

# Guide Me: Interacting with Deep Networks

Christian Rupprecht<sup>1,2,\*</sup>, Iro Laina<sup>1,\*</sup>, Nassir Navab<sup>1</sup>, Gregory D. Hager<sup>2,1</sup>, Federico Tombari<sup>1</sup>

\* equal contribution

<sup>1</sup> Technische Universität München, Munich, Germany

<sup>2</sup> Johns Hopkins University, Baltimore, MD, USA

## Abstract

*Interaction and collaboration between humans and intelligent machines has become increasingly important as machine learning methods move into real-world applications that involve end users. While much prior work lies at the intersection of natural language and vision, such as image captioning or image generation from text descriptions, less focus has been placed on the use of language to guide or improve the performance of a learned visual processing algorithm. In this paper, we explore methods to flexibly guide a trained convolutional neural network through user input to improve its performance during inference. We do so by inserting a layer that acts as a spatio-semantic guide into the network. This guide is trained to modify the network’s activations, either directly via an energy minimization scheme or indirectly through a recurrent model that translates human language queries to interaction weights. Learning the verbal interaction is fully automatic and does not require manual text annotations. We evaluate the method on two datasets, showing that guiding a pre-trained network can improve performance, and provide extensive insights into the interaction between the guide and the CNN.*

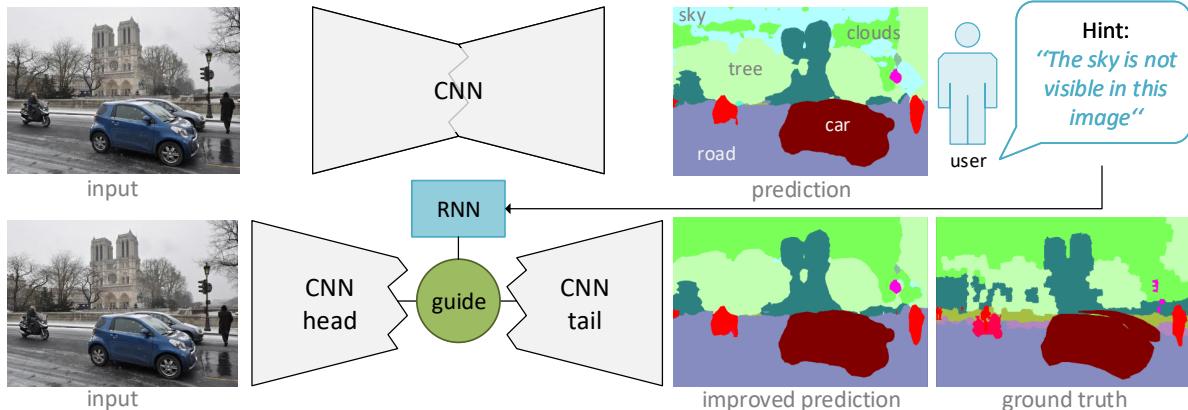
## 1. Introduction

Convolutional neural networks (CNNs) continue to grow in their breadth of application and in their performance on challenging computer vision tasks, such as image classification, semantic segmentation, object detection, depth prediction and human pose estimation. To date, the majority of the techniques proposed for these applications train specific network architectures once and subsequently deploy them as static components inside an algorithm. However, it is unlikely that any static network will be perfect at the task it was designed for. If the deployed CNN were adaptable to feedback or specifications provided by a human user *online*, this interaction would hold the potential to improve the model’s performance and benefit real-world applications.

For example, in photo editing, when a CNN is used to segment the foreground of an image from the background, the user might notice that the network has made a mistake. Instead of manually repairing the segmentation output or developing a post-processing algorithm based on some heuristics, a simpler and more effective way would be for the user to interact directly with the network through a directed hint, *e.g.* pointing out that “*the child on the bottom left of the image is in the foreground, not in the background*”. The user that was previously presented with a fixed, black-box prediction is now able to influence and alter the outcome according to his needs. This property is particularly useful in high risk domains such as medical image analysis and computer-assisted diagnosis, where fully automated segmentation is not always robust in clinical applications and the experience of trained practitioners matters. Another relevant example is speeding up labor-intensive and repetitive labeling tasks, such as those needed to create datasets for semantic segmentation, especially those for which annotations are scarce and expensive to obtain.

We propose a novel idea to allow user-network feedback-based interaction that aims at improving the performance of a pre-trained CNN at *test* time. The core idea is the definition of a spatio-semantic guiding mechanism that translates user feedback into changes in the internal activations of the network, thus acting as a means of re-thinking the inference process. The user input is modeled via a language-based approach, that enables interaction with a trained model in the form of a dialog. The user receives a first estimate, inputs a text query and the network replies with an updated prediction. Most previous interactive approaches place the user on the input/data side which means user input is required for the method to operate. In contrast, in our method, the user’s input is optional and modifies the network, this means that the network does not depend on human interaction but can be adjusted by it.

We showcase this interactive module on the task of semantic image segmentation. One advantage of our method is that it does not depend on any explicit annotation for text-



**Figure 1. Overview.** We introduce a system that is able to refine predictions of a CNN by injecting a guiding block into the network. The guiding can be performed using natural language through an RNN to process the text. In this example, the original network had difficulties to differentiate between the `sky` and the `clouds` classes. The user indicates that there is no `sky` and the prediction is updated, without any CNN weight updates and thus without additional training.

region correspondences. Yet, results indicate that the module can successfully transfer semantic information from the natural language domain to the visual domain, such that the network eventually produces a more accurate segmentation. As a side effect, this provides interesting insights into how CNNs structure their inference with respect to natural categories, providing an avenue for exploring the relationship between language and imagery.

## 2. Related Work

**Interaction with neural networks** Human-machine interaction is an extensively researched field. In [9] the user and an algorithm work together to solve fine grained recognition tasks, leveraging analytic strengths of the system and visual recognition power of the human. Prior to deep learning, several systems have been proposed for semi-automatic segmentation, that allowed the user to interfere with the result or to provide hints to the system via seed points [8, 52], bounding boxes [24, 37, 54], contours or scribbles [7, 23, 58], eye gaze [26] or in the form of binary yes/no answers to a set of questions [11, 55].

Most deep learning based segmentation methods, however, do not have an interface for human input during inference. The model and thus the attainable performance is fixed after the training phase. Directly integrating a human into a training loop with thousands of images is challenging. Nonetheless, some methods towards interactive deep learning have been proposed, such as weakly-supervised semantic segmentation from scribbles [38], user-provided clicks and Euclidean distance maps [61] or bounding boxes used as region initialization [17, 53]. Additionally, a method for sparse, user-guided colorization of grayscale images is proposed in [63]. In the field of medical imaging, [2] proposes to interactively improve segmentation by updating a seed-

map given by the user and [57] uses a second network operating both on the previous prediction and human feedback.

In our system, we integrate online interaction into the training by substituting the human input with an algorithm that dynamically generates hints from different modalities based on previous predictions. The CNN is already trained and only asks for the user’s directions for the purpose of conditioned (on-demand) adjustments of an initial estimate.

**The intersection of vision and language** To enable user interaction in a natural and intuitive way, we propose a novel idea that lies in the joint domain of natural language and vision. A relevant line of work in this field is Visual Question Answering (VQA). A question is posed and the answer is based on the image context [1, 45, 64]. Specialized systems for VQA ground the question in the input image and focus on the relevant parts to answer complex queries with text responses [4, 5, 27, 32, 43, 46]. Other examples include generation of referring expressions [36, 44, 47, 62], segmentation or object retrieval from referring expressions [19, 29, 28], image captioning [16, 21, 33, 34, 48, 56] and visual dialog [18]. Most works focus on the combination of CNN and RNN models, often building attention mechanisms [3, 42, 59, 60]. Most related to our approach is the recent method from [20, 50] that proposes the use of a conditional batch normalization layer [30] and feature-wise adjustment for visual reasoning.

A key distinction between our approach and most of the summarized literature is that our system’s output is visual and not textual, *i.e.* it is neither an answer nor an image caption. The output of the interactive CNN is in the same domain as the initial one. Another major difference is that we do not rely on vision-text correspondences with paired questions-answers or captions; user interaction is simulated via textual expressions that we generate automatically.

**Semantic segmentation** In this paper, we focus on the application of semantic image segmentation, which is widely studied in the computer vision literature and significantly boosted by the success of deep learning methods [6, 12, 31, 39, 41, 51]. Our goal is to deploy out-of-the-box, state-of-the-art models [12, 41] as estimators, that are then guided by our module to improve their former predictions with the help of a human user (or any given priors as hints).

### 3. Methods

In this section, we describe how our interaction module is integrated into a *fixed* CNN following two different approaches: guiding with user clicks and back-propagation (Section 3.2) or natural language inputs (Section 3.3). A general overview for the latter case is shown in Figure 1.

We first define the main elements of our framework, which we refer to throughout this paper. The module we insert into the CNN is called the *guide*. The guide interacts with the *guided CNN* through a *guiding block*, which is built to adjust activation maps of the CNN to improve the final prediction for the given input. The guided CNN is thus split into two parts: the *head*, which processes the input until it reaches the guiding block, and the *tail*, that is the rest of the guided network up to the final prediction layer. More formally, by decomposing the network into a head  $h$  and a tail function  $t$ , the output prediction  $\tilde{y}$  given input  $x$  can be written as  $t(h(x)) = \tilde{y}$ . We refer to the information that the guide uses to modify the guided network as the *hint*.

The split position is chosen manually. However, a reasonable choice is the (spatially) smallest encoding that the network produces, as this layer likely contains the most high-level information in a compact representation. We validate the choice of layers in the Section 4.2.

#### 3.1. Guiding Block

The guiding block is the integral piece of our approach, it enables feedback to flow from the guide into the guided network. Essentially, the guide must be able to modify a set of activations in the guided network. Since activation maps usually consist of a large number of elements (*e.g.*  $32 \times 32 \times 1024 \approx 1$  million), element-wise control is prone to over-fitting. The intuition behind our guiding block is that the network encodes specific features in each channel of a given layer. Thus, a guide that has the ability to strengthen or weaken the activations per channel, can emphasize or suppress aspects of the image representation and thus adapt the prediction in a semantically meaningful way.

The head predicts a feature representation  $h(x) = A \in \mathbb{R}^{H \times W \times C}$  with width  $W$ , height  $H$  and number of channels  $C$ . Then, guiding can be expressed as a per feature map multiplication with a vector  $\gamma^{(s)} \in \mathbb{R}^C$  and bias  $\gamma^{(b)} \in \mathbb{R}^C$ ,

$$A'_c = (1 + \gamma_c^{(s)})A_c + \gamma_c^{(b)}, \quad (1)$$

where  $c \in [1, \dots, C]$  indexes the channels of the feature volume  $A$  and the corresponding elements of the guiding vector  $\gamma = (\gamma^{(s)}, \gamma^{(b)})$ . Given this formulation, we are able to adjust a set of feature maps by emphasizing or hiding information per channel. Equation (1) can also be interpreted as the guide predicting a residual update (similar to ResNets [25]) for the activation map  $A_c$ .  $\gamma$  plays the role of a filter on the feature maps. When  $\gamma = 0$ , our guiding block reproduces the input feature map and thus has no effect in guiding the network. When  $\gamma_c^{(s)} = -1$ , channel  $c$  would become suppressed as all its units would be set to 0. Conversely, for  $\gamma_c^{(s)} = 1$ , the activation strength of that feature channel is doubled. Values smaller than  $-1$  *invert* a feature map, emphasizing aspects that would have been otherwise cut-off by the ReLU unit that typically follows the weight layer (or vice versa).

While this approach, which is similar to the conditioning layer in [20, 50], supports per-channel updates and feature re-weighting via  $\gamma$ , it is not flexible enough to adjust features spatially since it modifies the whole feature map with the same weight. In other words, it is impossible for this module to encourage spatially localized changes in each feature map (“*On the top left you missed ...*”). To overcome this limitation, we extend the approach to the spatial dimensions  $H$  and  $W$  of the feature map, *i.e.* we introduce two additional guiding vectors  $\alpha \in \mathbb{R}^H$  and  $\beta \in \mathbb{R}^W$  to modify the feature map  $A$  with spatial attention. In the following, we will index  $A$  with  $h, w$  and  $c$  to uniquely identify a single element  $A_{h,w,c} \in \mathbb{R}$  of  $A$

$$A'_{h,w,c} = (1 + \alpha_h + \beta_w + \gamma_c^{(s)})A_{h,w,c} + \gamma_c^{(b)} \quad (2)$$

The overall function that the *guided* network computes is thus modified to

$$y^* = t \left[ (1 \oplus \alpha \oplus \beta \oplus \gamma^{(s)}) \odot h(x) \oplus \gamma^{(b)} \right], \quad (3)$$

where the tiling of the vectors  $\alpha, \beta, \gamma$  along their appropriate dimensions is denoted with  $\oplus$  and the Hadamard product with  $\odot$ . This way  $\alpha$  and  $\beta$  have spatial influence and  $\gamma$  controls the semantic adjustment. Guiding with Equation 3 reduces the number of parameters from  $H \times W \times C$  to  $H + W + C = 1088$  in the previous example, which is manageable to predict with a small guiding module.

Since *fully convolutional* architectures are a common choice for image prediction tasks, we employ linear interpolation of  $\alpha$  and  $\beta$  when the feature map spatial resolution varies. This choice reflects two properties of the guiding block. First,  $\alpha$  and  $\beta$  do not depend on fixed  $H$  and  $W$ . Second, one can select the granularity of the spatial resolution by changing the dimensionality of  $\alpha$  and  $\beta$  to match the spatial complexity of the hints that the guide follows.

We describe two fundamentally different ways to employ the guiding block. The first one – guiding by back-

propagation (Section 3.2) – can be directly applied on a pre-trained CNN that is kept constant. The second one aims at online interaction with neural networks via user feedback. The network should be able to deal with hints from different modalities, such as natural language – “*the dog was mistaken for a horse*”. We describe how the guiding parameters  $\alpha$ ,  $\beta$  and  $\gamma$  can be predicted with an appropriate module given a hint from a different input domain in Section 3.3, which also speeds up processing.

### 3.2. Guiding by Back-propagation

In this setup, our goal is to optimize the guiding parameters such that the network revises its decision making process and, without further learning, improves its initial prediction for the current input. The guiding block is placed between head and tail, and the guiding parameters are initialized to 0. For a given sample  $x$ , we formulate an energy minimization task to optimize  $\alpha$ ,  $\beta$  and  $\gamma$ . The hint will be given as a sparse input  $\hat{y}$  associated to a mask  $\hat{m}$ .  $\hat{y}$  and  $\hat{m}$  have the same dimensionality as the prediction  $\tilde{y}$  and the ground truth  $y$ .  $\hat{m}$  is a binary mask that indicates the locations where a hint (*i.e.* prior knowledge) is given. Prior knowledge can be either directly given by the user or it could be a prior computed by another source.

In semantic segmentation, one can think of the hint as a single (or more) pixel(s) for which the user provides the true class – “*this [pixel] is a dog*” as additional information. Prior to guiding, a certain loss  $\mathcal{L}(t(h(x)), y)$  has been minimized during training of the network for a given task. We now optimize towards the same objective, *e.g.* per-pixel cross entropy for segmentation, but use the mask  $\hat{m}$  to only consider the influence of the hint and minimize for the guiding parameters, as opposed to the network’s parameters, *i.e.*

$$\alpha^*, \beta^*, \gamma^* = \underset{\alpha, \beta, \gamma}{\operatorname{argmin}} [\hat{m} \odot \mathcal{L}(y^*, \hat{y})]. \quad (4)$$

In this case, we only update the guiding variables for the *current* specific input  $x$  and hint  $\hat{y}$ , whereas the network’s weights are not trained further. The minimization finds the best parameters  $\alpha^*, \beta^*, \gamma^*$  conditioned on the hint. The key insight is that this results in an overall adjusted prediction.

Since the guiding block and the network  $t(h(x))$  are differentiable, we can minimize (4) using standard back-propagation and gradient descent with momentum. Intuitively, the tendency of gradient descent to fall into local optima is desirable here. We are looking for the smallest possible  $\alpha$ ,  $\beta$  and  $\gamma$  that brings the guided prediction closer to the hint while avoiding degenerate solutions such as predicting the whole image as the hinted class.

### 3.3. Learning to Guide with Text

While the previous idea is straightforward and simple to apply to any network, it requires the hint to be given in the

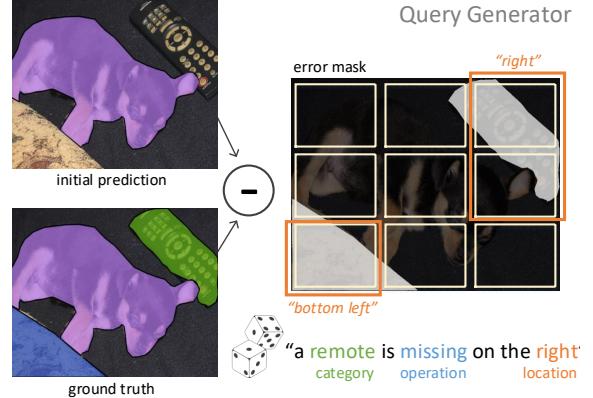


Figure 2. **Query Generator**. We illustrate the process to automatically generate queries to substitute the user during training.

same domain as the network’s output. We now explore a more natural way of human-machine interaction, in which the user can give hints to the network in natural language and the guiding mechanism is trained to update its parameters after parsing the user’s inputs. To the best of our knowledge, this is a topic that has not been previously studied.

**Training with queries** Similar to prior work in related fields, we use a recurrent neural network (RNN) for processing natural language inputs. We first encode the input query using a word embedding matrix, pre-trained on a large corpus, to acquire a fixed-length embedding vector per word. The embedded words are fed as inputs to a Gated Recurrent Unit (GRU) [14, 15] at each time step. We freeze and do not further train the word embedding alongside our guiding module, to retain a meaningful word representation. The guiding parameters  $\alpha$ ,  $\beta$  and  $\gamma$  are predicted as a linear mapping from the final hidden state of the GRU.

The language-guided module is trained as follows. We first generate an initial prediction with the fixed, task-specific CNN, without influence from the guide. We then feed prediction and ground truth into a hint generator, which produces a query (*e.g.* “*the sky is not visible in this image*”), thus mimicking the user. The query is then encoded into a representation that becomes responsible for estimating  $\alpha$ ,  $\beta$  and  $\gamma$  that will guide the feature map using (3) and subsequently update the prediction. The standard loss for the given task (pixel-wise cross entropy loss for semantic segmentation) is re-weighted giving positive weight (1) to the class(es) mentioned in the query to encourage changes in the prediction that coincide with the given hint. All wrongly predicted pixels are given a zero weight to prevent hints from being associated with other visual classes. Initially correct pixels are weighted 0.5 to discourage corrupting correctly classified regions.

**Generating queries** Previous work mostly relies on human-annotated queries, which make them rich in variety;

however, in this case they would limit the model to a single interaction, since new annotations cannot be recorded adaptively during training. Instead, our approach uses vision-only datasets and does not require visual/textual annotations, such as captions [40], referring expressions [28, 35] and region-description correspondences [47]. Our method aims at aiding the network to correct predictions with various mistakes, rather than producing a segmentation result on demand given an input expression. Therefore, it requires textual expressions that are synthesized on-the-fly from visual categories, by comparing the initial prediction and the ground truth segmentation map.

For the generation of the queries we use a combination of functionality, semantic categories and spatial layouts (see Figure 2). Functional categories are defined by a set of operations that can be carried out on the output to improve it, such as discovering missing semantic classes, suppressing noise or replacing wrongly predicted pixels with another class. The set we used in our experiments consists of two operations, *i.e.* `find` to handle classes missing in the initial prediction and `remove` to correct wrongly predicted labels.

Each query is built by its function and two placeholders, the entries of which are randomly selected at each training step from a set of plausible combinations based on prediction and the ground truth. We first divide the output of the network into a  $N \times N$  grid. In each grid cell, we search for all erroneous classes, either missing or mistaken, while ignoring tiny spurious regions comprised of only a few pixels. Next, we randomly sample a class from the generated list of possible choices and use its semantic name for the textual expression (*e.g.* “*find the person*”). The sampling probability is proportional to the potential improvement in the prediction. We then track the class position in the image based on the cells where it was found. Different combinations of cells define different spatial attention areas which can be then converted into text phrases such as “*on the top left*”, “*on the bottom*”, “*in the middle*”.

Eventually, the proposed approach can generate textual phrases automatically and online. The guide is thus trained to understand language using vision-only annotations, *i.e.* segmentation masks. The guiding block is able to discover semantic and spatial correspondences between the text input and the visual feature representation. During testing the guide can then interpret the commands of a real user.

## 4. Experiments

We evaluate our guiding framework under multiple aspects. First, we guide semantic segmentation by back-propagation. This allows us to directly evaluate the performance of the guiding and show how it can be deployed into a model without any additional training. Second, we thoroughly investigate guiding with textual hints.

#questions	0	1	5	10	15	20
<b>FCN-8s</b>						
mIoU	62.6	65.3	73.1	76.9	77.3	<b>81.0</b>
p.accuracy	91.1	91.8	94.1	95.3	96.0	<b>96.3</b>

Table 1. **Performance after a number of questions.** We guide a pre-trained FCN-8s [41] on PascalVOC 2012 *val* set [22] directly, using back-propagation. We report the mean intersection over union (mIoU) score and pixel accuracy. Every interaction with the user improves the result.

### 4.1. Guiding by Backpropagation

We investigate the performance gain by employing our guiding block directly on a fixed, pre-trained CNN. The task is semantic segmentation on the PascalVOC 2012 dataset [22]. We use a pre-trained FCN-8s network [41] and insert a guiding block in the smallest encoding layer.

A user interaction scheme similar to the 20-question game of [55] is set up. After an inference step, the network is allowed to ask the user for the class of a single pixel and the guiding layer updates the feature representation using (4). The queried pixel is the one with the smallest posterior probability difference between the two most confident classes. This pixel has the highest interclass uncertainty, meaning that it is the most likely to flip. After each question the prediction is updated and the mean intersection over union (mIoU) is computed.

We have intentionally chosen a somewhat “outdated” architecture since we believe that user interaction is mostly necessary in tasks in which the performance is not close to optimal. We list the performance after 0, 1, 5, 10, 15 and 20 questions in Table 1, where 0 denotes the initial performance of FCN-8s without guiding. Over the course of 20 interactions with the user, a significant improvement of the performance from 62.6% to 81.0% is recorded. It is noteworthy that the top entry on the PascalVOC leaderboard (DeepLab-v3 [13]) currently scores 86.9% mIoU, when trained with additional data. This demonstrates the benefit of guiding by back-propagation: it can be directly incorporated into a pre-trained CNN and, without any further training, it boosts a comparably low performance to reach the state of the art.

### 4.2. Guiding with Text Inputs

Due to the high accuracy of current methods on the PascalVOC semantic segmentation task, bringing a human into the loop to request improvements was not found to be meaningful with state-of-the-art models. We wish to evaluate our guiding module under a more challenging setting, in which even the performance of a state-of-the-art model is not satisfactory and interaction with a user can be beneficial.

For this purpose, we have chosen to use a dataset with

guiding module	mIoU	mIoU
	w/ res-blk	w/o res-blk
FiLM [50]	33.08	33.31
ours	33.11	<b>33.56</b>

Table 2. **Guiding Block Variants.** We evaluate mIoU performance when guiding res4a using `find` queries, in comparison to the conditioning layer of [50].

	res3a	res4a	res5a	res5c
mIoU	32.21	33.56	35.97	<b>36.50</b>

Table 3. **Location of the guiding block.** We evaluate mIoU performance when guiding different layers inside the CNN using `find` queries.

a limited number of images but rich categorical context. COCO-Stuff [10] is a subset of the popular MS Common Objects in Context (COCO) dataset [40] and consists of 10k images from the *train* 2014 set, further split into 9k training and 1k testing images. The images are labeled with pixel-level annotations of 91 “things” and 91 “stuff” classes.

**Implementation Details.** We first split the training set into two halves and use the first part for pre-training a DeepLab model [12] with a ResNet-101 [25] as back-end. The input dimensions are  $320 \times 320 \times 3$ . On this small, challenging dataset, this model scores only 30.5% mIoU. Next, we keep the weights of the semantic segmentation model fixed and only train the guiding mechanism using the remaining 4,500 images that were unseen during the pre-training phase. The guide is trained to translate embedded text queries through a recurrent model into relevant guiding parameters, as described in Section 3.3. For the word embedding we used a pre-trained matrix based on the GloVe implementation [49] that projects each word into a 50-dimensional vector space. The GRU consists of 1024 hidden units. A dense weight layer maps the last state to the vectors  $\alpha$  and  $\beta$ , that match respectively the height and width of the succeeding activations of the semantic segmentation model, and the weights and biases that are used as the scale and offset update for each activation map. We have experimented with two ways of applying the guide. The first one alters the CNN’s activations directly, therefore the weight vector size depends on the CNN layer that is being guided. The second wraps the predicted weights inside a residual block with 256 channels, as in [50]. For the hint generation process, instead of uniquely defining an operation as “*find the ...*” we randomly select from a set of variations with similar meaning such as “*the ... is missing*”, or “*there is a ... in the image*”. The grid size  $N$  is set to 3, resulting in 9 cells that specify the spatial location for the query. All experiments are averaged over five evaluation

hint complexity	guiding location	
	res4a	res5a
remove	31.53	32.56
find or rmv	32.22	33.73
find	33.56	<b>35.97</b>

Table 4. **Complexity of Hints.** We show performance of the method using two different types of hints.

# hints	0	1	2	3	4
mIoU	30.53	34.04	<b>35.01</b>	34.24	31.44

Table 5. **Guiding multiple times.** We guide iteratively with multiple `find or rmv` hints. After three hints performance decreases due to the guide over-amplifying certain features.

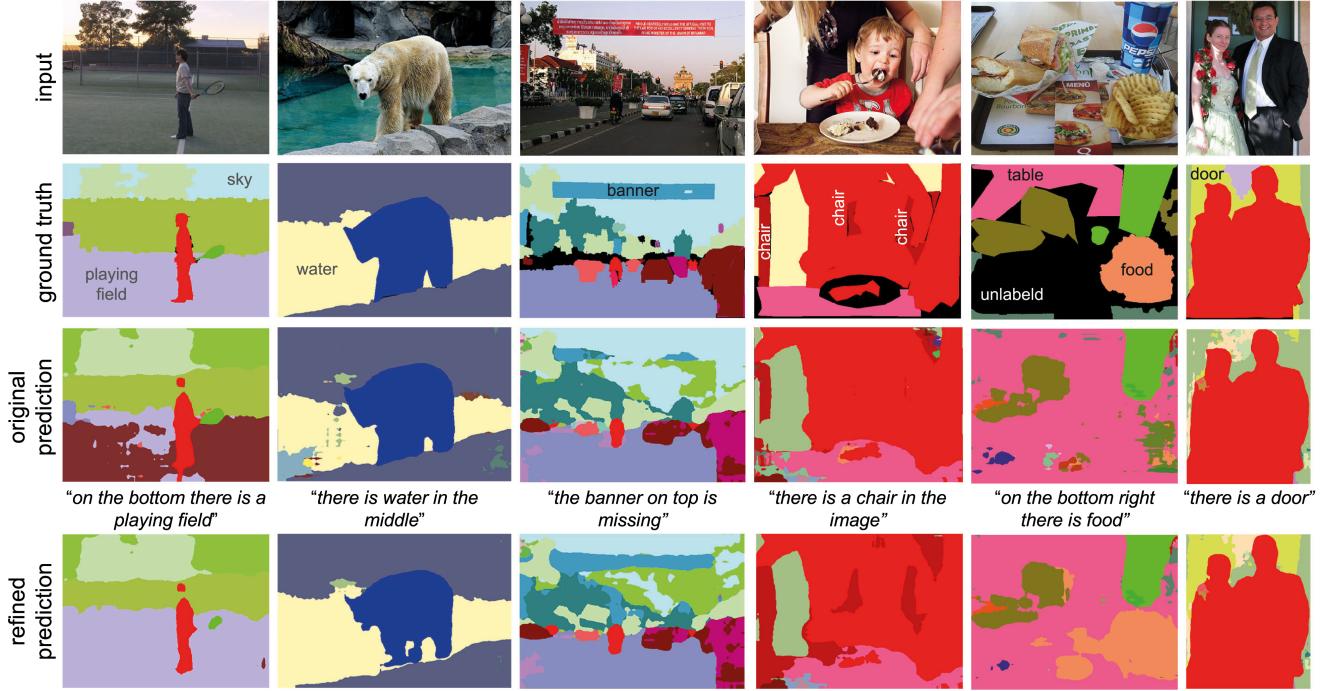
runs to account for the randomness in the queries.

Our best guided model improves the overall score from 30.5% to 36.5% with a single hint. We note that training DeepLab on the full train set is only marginally better than on the half, reaching 30.8% mIoU. Exemplary CNN predictions before and after guidance are shown in Figure 3. The guiding module was trained with `find` queries and does not modify the original CNN permanently, but only conditioned on the hints. We observed that our method helped resolving typical problems with the initial predictions, such as confusions between classes (columns 1, 2), partially missing objects (column 3, 4) and only partially visible objects in the background (column 6).

In the following, we compare our guiding block to the conditional batch normalization layer of [50]. Then, we explore the effect of guiding location by inserting the guide at different layers of the CNN. Further, we evaluate hint complexity using different query operations and apply repeated guiding to further improve the result. Finally, we provide some insights, by analyzing failure cases through heat map visualization and embeddings of the guiding vectors.

**Guiding Block Evaluation.** In a set of experiments we investigate different variants of the guiding block. The performance can be seen in Table 2. We analyze variants with and without an encompassing res-block around the guiding layer. We compare to the FiLM layer from Perez et al. [50]. The difference to our guiding block are the guiding components  $\alpha$  and  $\beta$ , that translate location information from the text to spatial attention in image space.

**Guiding Location.** Due to the flexibility of the guiding block, it can be plugged into the network at any location. In general, in our experiments we observe that a location that is very late - close to the prediction - inside the network often results in small, local changes in the output. Moving the block earlier results in more global changes that affect a bigger region and sometimes multiple classes. When the



**Figure 3. Qualitative Results.** We show qualitative results using `find` hints for missing classes. In the first example, we resolve a confusion between ground and playing field. In the second example, we show that the often occurring spurious predictions can also be handled. The third column shows that the network get the hint to find the banner, although it bleeds slightly into the building below. In the fourth and the last column, classes that are heavily occluded can be discovered too after guiding. The black ground truth label stands for unlabeled thus any prediction is allowed there. Please see the supplementary material for additional examples.

guide is placed too early in the network the feature maps that it guides do not contain enough high-level information to guide appropriately. This can be observed in Table 3, where we compute the mIoU score for guides in different locations inside the network.

**Complexity of Hints.** Automatically generating hints during training alleviates the need for manual vision-text annotations and also enables direct control of the query complexity. We differentiate between two distinct hints: `find` and `remove`. A `find` hint tells the network that it had missed a class: *"There is a person in the top right"*. `remove` is the opposite problem - the network had predicted a class that is not there or incorrect.

In Table 4 we show the performance for the different hint types. We observe that `remove` generally yields a lower performance gain than `find`. This is explained by the fact that `remove` is a more ambiguous query than `find`. When the network is told to remove a class from the prediction it does not know what to replace it with. Training with both queries simultaneously(`find` or `remove`), randomly selecting one each time, achieves average performance between the two types.

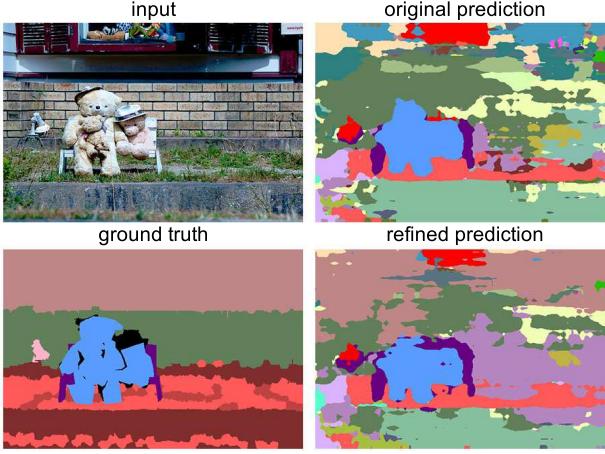
**Guiding multiple times.** We have conducted an experiment, similar to the one in Section 4.1 and Table 1, to show-

case an interesting property of the guiding module. Since it is trained to adjust the feature space in a way that improves the prediction, we hypothesize that the guided network can be guided repeatedly. The insight is that the guiding block will still result in a valid feature map. We iteratively direct the network (guided at layer res5a) to correct its mistakes via `find` or `rmv` queries, although it is not trained with subsequent hints, and report prediction accuracy in Table 5. We observe that the performance has further increased with a second hint. With three or more the guide starts to over-amplify certain features, causing noise in the predictions and decreasing performance. Nonetheless, we still observe a good gain over the non-guided model.

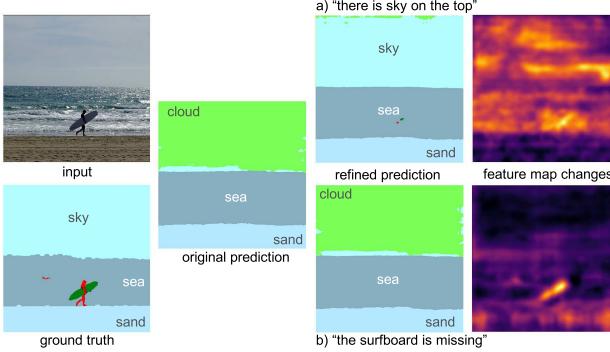
### 4.3. Insights into the Model

We provide further insight into learned models by examining failure cases and the learned joint embedding.

**Failure Cases.** When the initial prediction is particularly noisy, the guide has difficulties to fully repair the mistakes, as shown in Figure 4. Given a hint that a building is missing, the network can partly recover it, but a lot of spurious regions remain. We assume that the relevant features that would be needed for successful guiding, cannot be fully recovered from the noise in the guided activation map or are



**Figure 4. Failure case.** Hint: “*there is a building in the top*” When the initial prediction fails, our method has difficulties recovering the mistakes. The refinement includes the building only partially and it bleeds into `stone-wall` below.

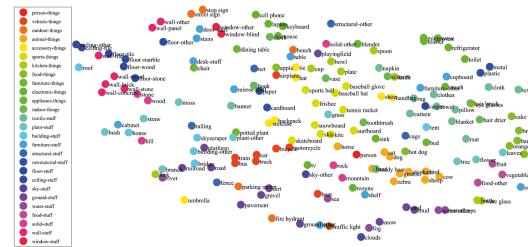


**Figure 5. Failure Case Visualization.** In the first example a) the refined prediction is correct. In b) the heatmap indicates that the guide has the right focus but it is not enough to change the output.

not present at all.

To understand how the activation map is influenced by the guide, we visualize a heatmap for different queries in the same image and investigate a failure case in Figure 5. In this visualization we can see that the system understands the hint about the `sky` (a). However, given the refined prediction for the `surfboard` hint (b), we would assume that it did not understand the query correctly. The heatmap shows that the guide indeed does emphasize the right parts of the image, but not strong enough to overpower the `sea` label. Potentially more precise queries during training could fix this problem. “*There is a surfboard where you predicted sea*” would let the guide not only emphasize `surfboard` related activations but simultaneously dampen the `sea` class, leading to better results in these cases.

**Semantic Analysis of the Learned  $\gamma$ -vectors.** We analyze the mapping from text to guiding vectors. To this end, we predict a  $\gamma$  vector for each class using a `find` query.



**Figure 6. Visualizing  $\gamma$ .** We visualize the learned  $\gamma$  vector for every class using t-SNE. The colors correspond to the higher level categories which are present in the dataset but not used in training. Best viewed in digital version.

In Figure 6 we display the t-SNE projection of these 256 dimensional vectors. The color categories that the 182 classes are grouped into, are set from higher level categories. The grouping into categories was never used during training. This space is the intersection between features learned from the CNN for segmentation and text representation learned by the RNN. The fact that semantically similar words cluster means that the joint embedding successfully correlates text and image features. A stronger clustering would mean that the  $\gamma$ -vectors are very similar inside the cluster, thus the network would have more difficulties guiding these classes. This can still be seen in a few cases such as the very close `sand` and `mud` classes, which are visually very similar and often do not improve after guiding.

## 5. Conclusion

In this paper, we have presented a system that allows for natural interaction of a human user with neural networks. The idea is to enable feedback from the user to guide the network by updating its feature representations on-the-fly, conditioned by the user’s hint, without further training the network’s parameters. An intuitive way of interaction is via text queries, sent by the human to the network, which aim at improving some initial estimation on a specific task.

We have created queries automatically with a specialized algorithm. In the future we would like to explore the possibility of generating queries with a second network that learns the role of the user, giving hints to the first. Further, image-guided attention mechanism can be incorporated into the RNN to improve the interaction mechanism.

## Acknowledgments

We would like to thank Robert DiPietro for discussions about the idea and Helen L. Bear, Helisa Dhamo, Nicola Rieke, Oliver Scheel and Salvatore Virga for proofreading the manuscript and their valuable suggestions.

## References

- [1] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Parikh, and D. Batra. Vqa: Visual question answering. *International Journal of Computer Vision*, 123(1):4–31, 2017. 2
- [2] M. Amrehn, S. Gaube, M. Unberath, F. Schebesch, T. Horz, M. Strumia, S. Steidl, M. Kowarschik, and A. Maier. Uinet: Interactive artificial neural networks for iterative image segmentation based on a user model. *arXiv preprint arXiv:1709.03450*, 2017. 2
- [3] 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. *arXiv preprint arXiv:1707.07998*, 2017. 2
- [4] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*, 2016. 2
- [5] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016. 2
- [6] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015. 3
- [7] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. icoseg: Interactive co-segmentation with intelligent scribble guidance. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3169–3176. IEEE, 2010. 2
- [8] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001. 2
- [9] S. Branson, G. Van Horn, C. Wah, P. Perona, and S. Belongie. The ignorant led by the blind: A hybrid human-machine vision system for fine-grained categorization. *International Journal of Computer Vision*, 108(1-2):3–29, 2014. 2
- [10] H. Caesar, J. Uijlings, and V. Ferrari. Coco-stuff: Thing and stuff classes in context. *arXiv preprint arXiv:1612.03716*, 2016. 6
- [11] D.-J. Chen, H.-T. Chen, and L.-W. Chang. Interactive segmentation from 1-bit feedback. In *Asian Conference on Computer Vision*, pages 261–274. Springer, 2016. 2
- [12] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 3, 6
- [13] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 5
- [14] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 4
- [15] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 4
- [16] B. Dai, D. Lin, R. Urtasun, and S. Fidler. Towards diverse and natural image descriptions via a conditional gan. *arXiv preprint arXiv:1703.06029*, 2017. 2
- [17] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1635–1643, 2015. 2
- [18] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, and D. Batra. Visual dialog. *arXiv preprint arXiv:1611.08669*, 2016. 2
- [19] H. De Vries, F. Strub, S. Chandar, O. Pietquin, H. Larochelle, and A. Courville. Guesswhat?! visual object discovery through multi-modal dialogue. In *Proc. of CVPR*, 2017. 2
- [20] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pages 6597–6607, 2017. 2, 3
- [21] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015. 2
- [22] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>. 5
- [23] L. Grady. Random walks for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1768–1783, 2006. 2
- [24] L. Grady, M.-P. Jolly, and A. Seitz. Segmentation from a box. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 367–374. IEEE, 2011. 2
- [25] 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*, pages 770–778, 2016. 3, 6
- [26] R. Hebbalaguppe, K. McGuinness, J. Kuklyte, G. Healy, N. O’Connor, and A. Smeaton. How interaction methods affect image segmentation: user experience in the task. In *User-Centered Computer Vision (UCCV), 2013 1st IEEE Workshop on*, pages 19–24. IEEE, 2013. 2
- [27] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. Learning to reason: End-to-end module networks for visual question answering. *arXiv preprint arXiv:1704.05526*, 2017. 2
- [28] R. Hu, M. Rohrbach, and T. Darrell. Segmentation from natural language expressions. In *European Conference on Computer Vision*, pages 108–124. Springer, 2016. 2, 5
- [29] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell. Natural language object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4555–4564, 2016. 2

- [30] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2
- [31] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1175–1183. IEEE, 2017. 3
- [32] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Inferring and executing programs for visual reasoning. *arXiv preprint arXiv:1705.03633*, 2017. 2
- [33] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2
- [34] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015. 2
- [35] S. Kazemzadeh, V. Ordonez, M. Matten, and T. L. Berg. Referitgame: Referring to objects in photographs of natural scenes. 5
- [36] S. Kazemzadeh, V. Ordonez, M. Matten, and T. L. Berg. Referit game: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 2
- [37] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 277–284. IEEE, 2009. 2
- [38] D. Lin, J. Dai, J. Jia, K. He, and J. Sun. Scribble-sup: Scribble-supervised convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3159–3167, 2016. 2
- [39] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. *arXiv preprint arXiv:1611.06612*, 2016. 3
- [40] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5, 6
- [41] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 3, 5
- [42] J. Lu, C. Xiong, D. Parikh, and R. Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *arXiv preprint arXiv:1612.01887*, 2016. 2
- [43] J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297, 2016. 2
- [44] R. Luo and G. Shakhnarovich. Comprehension-guided referring expressions. 2
- [45] M. Malinowski, M. Rohrbach, and M. Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1–9. IEEE Computer Society. 2
- [46] M. Malinowski, M. Rohrbach, and M. Fritz. Ask your neurons: A deep learning approach to visual question answering. *International Journal of Computer Vision*, 125(1–3):110–135, 2017. 2
- [47] J. Mao, J. Huang, A. Toshev, O. Camburu, A. Yuille, and K. Murphy. Generation and comprehension of unambiguous object descriptions. In *CVPR*, 2016. 2, 5
- [48] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014. 2
- [49] 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)*, pages 1532–1543, 2014. 6
- [50] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018. 2, 3, 6
- [51] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. *arXiv preprint arXiv:1611.08323*, 2016. 3
- [52] B. L. Price, B. Morse, and S. Cohen. Geodesic graph cut for interactive image segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3161–3168. IEEE, 2010. 2
- [53] M. Rajchl, M. C. Lee, O. Oktay, K. Kamnitsas, J. Passerat-Palmbach, W. Bai, M. Damodaram, M. A. Rutherford, J. V. Hajnal, B. Kainz, et al. Deepcut: Object segmentation from bounding box annotations using convolutional neural networks. *IEEE transactions on medical imaging*, 36(2):674–683, 2017. 2
- [54] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004. 2
- [55] C. Rupprecht, L. Peter, and N. Navab. Image segmentation in twenty questions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3314–3322, 2015. 2, 5
- [56] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015. 2
- [57] G. Wang, M. A. Zuluaga, W. Li, R. Pratt, P. A. Patel, M. Aertsen, T. Doel, A. L. David, J. Deprest, S. Ourselin, et al. Deepigeos: A deep interactive geodesic framework for medical image segmentation. *arXiv preprint arXiv:1707.00652*, 2017. 2
- [58] C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on image processing*, 7(3):359–369, 1998. 2
- [59] H. Xu and K. Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answer-

- ing. In *European Conference on Computer Vision*, pages 451–466. Springer, 2016. 2
- [60] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015. 2
- [61] N. Xu, B. Price, S. Cohen, J. Yang, and T. S. Huang. Deep interactive object selection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 373–381, 2016. 2
- [62] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg. Modeling context in referring expressions. In *European Conference on Computer Vision*, pages 69–85. Springer, 2016. 2
- [63] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros. Real-time user-guided image colorization with learned deep priors. *arXiv preprint arXiv:1705.02999*, 2017. 2
- [64] B. Zhou, Y. Tian, S. Sukhbaatar, A. Szlam, and R. Fergus. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167*, 2015. 2