P-CNN is the sequence of multi-modal answer embeddings $[\mathbf{f}_1, \ldots, \mathbf{f}_k]$. These are passed through three layers of 1-dimensional convolutional layers with H_{dim} filters size of $k = 3$, where the last layer transforms the hidden state to a single scalar for each of the multi-modal answer embeddings, corresponding to the unnormalized log-probability of the answer being correct. Using convolutional layers instead of fully connected layers, makes the model invariant to the ordering of the set of answers and means that the model only needs to learn a mapping from k H_{dim} to H_{dim}, instead of having to learn a mapping from $|A_{I,Q}|$ H_{dim} to H_{dim}. Using convolutions also allows us to train and evaluate the model on questions with a varying number of answer candidates as opposed to most current methods that require a fixed number of answer candidates.

The output $\mathbf{o}_l$ of all but the last convolutional layers l of the P-CNN is defined in 11 and the inputs $[\mathbf{x}_1 \dots \mathbf{x}_n]$ are the multimodal answer embeddings $[\mathbf{f}_1 \dots \mathbf{f}_n]$ for the first layer, and the outputs of the previous layer $[\mathbf{o}_{l-1,1} \dots \mathbf{o}_{l-1,n}]$ for the subsequent layers.

$$\mathbf{o}_{l,n} = \sigma(\text{concat}(\mathbf{x}_{n-\ k}, \dots, \mathbf{x}_{n+\ k})\mathbf{W}_l + \mathbf{b}_l) \text{ for } \mathbf{x}_1 \dots \mathbf{x}_n \tag{11}$$

Where the learnable weights $\mathbf{W}_l$ are of dimensions $(H_{dim} \quad 3) \times H_{dim}$. The output $\mathbf{o}_L$ of the last convolutional layer L is defined as follows:

$$\mathbf{o}_{L,n} = \text{concat}(\mathbf{x}_{n-\ k}, \dots, \mathbf{x}_{n+\ k})\mathbf{W}_L + \mathbf{b}_L \text{ for } \mathbf{x}_1 \dots \mathbf{x}_n \tag{12}$$

Where the dimensions of $\mathbf{W}_L$ are $(H_{dim} \quad 3) \times 1$ to transform the embedding to a single output logit for each answer candidate. If we model the problem as multi-class, the log-probabilities are normalized by a softmax, producing a multinomial distribution over answer choices. In the case of multi-label, we normalize each log-probability using a sigmoid, obtaining a set of binomial distributions, one for each answer candidate.

4.7. Training

To train the model we minimize an objective function $L(B)$ using the Adam optimizer[31], with default parameters: learning rate $= 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\quad = 1e^{-8}$. Adam is a variant of Stochastic Gradient Descent(SGD) and B is a batch in the form of a set containing tuples of $(\mathbf{y}, \hat{\mathbf{y}})$ where $\mathbf{y}$ is the ground truth, and $\hat{\mathbf{y}}$ is the model prediction. The size of the batch can vary, depending on the experiment. Teney, Anderson, He, and Hengel [10] find that using large batch sizes of 512 improves performance on their models. We use a batch size of 384, unless otherwise specified under experiments, due to memory limitations of the GPU.

Depending on the way we model the problem we use one of two different objective functions. When modelling the problem as a multi-class classification problem, where an example belongs to one, and only one, class, we use categorical cross entropy loss, defined as follows:

$$L_{CCE}(B) = - \sum_{\mathbf{y},\hat{\mathbf{y}} \ B} \sum_{i=1}^{|\mathbf{y}|} y_i \quad log(\hat{y}_i) \tag{13}$$

Where $\mathbf{y}$ is the ground truth in the form of a one-hot vector, and $\hat{\mathbf{y}}$ is the model prediction as a probability distribution. When modelling the problem as a multi-label classification problem, we use the binary cross entropy loss:

$$L_{BCE}(B) = - \sum_{\mathbf{y},\hat{\mathbf{y}} \ B} \sum_{i=1}^{|\mathbf{y}|} y_i \quad log(\hat{y}_i) + (1 - y_i) \quad (1 - log(\hat{y}_i)) \tag{14}$$

Where y_i is the ground truth probability of the example having the i-th label, and $\hat{y}_i$ is the model prediction. Note that in this formulation an example can have zero or more labels.

The model is regularized using Dropout[32] between layers, to reduce overfitting. Dropout is used between each layer of the network. We generally use use a keep

probability of $p = 0.5$, following the approach of [33]. Except for after the layers just before the answer fusion, where we use a keep probability of 0.8, since the multiplicative interaction of the fusion results in an effective dropout of p^3 for the fused features, and $0.8^3 \approx 0.5$.

We perform model averaging, by maintaining a moving average of the model parameters, that is updated after every gradient descent step, as follows: $\theta_{moving} = \theta_{moving} \quad \lambda + \theta \quad (1 - \lambda)$ where λ is the moving average factor of 0.999, θ are the model parameters after the gradient update, and θ_{moving} are the moving average parameters. The moving average parameters are only used during evaluation.

4.8. Open-Ended

In this section we briefly describe how our proposed model can be adapted to be used on the open-ended VQA task. We describe where this model differs from the one proposed for the multiple choice task.

Input For this model the input is still an image and a question as above, but no answer candidates are provided. Instead we use the top-K most common answers in the training data as a fixed set of answer candidates $A_{Q,I}$ for all question image pairs. Though we set this fixed K when training, we can add additional answer candidates when evaluating the model.

Answer embedding Embedding all the top-K answer candidates using an RNN as above, would be costly and time-consuming. Instead we create a fixed BoW embedding of each answer candidate, that consists of the average over the word-embeddings that constitute the answer.

Answer fusion Fusing all the top-K answers with the image and question, proved to be too memory consuming to be feasible. Instead we use an approach inspired by Ha, Dai, and Le [34], where the model learns to produce the weights of a transformation matrix T_n of dimensions $H_{dim} \times m$, given an answer candidate. The fused embedding of the question and answer, is transformed by the matrix produced for each answer candidate. This results in a feature vector of size m for each answer candidate, that is the fused multi-modal answer embedding.

Training The training procedure is the same as above using the binary cross entropy loss $L_{BCE}(B)$, except that the ground truth answers are soft-scores, as used by Teney, Anderson, He, and Hengel [10].

4.9. Implementation

For reproduction purposes we state the graphics card and ML framework used in this project as well as the pre-processing done on the inputs of the model. We have implemented our models using PyTorch, and performed experiments using a GTX 1080ti. It takes between 25-30 epocs for the models to converge, taking around 10-12 hours.

4.9.1. Image pre-processing

When using R-CNN object detection features, the images are simply mapped to their corresponding R-CNN features, as described in 4.1. In the configurations where we use pre-trained convolutional neural networks, the RGB images are scaled to 224×244 pixels. They are then normalized to have the same mean and standard deviation as the data the networks were trained on. This results in a mean of $0.485, 0.456, 0.406$, and standard deviation of $0.229, 0.224, 0.225$, for the three colour channels respectively.

4.9.2. Text pre-processing

All text, both questions and answers, is first tokenized by lower casing all characters, replacing all punctuation with spaces and replacing all numbers with their corresponding numerals. The text is then split by spaces into tokens. We construct a vocabulary that is a mapping from each unique token, to a unique integer id. Before being fed to the model, questions and answers are tokenized, and their tokens are mapped to their vocabulary ids.

5. Experiments

In this section we evaluate the model proposed above on three datasets and two tasks, compare our results with state of the art, and discuss our results. We conduct an ablation study to motivate the final configuration of the proposed model.

5.1. Datasets

We evaluate our model on two different tasks within VQA, the multiple choice and open ended task. In the multiple choice task the datasets provide an image, a question, a number of answer candidates for each question and the ground truth. In the open ended task the dataset only provides an image, a question and a ground truth answer. We evaluate our model on 3 different datasets. Information about the datasets can be found in table 1.

	Number of Images	Number of Questions	Avg. Q per image	Average Q length	Average A length	Multiple Choice	Open ended
VQA v2.0	204,721	1.105,904	-	-	-	no	yes
VQA v1.0	204,721	614,163	3.0	6.2	1.1	yes	yes
V7W	47,300	327,939	6.93	6.9	2.0	yes	no

Table 1: VQA datasets

Visual7W(V7W) is the smaller of the datasets. It provides 4 answer choices per question, where the incorrect answers are generated by humans.

VQA v1.0 provides 18 answer choices per question, of which only 1 is correct. The incorrect answers were generated automatically by random sampling from a fixed set of answers. This dataset has been shown to contain a bias in the questions, meaning that

models quite easily gain performance, relying only on the statistics of the questions and the corresponding answers.[9] For instance, for questions starting with *"do you see a.."*, the answer *"yes"* is the correct for 87% of the questions. Likewise *"tennis"* is the correct answer 41% of the time for questions starting with *"what sport is.."*.

VQA v2.0 tries to combat this bias, by adding more questions, and having a more balanced set of answers.

All three datasets are split into three parts, training, validation and test sets. In the case of V7W all data for all 3 splits are available to the public. For VQA v1.0 and v2.0 only the answers for training and validation are publicly available, and the full test-set is refered to as test-standard, with a subset of it refered to test-dev. To validate the model on the test-standard and test-dev set, model predictions are submitted for evaluation on a website provided by the authors of the dataset. This website maintains a leaderboard of all public submissions. Submissions are limited to 10 times a day and 9999 times in total on test-dev set and to 1 per day and 5 in total on the test-standard set.

5.2. Methodology

The authors of the VQA v1.0 and VQA v2.0 datasets introduce a soft-score evaluation metric, to account for subjectivity in answers, as each question has 10 ground truth answers from 10 individual annotators:

$$accuracy(ans) = min(\frac{\#\text{ humans that said } ans}{3}, 1) \tag{15}$$

The metric in 15, gives the maximum score of 1, if at least three annotators agree with the model predictions, and otherwise a fraction of the number of annotators that agree with the model divided 3. The performance on the Visual7W dataset is simply the percentage of correct answers. Before evaluating the answer a number of processing steps is done[15]:

- Making all characters lowercase
- Removing periods except if it occurs as decimal
- Converting number words to digits
- Removing articles (a, an, the)
- Adding apostrophe if a contraction is missing it
- Replacing all punctuation (except apostrophe and colon) with a space character.

The results reported on the test-dev and test-standard set on VQA v1.0 and VQA v2.0, and the test-set on V7W, are obtained by first training the model for 55 epochs on the training set, while evaluating its performance on the validation set every epoch. The epoch that produces the best results on the validation set is recorded. Then the model is trained from scratch on both the training and validation set, for the number

of same number of epochs that produced the best results on the validation set. This is avoid unintentionally overfitting on the test-set. This is the approach followed by Teney, Anderson, He, and Hengel [10].

5.3. Results

We report results for the multiple choice task on V7W and VQA v1.0, along with the open-ended task on the VQA v2.0 dataset. We compare our results with the state of the art results found in VQA papers, along with the the best models on the VQA leaderboards. Since some leaderboard results are posted recently, not all of them have accompanying papers describing the underlying approach. Thus we are not able to compare the technical differences between those models and ours. Since we are only allowed 5 submissions in total on the test-standard set, we mostly report on the test-dev set. Models with a "*" indicate that they are evaluated on test-dev.

Although the main focus of our work has been on the multiple choice task, we also report preliminary results on the open-ended task.

5.3.1. Ablation study

To motivate our final architecture, and to show the effect of different design choices, we perform an ablation study. To be able to run more experiments, we have only systematically tested the different model configurations on the VQA v1.0 multiple choice task. Table 2 shows the results of different configurations of our models. The reference model is the configuration with the highest accuracy. The other models are identical to the reference model, with the exception of a single component or hyper-parameter change to measure the effect of this component. A description of the different configurations and their results can be found below:

Reference model Figure 2 shows the architecture of the reference model. Questions and answers are embedded using pre-trained GloVe embeddings that are fed through an RNN. Image features are from a pre-trained object detection R-CNN, and attended over by the hadamard attention module. The question, image and answer embeddings are fused by hadamard product. Finally a P-CNN of three layers with filter size 3 produces the un-normalized log-probabilities of the answer candidates, that are normalized by a softmax, for multi-class classification. This configuration uses model averaging.

Bow In this configuration the RNN is replaced with a simple sum over the pre-trained word embeddings, as discussed in section 4.2. The BoW language encoder performs worse than the RNN. This is expected as the RNN is able to capture the sequential information in the questions, as well as choose and weight what information is important in the sentence.

Imagenet This model uses image features produced by a Resnet101[27] pre-trained on the Imagenet[2] dataset. We find that using these features reduces model accuracy,

VQA v1.0					
Models	All	Yes/No	Numb	Other	Test-dev
Language					
BoW	69.52	80.21	44.74	67.75	-
Image					
Imagenet	67.62	80.60	43.45	63.97	-
Attention					
Implicit	68.02	81.47	44.85	63.76	69.82
Concatenation	68.44	81.36	45.26	64.60	70.18
Classifier					
Sigmoid	**71.38**	**82.71**	45.80	**69.32**	73.00
MLP	71.16	81.96	46.70	69.21	72.82
Other					
No-ans-interaction	69.33	80.19	44.20	67.52	**73.44**
No-model-avreage	70.34	81.47	45.80	68.18	71.82
ReLU	69.52	79.58	45.27	68.08	69.82
Reference Model	**71.28**	**82.57**	**47.26**	68.87	73.09

Table 2: Experiments with different components and parameters on multiple choice for VQA v1.0

consistent with the findings of Anderson, He, Buehler, Teney, Johnson, Gould, and Zhang [11].

Implicit Using implicit attention instead of explicit. As discussed in section 4.7.

Concatenation Concatenation to fuse image and question embeddings before passing them to the attention layer significantly reduces accuracy. [10].

Sigmoid Modelling the problem as multi-label classification, by normalizing the model outputs with sigmoids, produces results comparable with those of the reference model.

Multilayer Perception(MLP) Replacing the P-CNN (section 4.6.1) with an MLP increases the number of model parameters, and reduces the accuracy by a few percentage points.

No-ans-interaction Using a filter size of 1 in the P-CNN, removes the interaction between the answer candidates (see 4.6.1), and results in lower accuracy on the validation set. Surprisingly, though, it increases overall accuracy on the test-dev set. This indicates that the effects of answer interaction needs to be investigated further.

No-model-average Not applying model averaging. As discussed in section 4.7 model averaging could be interpreted as a cheap ensemble model, thus performing worse than a model with this component.

RelU Using a ReLU activation function instead of an ELU, lowers accuracy on both the validation and test-set. This might be due to the hard threshold of ReLU, negatively impacting the hadamard fusion. When the value passed through a ReLU is negative, its activation is 0, meaning that no gradient can flow through it. When fusing with the hadamard, this results in embedding elements of $e_i = 0$ in the fused embedding whenever just one of the corresponding elements in the question, answer or image embeddings are zero.

Looking at the results, we see that all of the different components each contribute to the overall performance. The attention mechanism and activation function seem to be very important. We observe that the models generally obtain higher accuracy on the test-dev set compared to the validation set. We suppose this is due to a different distribution of questions and answers in the test-dev set.

5.3.2. Multiple Choice

We report results on Visual7W and VQA v1.0 for the multiple choice task. In table 3 we compare our models with state of the art models.

	VQA v1.0				Visual7W						
	All	Yes/No	Numb.	Other	All	What	Where	When	Who	Why	How
Methods											
MLP [9]	-	-	-	-	67.1	64.5	75.9	82.1	72.9	68.0	56.4
MLP - Pre-trained VQA [9]	-	-	-	-	**68.5**	**66.4**	**77.1**	**83.2**	**73.9**	**70.7**	56.7
MCB [35]	70.1	-	-	-	62.2	60.3	70.4	79.5	69.2	58.2	51.1
Hierarchical Co-attention[23]	66.1	-	-	-	-	-	-	-	-	-	-
HDU-USYD-UNCC	75.35	**87.70**	48.66	70.59	-	-	-	-	-	-	-
Our Models											
Pointer-CNN	74.55	84.26	47.76	72.04	66.5	64.2	73.5	80.9	72.9	66.1	**57.7**
Ensemble Model*	**76.62**	85.89	**51.87**	**74.26**	-	-	-	-	-	-	-

Table 3: Multiple choice results on VQA v1.0 and V7W. Results are reported on the test-standard for VQA v1.0 and the test set for Visual7W. The "-" character indicates that the accuracy for the given category was not available. Models with a "*" indicate that they are evaluated on test-dev. Models that are not cited directly in the table, are taken from the official leaderboard of the challenge, at the time of submission, thus papers are not available.

Ensemble networks have been shown to be a powerful method to improve accuracy of machine learning models, and is commonly used in VQA [10]. We use a simple ensemble model, where the configuration is the same for all constituent networks, but with random initializations. Due to the stochastic nature of the training producedure, this makes the constituent networks converge to different local minima, and thus produce slightly different results. The output of the ensemble is produced by summing all the output logits of each constituent network, and selecting the answer with the highest log-probability. We experimented with the effects of ensembling before we found the best model configuration. Since training ensemble models are very time, we have only made one for the *Sigmoid*

model, however the positive effects of ensembling can be seen by the performance increase from 73.00 to 76.62, when using an ensemble model of 9.

Comparing the results on the Visual7w dataset we see that, the state of the art model have been pretrained on VQA, thus the most comparable model is MLP only trained on Visual7W. Our results are not quit as good at that model. We suspect that this is due to the fact that we did not have the R-CNN features available for this dataset, thus we would expect a similar increase in performance as on the VQA v1.0 dataset.

5.3.3. Open-ended

In table 4 we see two baselines listed. These baselines are provided by the VQA team [14] to put the accuracy of the models into perspective. *Prior*, is the accuracy obtained by always answering the most common answer in the dataset, which is "*yes*". Language only, is an LSTM-based model, where the model is not given the image as input. Furthermore table 4 shows a list of methods, which are a combination of methods found the literature as well as the top performing models on the VQA leaderboard. As the latest VQA challenge is fairly new not many papers have been released stating accuracy. Thus, we use reference model from Goyal, Khot, Summers-Stay, Batra, and Parikh [14] as well as the winners of the last challenge [10].

	VQA v2.0			
	All	Yes/No	Numb	Other
Baselines				
Prior [14]	25.98	61.20	00.36	01.17
Language only [14]	44.26	67.01	31.55	27.37
Methods				
Tips and tricks [10]	70.34	86.60	48.64	61.15
Tips and tricks Single Model* [10]	65.38	81.82	44.21	56.05
MCB [14], [36]	62.27	78.82	38.28	53.36
d-LSTM+n-I [14]	54.22	73.46	35.18	41.83
Tohoku CV Lab	71.12	87.29	**53.25**	61.13
casia_iva	71.31	86.98	51.05	62.31
HDU-USYD-UNCC	**72.09**	**87.61**	51.92	**63.19**
Our Model				
Pointer-CNN*	65.98	82.47	45.69	56.52

Table 4: Open-ended Results on VQA v2.0 test-standard. Models that are not cited directly in the table, are taken from the official leaderboard of the challenge, at the time of submission, thus papers are not available. Models with marked with "*" are evaluated on the test-dev set instead of test-standard.

Looking at the different answer categories, it is noticeable, that every model performs the worst in the *number* category. This could imply that, even though features from object detection models have been used, the models still find it difficult to reason about

the connection between the *k*-regions provided. Even though the number of possible answer candidates is limited to a number, usually $1 - 10$. *Yes/No* answers perform the best among the categories. This is expected as the possible answer candidates of this answer type is limited to either "*yes*" or "*no*". Likewise, this category has an advantage, because many answers are related to this category as "*yes*" and "*no*" are the most common answers.

We have included the Single model from Tips and Tricks paper [10] also evaluated on test-dev, to get a sense of comparison. Based on the results of the single model, we assume that we would get reasonable performance, if we were to make an ensemble model as well. In the future work we will discuss ideas of how to make our open-ended model perform better.

We have experimented with expanding the number of answer candidates at evaluation time, going from 3129 to either 4000 or 10000 candidates. This does not seem to impact the results of the model, in either a positive or negative way. We presume this is due to a couple elements of our model. The model used for this experiment is the reference model, since the P-CNN has a filter size of 3, and we do not shuffle the answer categories, the CNN always see the same surrounding answers for each given answer candidate. This might result in that the model is making some false assumptions. If we were to shuffle the data, this would make the problem more closely related to the multiple choice task. Moreover, the additional answer candidates added appear less times in the data, and might not have word embeddings available, thus making it near impossible to generalize over all answers. This will be discussed more extensively in the section future work. Though the results of this does not seem impressive, we believe that in order to push the VQA domain further, models need to produce an answer based on a larger language corpora.

5.4. Discussion

Each question in the VQA datasets is assigned to exactly one of 65 question categories. These categories can help us understand where the different models perform well and where they struggle. In table 5 we see a comparison between 4 different models and their accuracy in the different question categories.

Question Category	ImageNet	NoAnsInt	Reference	# Ques
are there	71.63	75.68	74.68	5877
what brand	56.01	54.96	58.94	1600
what room is	88.52	93.88	93.86	1647
what color is	65.67	83.05	84.67	2649
is	80.03	78.41	82.36	6079
are they	79.30	79.47	82.41	3074
what number is	39.47	34.11	35.55	1668
what sport is	95.47	96.59	96.68	2527
are	77.37	75.49	77.54	4912
is the	78.03	78.11	80.33	34927

what is the person	79.92	79.88	82.88	1729
how many	43.13	44.81	47.57	42339
does this	80.60	79.93	81.31	4396
is there a	89.34	89.42	89.85	9982
is he	79.79	82.38	84.77	2534
what	59.48	60.42	62.10	34608
does the	79.84	77.73	81.03	6103
is the person	78.51	75.94	78.60	1694
where is the	54.58	54.96	55.09	6734
what animal is	82.58	82.07	83.53	2001
how	40.46	40.09	42.63	4740
what is the woman	75.37	79.74	78.37	1706
none of the above	66.76	65.09	67.91	16973
who is	53.00	53.06	53.92	2154
is the woman	77.34	77.64	79.15	1938
are the	77.99	77.52	79.12	10701
how many people are	45.33	49.60	49.80	4276
what is on the	60.23	64.36	65.30	4254
has	76.79	76.02	79.73	1827
was	86.39	81.84	88.46	1551
what type of	67.95	68.81	70.46	7962
is this an	81.69	81.59	82.52	1981
do	74.38	76.07	77.90	3012
what is the man	79.71	81.41	82.15	5238
which	54.87	53.32	56.14	5382
are these	80.75	79.54	82.38	5782
what are	75.68	80.13	81.12	3277
what is the	64.16	66.13	66.87	24502
where are the	57.46	53.73	56.26	2161
is this a	82.89	81.05	84.12	16024
can you	77.07	74.12	78.29	1728
what time	67.85	65.58	69.09	2914
what are the	67.44	69.64	70.96	7225
are there any	72.96	78.64	80.80	2790
what color are the	62.09	76.56	78.78	6183
why	34.98	36.09	35.06	3347
what is this	80.58	81.54	82.36	3970
how many people are in	46.29	48.55	53.28	2071
do you	82.57	80.52	84.62	1971
is this	80.62	81.00	82.87	16444
why is the	38.07	36.11	40.14	1544
what is the color of the	74.05	79.89	85.31	1750
what is	62.97	65.18	66.20	13561

could	89.64	89.08	90.47	1698
is that a	81.13	79.28	83.05	1585
what is in the	66.91	70.82	71.61	3990
what does the	51.42	47.21	51.75	4075
what kind of	66.44	68.57	69.48	11192
is it	86.44	87.67	89.52	7345
is the man	79.10	77.92	81.13	4972
what is the name	44.57	37.64	47.67	1618
is there	84.65	84.86	86.41	6513
what color is the	65.06	77.18	78.99	27962
what color	51.11	71.86	71.67	3032
is this person	76.34	77.48	80.05	1756

Table 5: Accuracy over question types for 4 different models for 3 models from the ablations

Comparing the results of NoAnsIt (no-answer-interaction) and the reference model we see a small increase in accuracy across almost every category. Surprisingly there is a big difference between the results in "what is the name" with 10.03%. Apparently the the answer interaction helps the model get a better sense of what is written.

Looking at the results of the model trained on Imagenet, we observe that categories that require reasoning over multiple objects or where the question could reference any object located in the image, this model seems to get worse results. This evident on question categories such as: "what color are the", "what is the color of the", "what room is" and "what is on the". For question types closely related to the object classification task, such as "what animal is" and "what sport is", this model performs similarly to the reference model.

We see a big differences in accuracy between the question types. Questions that are similar to the classification problem are easily answered. These include "what sport is", "is there a", "could", "is it" and "what room is". More complex question types, such as "why", "how" and "how many", have considerably lower accuracy. This could be due to the fact that the image features used, both from object detection and classification, are used for classification purposes in their original domains, and are therefore not required to contain knowledge of concepts that do not pertain to the classification task.

Another reason for the models' inability to answer many of the more complex questions, might be that the knowledge required to answer these types of questions is not present in the datasets, even when transferring knowledge implicitly through pre-trained image and word embeddings. Especially in the case of "why" questions, access to additional knowledge might allow the model to reason about more complex concepts.

Categories such as "what number is" and "what is the name", would require that the model be able to read text and digits in the images, which would most likely require much more examples, or additional features transferred from a model trained to solve this task. The highest scoring model gets 35.55 and 47.67 in these two categories respectively.

5.4.1. Transferability

We conduct experiments where we evaluate the model trained on the Visual7W dataset on the VQA v1.0 and vice versa. If the model is able to generalize over both language and images it should achieve decent performance when transferred to another dataset.

When evaluating a model on VQA v1.0 trained on Visual7W, we see that the model struggles in several categories. It achieves an overall accuracy of 21.54. Though this seems low we observe that the model gets 88.93 in "what sport is", 61.64 in "what is the person" and 58.97 in "what is the man". We suspect this performance is due to, that these questions are similar to questions available in the Visual7W dataset. Most models get the highest score on "Yes/No" answers on this data set, however the transferred model only gets 13.47. As all the questions in the Visual7W dataset starts with either "What, Where, When, Who, Why or How" the model has little knowledge about a simple "Yes/No" questions. Furthermore, as the amount of data in Visual7W is fairly low compared to VQA(Table 1), it would be impressive if the model were able to be fully or partly transferable. Evaluating a model trained on VQA, on Visual7W achieves 51.51. Though this is more than the 21.54, it sill do not perform as well as models only trained only on the Visual7W data. This indicate that the models not able to fully generalize over the problem, but instead finds dataset specific patterns.

6. Conclusion

We have proposed a novel discriminative classification model for Visual Question Answering, that can be trained on different datasets with the same hyper-parameters and architecture, and achieve state of the art performance on the multiple choice task on the VQA v1.0, and reasonable performance on the Visual7W dataset, while only being trained on the data available in the dataset. We discussed the different elements of the model, where it falls short and where it stands out, as well as in what domains we believe it can be improved in future work.

We do not claim to have made any breakthrough advancements in the VQA area, but we hope that this work will be useful in future work on investigating the use of different classification mechanisms, that ultimately will be able to generalize better as well as not being limited by a fixed number of answer candidates.

7. Future Work

As previously discussed, the models seem to struggle with questions regarding text and digit recognition. Since non of the pretrained image or language features are with in this domain, this is to be expected. We believe that introducing transferred knowledge from this domain could increase the performance of the model in these domains.

When using GloVe, some words do not appear in the word embeddings, this makes it almost impossible for the model to extract information for those words. This could partly explain why rare answer candidates that do not appear often are difficult for the

model to predict. FastText provide word embeddings similar to GloVe, but contrary to GloVe, these provide features for words not encountered during the training of the embeddings. These embeddings are generated using a n-gram approach. We suspect that using FastText could help the model generalize over more answer candidates.

Finding a common embedding space for the models seems to be difficult for the models, as the model jointly have to learn this along with solving the task. We suspect that experiments with providing the model with word embeddings, of the predicted classed from the object detection model, could help the model with counting as well as positional questions. Likewise experiments, extracting image features from the words, by using the image vector for dog instead of the word vector could be interesting.

As seen in table 2 some models had very similar performance. The random initialization of the weights might have a say in this. In order to get a better sense of how the model performs across different initializations, we could run the models several times and report the standard deviation given those results. When working with deep learning, many assumptions are made regarding how the model interprets different parts of the input data. In order to understand the reasoning of the model better, we could trace back the flow of the model from the answer prediction to the inputs, to see what pixel values as well as words it weighs higher than other for a given example. LIME is a tool, that should allow us to do that[37].

Even though VQA models seem to achieve relatively good results, a sub-task within the VQA datasets have been made, called balanced pairs[14], where each question is assigned two disjoint answers, each with an image assigned. In order for the answer to be considered correct, the model have to answer each of the answers correctly. Teney, Anderson, He, and Hengel [10] report an accuracy of 34.66 on a model that otherwise have achieved 63.15. While this is far less impressive, we feel like this is a more suitable measure for the VQA models.

Acknowledgments

We would like to thank our supervisor Peter Dolog, and the Department of Computer Science at Aalborg University, for the guidance and discussions throughout this work.

References

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100, 000+ questions for machine comprehension of text", *CoRR*, vol. abs/1606.05250, 2016. arXiv: 1606.05250. [Online]. Available: http://arxiv.org/abs/1606.05250.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, 2009, pp. 248–255.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge", *Int. J. Comput. Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015, issn: 0920-5691. doi: 10.1007/s11263-015-0816-y. [Online]. Available: http://dx.doi.org/10.1007/s11263-015-0816-y.

[4] A. Santoro, D. Raposo, D. G. T. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. P. Lillicrap, "A simple neural network module for relational reasoning", *CoRR*, vol. abs/1706.01427, 2017. arXiv: 1706.01427. [Online]. Available: http://arxiv.org/abs/1706.01427.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", *CoRR*, vol. abs/1409.4842, 2014. arXiv: 1409.4842. [Online]. Available: http://arxiv.org/abs/1409.4842.

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

[7] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", *CoRR*, vol. abs/1506.02640, 2015. arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640.

[8] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks", *CoRR*, vol. abs/1506.01497, 2015. arXiv: 1506.01497. [Online]. Available: http://arxiv.org/abs/1506.01497.

[9] A. Jabri, A. Joulin, and L. van der Maaten, "Revisiting visual question answering baselines", in *European conference on computer vision*, Springer, 2016, pp. 727–739.

[10] D. Teney, P. Anderson, X. He, and A. van den Hengel, "Tips and tricks for visual question answering: Learnings from the 2017 challenge", *CoRR*, vol. abs/1708.02711, 2017. arXiv: 1708.02711. [Online]. Available: http://arxiv.org/abs/1708.02711.

[11] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and VQA", *CoRR*, vol. abs/1707.07998, 2017. arXiv: 1707.07998. [Online]. Available: http://arxiv.org/abs/1707.07998.

[12] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation", in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[13] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh, "Yin and Yang: Balancing and answering binary visual questions", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "VQA: Visual Question Answering", in *International Conference on Computer Vision (ICCV)*, 2015.

[16] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei, "Visual7w: Grounded question answering in images", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4995–5004.

[17] M. Malinowski and M. Fritz, "A multi-world approach to question answering about real-world scenes based on uncertain input", *CoRR*, vol. abs/1410.0210, 2014. arXiv: 1410.0210. [Online]. Available: http://arxiv.org/abs/1410.0210.

[18] L. Yu, E. Park, A. C. Berg, and T. L. Berg, "Visual madlibs: Fill in the blank image generation and question answering", *CoRR*, vol. abs/1506.00278, 2015. arXiv: 1506.00278. [Online]. Available: http://arxiv.org/abs/1506.00278.

[19] M. Ren, R. Kiros, and R. S. Zemel, "Image question answering: A visual semantic embedding model and a new dataset", *CoRR*, vol. abs/1505.02074, 2015. arXiv: 1505.02074. [Online]. Available: http://arxiv.org/abs/1505.02074.

[20] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L. Li, D. A. Shamma, M. S. Bernstein, and F. Li, "Visual genome: Connecting language and vision using crowdsourced dense image annotations", *CoRR*, vol. abs/1602.07332, 2016. arXiv: 1602.07332. [Online]. Available: http://arxiv.org/abs/1602.07332.

[21] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the v in vqa matter: Elevating the role of image understanding in visual question answering", *arXiv preprint arXiv:1612.00837*, 2016.

[22] J. Singh, V. Ying, and A. Nutkiewicz, "Attention on attention: Architectures for visual question answering (VQA)", *CoRR*, vol. abs/1803.07724, 2018. arXiv: 1803.07724. [Online]. Available: http://arxiv.org/abs/1803.07724.

[23] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering", *CoRR*, vol. abs/1606.00061, 2016. arXiv: 1606.00061. [Online]. Available: http://arxiv.org/abs/1606.00061.

[24] M. Malinowski, M. Rohrbach, and M. Fritz, "Ask your neurons: A neural-based approach to answering questions about images", *CoRR*, vol. abs/1505.01121, 2015. arXiv: 1505.01121. [Online]. Available: http://arxiv.org/abs/1505.01121.

[25] Q. Wu, C. Shen, A. van den Hengel, P. Wang, and A. R. Dick, "Image captioning and visual question answering based on attributes and their related external knowledge", *CoRR*, vol. abs/1603.02814, 2016. arXiv: 1603.02814. [Online]. Available: http://arxiv.org/abs/1603.02814.

[26] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks", in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 2692–2700. [Online]. Available: http://papers.nips.cc/paper/5866-pointer-networks.pdf.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[28] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation", *CoRR*, vol. abs/1406.1078, 2014. arXiv: 1406.1078. [Online]. Available: http://arxiv.org/abs/1406.1078.

[29] J. Kim, S. Lee, D. Kwak, M. Heo, J. Kim, J. Ha, and B. Zhang, "Multimodal residual learning for visual QA", *CoRR*, vol. abs/1606.01455, 2016. arXiv: 1606.01455. [Online]. Available: http://arxiv.org/abs/1606.01455.

[30] S. Wang and J. Jiang, "Machine comprehension using match-lstm and answer pointer", *CoRR*, vol. abs/1608.07905, 2016. arXiv: 1608.07905. [Online]. Available: http://arxiv.org/abs/1608.07905.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *CoRR*, vol. abs/1412.6980, 2014. arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014, issn: 1532-4435. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2670313.

[33] Z. Yang, X. He, J. Gao, L. Deng, and A. J. Smola, "Stacked attention networks for image question answering", *CoRR*, vol. abs/1511.02274, 2015. arXiv: 1511.02274. [Online]. Available: http://arxiv.org/abs/1511.02274.

[34] D. Ha, A. M. Dai, and Q. V. Le, "Hypernetworks", *CoRR*, vol. abs/1609.09106, 2016. arXiv: 1609.09106. [Online]. Available: http://arxiv.org/abs/1609.09106.

[35] 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 preprint arXiv:1606.01847*, 2016.

[36] ——, "Multimodal compact bilinear pooling for visual question answering and visual grounding", *CoRR*, vol. abs/1606.01847, 2016. arXiv: 1606.01847. [Online]. Available: http://arxiv.org/abs/1606.01847.

[37] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier", *CoRR*, vol. abs/1602.04938, 2016. arXiv: 1602.04938. [Online]. Available: http://arxiv.org/abs/1602.04938.

A. Apendix

A.1. Neural Networks

The architecture of a simple neural network generally consists of 3 layers. An input layer, which takes a set of features ($\mathbf{n}$) from our data as input ($\mathbf{x}$), and passes them on to the next layer. A hidden layer, which is a set of neurons ($\mathbf{h}$) that has a weight vector ($\mathbf{w}$) as well as a bias assigned to it. We take the dot product between the input and the weight vector, add the bias ($\mathbf{b}$), and then apply the activation function (σ). The result is passed on the the next layer ($\mathbf{z}$). The mathematical expression of this operation for a single neuron on the hidden layer, is described in equation (16). The output layer is basically the same as the hidden layer, except the output vector ($\mathbf{y}$) is the size of the desired output.

$$z_i = \sigma_i(\sum_j^n x_j \quad w_{ji} + b_i) \tag{16}$$

Feeding input data trough the network and transforming it trough the network parameters (W, B), is known as the *forward pass*. In order to make the network lean from the data we need two more steps.

After the forward pass we calculate the error (E) using our objective function L also known as the loss function. As shown on equation (17). Where y is the output and $\hat{y}$ is the target.

$$E = L(y, \hat{y}) \tag{17}$$

We want to minimize the total loss, using stochastic gradient decent or variations of it. To do this, we need to calculate the gradients of the network parameters. The gradients is calculated using *Backpropagation* which is applying the chain rule recursively trough the network.

Finally these steps are repeated until the error is minimized.

A.1.1. Activation Functions

To allow the neural network to approximate non-linear functions, a non-linearity is applied after each linear transformation. These non-linearities are called activation functions.

A common choice is the Rectified Linear Unit(ReLU), as it has been shown to perform well in deep neural networks, especially on computer vision tasks.

$$\sigma(z_i) = \begin{cases} x, \text{for } x > 0 \\ 0, \text{otherwise} \end{cases} \tag{18}$$

One of the nice properties of the ReLU functions is that it has a constant gradient of 1, when it is active, allowing gradients to propagate further in the network. However, networks using ReLU suffers from dead neurons, since both the output and the gradient

of the ReLU becomes 0 when it is inactive, which stops gradients from flowing completely, which can cause some neurons to never be activated, on any training example.

The Exponential Linear Unit(ELU) maintains the constant gradient of 1 for inputs greater than 1, but has a small gradient for negative inputs, allowing gradients to flow, and alleviating the problem of dead neurons.

$$\sigma_i(z_i) = \begin{cases} z_i, \text{ if } z_i > 0 \\ e^{z_i} - 1, \text{otherwise} \end{cases} \tag{19}$$

The output of the sigmoid is a value in the range $[0, 1]$, making it useful for producing outputs that can be interpreted as probabilites. We use the sigmoid at the output layer of the model, when treating the problem as multi-label classification.

$$\sigma(z_i) = \frac{1}{1 + e^{-z_i}} \tag{20}$$

The softmax activation function normalizes the inputs, so that they sum to one. For this reason its output is a valid probability distribution, and it is often used in multiclass classification problems. It is also used in the explicit attention model, to normalize the attention weights.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \; for \; j = 1..k \tag{21}$$

A.2. Convolutional Neural Networks

An image is usually represented by a number of pixels structured in a tensor with the dimensions Chanel × Width × Height ($C \times W \times H$) The images provided to VQA models are generally $224 \times 224 \times 3$. If a simple neural network were to take this as input it would be huge amount of parameters to train. Furthermore, the network would not capture the spacial relations of the pixels.

An effective way of dealing with these problems is by using Convolutional Neural Networks (CNN). A CNN is essentially a feed-forward network, that we slide across the image. This provides of with a number of features, on which we then can perform the same operations on, to gradually extract features until we are able to classify an image. Since objects locations may vary for different images. This smaller network learns a function able to detect different objects located in different places of the image. This is the notion of spatial invariance. A CNN consists of two main operations: convolutions and pooling.

Convolutions extract features from a image, called a feature map (z). These are extracted by what is known as a filter, which is a weights tensor ($\mathbf{W}$) that we stride across the image. We multiply the input values with the weight and sum them together. The filter size and the stride determine the the size of the resulting feature map. Considering a 5×5 image on which we apply a filter of size 3×3 with a stride of 1, would result in a feature map of size 3×3. To extract more features from the image we simply apply more features, which increases the depth of the resulting feature map. As well as in standard

neural networks, we add an activation function to be able to approximate non-linear functions.

The k-th entity of the feature map can be found using equation 22. Where i and j are the positions of the map and x_{ij} is the corresponding window of the input image.

$$z_{ij}^{k} = \sigma(\square \; W^{k} \quad x_{ij} + b^{k}) \tag{22}$$

The pooling operation is used to reduce the dimensionality of the features while keeping the most important. This is also known as downsampling. There a 3 different types of pooling: max, average and sum. For pooling we define a filter, apply one of the 3 types on all the elements in that filter, resulting in a down-sampled set of features. Considering a 4×4 feature map and a pooling operation with a 2×2 filter would result in a 2×2 feature map.

These two operations are used several times to gradually extract features and lowering the dimensionality. In order to classify an image a feed forward neural network is added at the end. Using the features the network is able to predict the labels. Similar to other neural networks the CNN is trained iterative using stochastic gradient decent, where the gradients are found using back propagation, until the error is sufficiently minimized.

A.3. Word vectors

To add additional information to the words, we can encode the words as word-vectors.

A popular way of encoding words, is by using GloVe [12]. GloVe is an unsupervised learning algorithm, that have been trained on different data sets(Common Crawl, Wikipedia, Twitter). The different data result in different word-vectors. On the website the different embeddings are available for download. Thus, we do not have to run the algorithm our-self. There exist feature vectors of different dimensions, spanning from 50d to 300d. The most commonly used vectors are 300d. Along with providing more information, having a dense vector with lower dimensionality reduces the number of weights in the first layer from the input layer. Since the input layer is of size 300 instead of the size of the vocab encoded as a one-hot vector. Since questions and some answers are represented as sentences, we aim to encode the whole sentence, which we can then pass along in the model. One way of representing a sentence is to treat it as a bag of words, an unordered set of word encodings. To obtain a sentence embedding from this bag of words, one simply sums or averages the word encodings in the bag of words. However, using this method, the sentence structure is discarded, which might be essential for certain problems.

A.4. Sequential Models

In order to capture the sequential information from the word order of sentences, a common approach is to use different variations of Recurrent Neural Networks (RNN).

RNNs have been proven to be able to solve sequential problem within the NLP domain. This is done by adding the notion of time to a simple feed forward neural network.

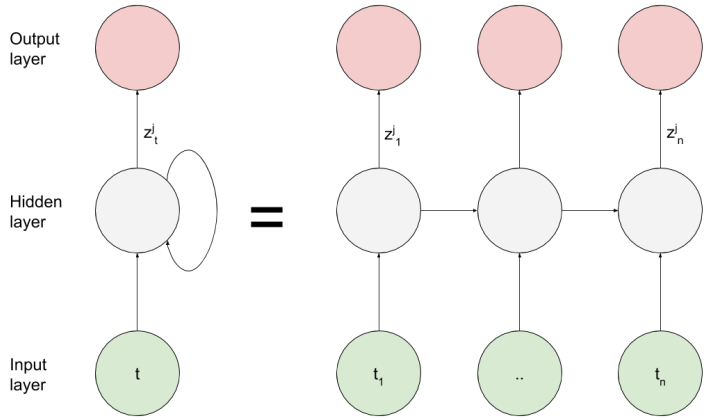

Figure 3: Visual representation of the notion of time in a RNN

In practice this is done by using a neural network for each time step (t) in the sequence $\mathbf{S}$, with shared parameters for each timestep. In order to capture the sequentiality, a weight vector $\mathbf{v}$ is added between the corresponding neurons in the hidden layer for each timestep. Visualized in figure 3. Likewise equation 23 shows the procedure for a single neuron in the hidden layer.

$$z_j^t = \sigma_j^t((\sum_i^{|n|} x_i^t \quad w_{ij}^t) + (\sum_i^{|h|} z_i^{t-1} \quad v_i^{t-1}) + b_j^t) \tag{23}$$

However, simple RNN models suffer from what is known as the vanishing gradient and exploding grading problem. When dealing with long sequences, the multiplicative effects with in the network, make the gradient either explode or vanish as we back propagate further along the time steps.

Popular RNN variations that deal with the vanishing gradient problem includes Long Short-term Memory(LSTM) and Gated Recurrent Unit(GRU). We use a GRU RNN, as it has been shown to perform as well as the LSTM, while being less computationally expensive