From "Before" to "After": Generating Natural Language Instructions from Image Pairs in a Simple Visual Domain

Robin Rojowiec[1], Jana Götze[1], Philipp Sadler[1], Henrik Voigt[2], Sina Zarrieß[2] and David Schlangen[1]
[1]University of Potsdam [2]University of Jena
[1]first.last@uni-potsdam.de [2]first.last@uni-jena.de

Abstract

While certain types of instructions can be compactly expressed via images, there are situations where one might want to verbalise them, for example when directing someone. We investigate the task of *Instruction Generation from Before/After Image Pairs* which is to derive from images an instruction for effecting the implied change. For this, we make use of prior work on instruction *following* in a visual environment. We take an existing dataset, the BLOCKS data collected by Bisk et al. (2016) and investigate whether it is suitable for training an instruction *generator* as well. We find that it is, and investigate several simple baselines, taking these from the related task of image captioning. Through a series of experiments that simplify the task (by making image processing easier or completely side-stepping it; and by creating template-based targeted instructions), we investigate areas for improvement. We find that captioning models get some way towards solving the task, but have some difficulty with it, and future improvements must lie in the way the change is detected in the instruction.

1 Introduction

"*A picture is worth a thousand words*" – nowhere does that old adage seem to be more true than in the domain of assembly instruction giving. As Figure 1 illustrates, quite complex sequences of actions can be expressed compactly using pictorial instruction formats. [1] And yet, sometimes words are necessary; for example, when directing someone who has their eyes and hands on the object that is to be assembled. As a first step towards a vision of a collaborative, situated construction system, we present the task of *Instruction Generation from Before/After Images* (IG-BA).[2]

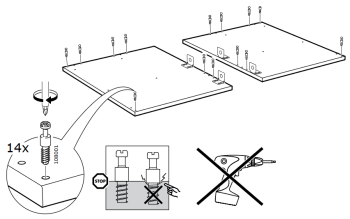


Figure 1: A Visual Instruction (Step 2 of Assembling IKEA's "Godmorgon" Case)

We leverage a related task that has become popular in recent years, that of Instruction Following (IF). In recent approaches to IF, a model is trained that is given an image and a (verbal) instruction, and predicts, within the context provided by the image, the action denoted by the instruction. (See, *inter alia* (Mei et al., 2016; Bisk et al., 2016; Anderson et al., 2018; Misra et al., 2018).) Here, we investigate whether this setup—and, more specifically, the BLOCKS dataset of Bisk et al. (2016)—can be used to derive data for IG-BA.

For IG-BA, we need pairs of images (depicting the *before* and the *after* state) as shown in Figure 2, and as output examples of instructions that when executed turn the former into the latter state. Making use of techniques developed for the task of *Image Captioning* (that is, the description of single images), we explore to what extent this setup allows us to sidestep the explicit planning steps

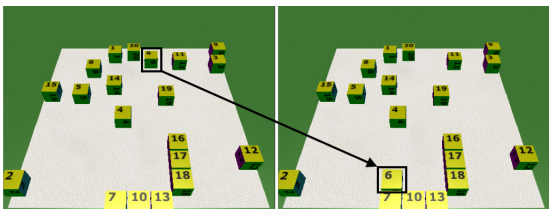


Figure 2: An image pair from BLOCKS annotated with the natural language instruction "Bring block 6 down and place it above block 7". The associated action is indicated with an arrow and boxes for visual aid.

[1]From https://bit.ly/34pmKCt, (C) IKEA.

[2]Related settings have been explored in human/computer interaction, e.g. by Kontogiorgos et al. (2018), but not from the perspective of NLP / natural language generation.

that were used in earlier Natural Language Generation work on instructions, for example in the context of the GIVE challenge (Byron et al., 2007).

The contributions of this paper are:

- Specification of the task of *Instruction Generation from Before/After Images* (IG-BA).
- BLOCKS$_{gen}$, a preparation of the BLOCKS dataset (Bisk et al., 2016) specifically for IG-BA.
- A set of baseline models for IG-BA, together with ablation studies pointing towards areas of future model adaptations.

2 Related Work

Generating descriptions of single images is by now a well-studied task (Bernardi et al., 2016; Mogadala et al., 2019), where the best-performing models typically make use of image encodings computed by Convolutional Neural Networks (CNNs) and caption generation (decoding) based on auto-regressive recurrent networks (Hossain et al., 2019). We will directly build on these approaches and discuss the ones we utilise in more detail below.

There is also some recent work that looks at a task which, similar to ours, takes image pairs as input. In *change captioning* (Oluwasanmi et al., 2019; Park et al., 2019), the task is to verbalise what is different between two otherwise very similar images. Our task contains this, but goes beyond it in that it also has to be verbalised how that difference can be effected. In that work, specialised architectures are presented that can more easily extract differences. Here, we wanted to start by exploring more standard captioning approaches as a baseline in order to fully understand the requirements of our task, leaving further architectural adaptations to future work.

After early work in the context of the GIVE challenge (Gargett et al. (2010), Byron et al. (2007)), there is some renewed interest in instruction giving. Köhn et al. (2020) presented an instruction giving platform called *MC-Saar-Instruct* where players can interact with a bot in the Minecraft world which instructs them to build something. This is very related to our interest in this project; for now, however, that work still assumes a symbolic representation as input.

In the field of natural language generation, the production of referring expressions is a well-established task (Krahmer and van Deemter, 2012), increasingly also tackled with neural methods (Zarrieß and Schlangen, 2018; Castro Ferreira et al., 2018). IG-BA includes this; but as will become clear in the next section, the data that we use here allows us to factor it out, as references to objects can simply be done via unique names. The complexity in our task comes from the spatial language required to denote locations, which is something not found to that degree neither in image captioning nor referring expression generation. Generation of spatial expressions has seen some attention in recent years, e.g. by Ghanimifard and Dobnik (2019a), who investigate the spatial language that neural language models can learn and express.

3 Data: BLOCKS and BLOCKS$_{gen}$

We will now describe how we can make use of data collected for instruction *following* for our task of instruction *giving*.

3.1 BLOCKS: Instruction Following

Bisk et al. (2016) collected the BLOCKS dataset in order to study instruction following in a simple visual environment. The environment consists of up to 20 blocks of the same size, which are placed on a board. The blocks are uniquely labelled either with a number between 1 and 20, or with the logo of a major company; this makes reference to blocks unambiguous and straightforward. To collect instructions, the authors created pairs of states s_t, s_{t+1}, represented as images i_t and i_{t+1} (computer-generated using a 3D-Engine), that differ in the placement of only a single block, and presented these to crowd workers who were tasked with producing a natural language instruction whose execution would turn s_t into s_{t+1}.

To create a sequence of such pairs, the authors started with a final configuration s_T which they successively distorted by moving a randomly selected single block to a random free location. This results in an initial state s_1 in which all blocks appear to be placed randomly. In the part of the dataset that we use, the final configuration constituted an interpretable pattern (numbers as sampled from the MNIST dataset (LeCun and Cortes, 2010)), with the assumption that this makes high-level instructions possible ("build a number 5"), and makes instructions towards the end of the sequence more easy to interpret. In the final configuration, the blocks are near-sorted (in that block 1 is likely to be placed near block 2, etc.). We discuss below consequences of this for use in our task. In the part of the dataset that we use here, only movements in the

	Train		Development		Test	
	logo	digit	logo	digit	logo	digit
Number of image pairs	667	652	95	96	181	172
Number of instructions	6003	5868	855	864	1629	1548
Number of sequences	35	35	5	5	10	10
Avg. number of blocks per sequence	19.23	18.91	19.08	19.33	18.76	18.0
Avg. number of image pairs per sequence	19.06	18.63	19.0	19.2	18.1	17.2
Avg. instruction length (num. of tokens)	14.60	13.60	16.35	17.23	15.30	13.30
Standard deviation of instruction length	7.09	8.15	8.39	9.34	7.10	7.36
Number of tokens	87,650	79,777	13,979	14,883	24,923	20,592
Number of types	715	650	405	391	470	388

Table 1: Statistics of the MNIST subset of the BLOCKS dataset.

plane were allowed (i.e., no stacking of blocks occurs). The pairs of images were presented to crowd workers, who were asked to formulate instructions that lead from the *Before* to the *After* state. Each pair was presented to three workers, who each produced three different instructions for it. Figure 2 shows an example of a state (image) pair and collected instructions. Table 1 gives an overview of the dataset through some statistics. Notable is the small size of the vocabulary, which is quite in contrast to typical image captioning, indicating that the crucial information is not in the lexical choice, but in the composition of the utterances.

To summarise, what the BLOCKS dataset gives us is a collection of state pairs (symbolically represented as well as rendered into images), ordered into sequences leading to a final state, where each transition is encoded linguistically in an instruction. Formally, each data point is of the form $(s_t^j, i_t^j, s_{t+1}^j, i_{t+1}^j, E_{t\to t+1}^j)$, where j is the index of the sequence, $1 \leq t \leq T$, and E is the set of instructions for a given state pair.

3.2 Requirements for a Dataset for IG-BA

As described above, the task of *Instruction Generation from Before/After Images* (IG-BA) consists in the generation of a natural language instruction given a pair of images, with the understanding that applying the instruction to the *Before* state would result in the state shown in the *After* image. Instructions can be given at different levels of specificity. As we are starting out with the IG-BA task here, we target what we call *simple instructions*, which map to actions that can naturally be seen as being atomic in their domain. At the same time, the task should be challenging enough to be interesting, both on the level of image understanding, and the level of language generation.

Prima facie, it seems that the BLOCKS data gives us what we want. By design of the dataset, the instructions that come with pairs of successive states are *simple instructions*, as the change they describe involves only a single block and can be effected with an atomic action. To specify this action, it is necessary to identify the block that has moved (we will call this block the *target block*). This has to be done by comparing the images in the pair, which is a visual task that goes beyond those found in image captioning. As all blocks are uniquely labelled, referring to blocks is straightforward in this domain. Denoting the target *location*, however, is challenging, as it requires the identification of a *landmark block* and its *spatial relation* to the target location. These are visual and linguistic tasks that are partially present in image captioning as well. However, in naturalistic scenes, spatial relations are to a large extent predictable (e.g., a rider will likely not be *under* the horse; (Ghanimifard and Dobnik, 2019b)), whereas here no systematic preference should be expected.

We will investigate in more detail in the next section whether these assumptions about the dataset do indeed hold and to which degree it is compositional.

3.3 Analysis of BLOCKS

3.3.1 Preprocessing

To support the analysis of the corpus (and later, for evaluation), we automatically process both the instructions and the symbolically represented scenes. (The results presented here are for the training split of the corpus; the preprocessing methods of course can also be applied to the other splits.)

The goal of the **instruction parser** is to extract from the instruction the reference to the *target block*, and if present, the reference to the *landmark* and the *spatial relation*, as in example (1) below.

(1) Input: *move the BMW block below the Adidas block*
Output: `bmw, adidas, below`

Pattern & Example	Logo	Digit	Total	Instr. Logo	Instr. Digit
`$BLOCK? (number)? $DECORATION` move **block 11** to the right of **block 6** **ups** should be south of **twitter** and northwest of **bmw**	5782	8931	14713	0.47	0.71
`(the)? (number)? $DECORATION $BLOCK?` move **the 14** to sit on top of **the number 15**, move **ups** so it is below **twitter**, **11** should be east of **6** move **the bmw block** below **the adidas block**	7344	4620	11964	0.60	0.36
Total	13126	13551	26777		

Table 2: A Simple Grammar For Block References; Examples and match counts; Proportion of Instructions containing a match (instruction can contain several patterns).

To do this, the candidate instruction is first run through the spellchecker hunspell and then dependency parsed, using spaCy's medium-sized pretrained model for English.[3] We identify block references by applying a small set of regular expressions (Table 2). We distinguish between target block reference (the one that moved) and landmarks using the rules shown in Appendix A. We extract string spans denoting the spatial relation by extracting phrases that function as modifier, such as prepositional modifiers, shown in the same appendix.

Evaluating whether the instruction parser correctly extracted the target reference is easy (assuming that the instruction is correct), as we can identify it from the symbolic representation. For the instructions in the test set, we obtain an accuracy of 89%. In 5% of the instructions, the parser does not identify any target block. When it comes to the landmark blocks, a fully objective evaluation is not possible, as here the instruction givers had some choice. We can identify from the state representation which blocks are close to the target location; 86% of the blocks that the parser identified as landmarks overlapped with this set in their respective scenes. We also manually checked 100 randomly selected parsed instructions against the images, and found an even slightly higher accuracy (88%). These numbers lead us to assume that we can put some trust in the output of the parser; which is important as it has a role to play in the evaluation of our models discussed below.

The **scene parser** extracts information from the symbolic state representations. Target block and potential landmarks are extracted as described above in the evaluation of the instruction parser. The objective spatial relation between target location and landmark candidate can be determined programmatically from the angle between the two coordinates. We segment the circle around the landmark candidate into equal-sized segments and classify the relation using compass directions `N, NE, E, SE, S, SW, W, NW`. We also similarly categorise the direction in which the target block was moved.

With this in hand, we can investigate the complexity of the language in the corpus (and hence that of the generation task), and whether there are any biases in the data that would give a learning algorithm opportunities to find shortcuts.

3.3.2 Complexity

As mentioned above, the main driver of linguistic complexity in this dataset is the specification of the intended target location. In a substantial subset of the training split our instruction parser identifies the mention of more than one landmark (14.4%; vs. 83.3% with only one, and 2.3% where none was identified). Table 3 shows some examples of spatial expressions, for the relative configuration *north of* and *southeast of*. For the cardinal direction *north*, there is more re-use of the same expression ("above"), indicating that this direction is somewhat easier to express. This is confirmed by the analysis in Table 4, which shows a tendency for verbalisations of diagonal directions to be longer and more unique. This is in line with previous findings on the complexity of expressions for different spatial configurations. Mast et al. (2014) found that segmentations of space that are finer-grained than the basic concepts of *left, right, front, back* require more complex spatial expressions.

3.3.3 Bias

First, we investigate whether the dataset contains biases for particular actions. As Figure 3 (left) shows, most movements of blocks were towards the right of the board. However, as the direction

[3] http://hunspell.github.io; https://spacy.io/models/en#en_core_web_md

N (N=2889)	SE (N=637)
above BLOCK (0.14)	to the right of BLOCK (0.08)
north of BLOCK (0.08)	southeast of BLOCK (0.06)
in the first open space above BLOCK (0.05)	below (0.06)
north of (0.04)	then (0.05)
so it is above BLOCK (0.04)	southeast of (0.04)
so its bottom edge touches BLOCK 's top edge (0.03)	up (0.03)
on top of BLOCK (0.03)	down (0.03)

Table 3: The most frequent spatial expressions for two of the 8 target-landmark configurations (*includes only instructions that mentioned exactly one landmark)

Relation	Instructions*	Tokens / phrase	Phrases / instr.	Unique phrases	Example phrase
N	2225	6.05	1.30	0.23	above BLOCK
S	1777	6.20	1.35	0.26	under BLOCK
E	1368	7.41	1.43	0.25	to the right of BLOCK
W	2121	7.40	1.41	0.21	left of BLOCK
NE	763	9.18	1.68	0.26	in the first open space northeast of BLOCK
NW	420	9.82	1.68	0.37	above and to the left of BLOCK
SE	358	10.34	1.78	0.44	so it is below and to the right of BLOCK
SW	766	9.23	1.60	0.31	in the first open space southwest of BLOCK

Table 4: Where in relation to the mentioned landmark the target block was put and how long (number of tokens) corresponding spatial extractions are as extracted by the instruction parser. Unique phrases is the proportion of unique phrases wrt. total number of phrases (*includes only instructions that mentioned exactly one landmark)

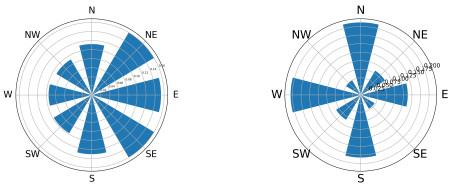


Figure 3: Left: Block movement on the board: relative count that the target block was moved in a direction (training data). Right: Target location relative to landmark.

of movement (as opposed to the relation between landmark and target location) is rarely mentioned in the data, this bias is harmless for the generation task.

Figure 3 (right) shows that among the landmarks mentioned by the instructors, cardinal directions dominate. In other words, in the data, the blocks that are to be placed more often form straight lines along the major axes.

We also investigated other potential influences of block configuration on the structure of the instructions. In the training data, 78.44% of the instructions contain a reference to the block that is positioned closest to the moved block's target position, leading us to assume that landmarks in this setting are typically chosen from the objects close to the target. However, this cannot be considered a (negative) bias in the data, but rather indicates a task-specific strategy.

We have already mentioned above that due to the design of the task in the original dataset, there is a bias for the landmark to be an "alphabetically close" block. (That is, in the final configuration, block 3 will be next to blocks 2 and 4, and so on; the same with a canonical ordering of the logo blocks.) This is a bias that a learner could indeed pick up on and use to simplify the task of naming a landmark; if the visual task of identifying the target block succeeded, a likely landmark can be named without verifying visually. However, correctly identifying the spatial relation still requires input from the scene.

3.4 BLOCKS$_{gen}$

We release BLOCKS$_{gen}$, an augmentation of BLOCKS for the tasks of IG-BA. Besides the instruction and scene parsers described above, the augmentation also contains a rule-based generator that creates simple and correct instructions out of the symbolic state representations. The templates we use are shown in the Appendix. The original dataset is available from https://groundedlanguage.github.io/. By the time of the conference, we will make our additions publicly available as well.

4 Baselines for IG-BA

4.1 Task Variants

The main task of IG-BA is as defined above: Given a pair (i_t, i_{t+1}) of *Before* and *After* images, generate a verbal instruction e that would tell an imagined instruction follower how to turn the former state into the latter; the generator is to be trained on the respective instructions $E_{t \to t+1}$ found in the corpus.

To analyse where the difficulties in this task lie for the various model types that we apply, we also define variants of this task at the image understanding and at the language generation phase. For *image understanding*, we support the model by performing simple pixel-wise operations, resulting in modified versions of the image pair: *subtracted* $(i_{t+1} - i_t)$, in which only the moved block remains visible; *added* $(i_{t+1} + i_t)$, in which all blocks remain visible, with the target block differing visually. Figure 4 provides an example respectively. In *both*, we put *subtract* and *add* side-by-side.

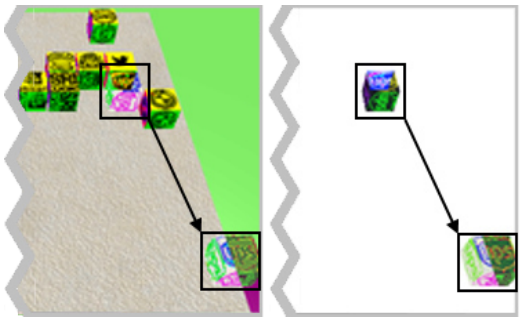


Figure 4: Snippets of samples for the *add* (left side) and *subtract* (right side) operation on an image pair. Arrows and boxes only for visual aid. The right side is inverted in colors for better printing.

We also side-step image processing completely, by turning the symbolic state representations into a 112-dimensional input vector: 20 dimensions for a one-hot encoding of the target block identity, 8 dimensions for the movement direction; and for up to 3 possible landmarks, their identity (20 dimensions) and spatial relation to the target location (8 dimensions), parsed from the states using the instruction parser (see 3.3). This variant will be called *symbolic-state* below.

On the *language generation side*, we simplify the task by using output of the template generator from section 3.4 as training material, hence massively reducing the variability on the output side. (Below, *synthetic-out*.)

These modifications give us a range of combinations that allow us to quantify the difficulty of the component tasks, with the expectation that *symbolic-state*/*synthetic-out* should be the easiest variant, and *full* (no preprocessing) the hardest.

4.2 Model Types

We also test different types of generation models.

NN-retrieve As a baseline, we use a retrieval approach (as done for example by Devlin et al. (2015) for image captioning). We put the images of the pair side-by-side and encode the resulting image using a pretrained ResNet101 (He et al., 2016) up to the final pre-classification layer ($d = 2048$). The width of the concatenated image is 240 pixel, the height 180 as we resize it after concatenation to half the size. At test time, we retrieve the nearest neighbour of the test image in the training data and sample an instruction from its instruction set.

CNN+LSTM A simple, but efficient approach to Image Captioning is to use an encoder-decoder architecture with a pretrained image model as encoder and a language model as decoder (Vinyals et al., 2014). We study to what degree this approach can also work for our task. We again encode images using a pretrained ResNet101(He et al., 2016), resulting in a 2048-dimensional representation. The language model is implemented using a standard LSTM with hidden size 512. We initialize the hidden state of the LSTM by feeding in the feature vector produced by the image encoder at time step t_{i-1}. Initial experiments showed that also adding the image vector at each successive step did not improve quality (supporting similar findings by Vinyals et al. (2014)).

CNN+LSTM+Attention This model type extends the previous one in two aspects: first, it initializes hidden and cell state of the LSTM directly, meaning the initial encoded image features are transformed with a linear layer and passed to the network. In the simple version, the hidden state is initialized by feeding the feature vector as input before the first time step. Second, an Attention mechanism is used to produce a weighted image feature vector on each time step conditioned by the current hidden state of the LSTM. For prediction, this feature vector is concatenated with the previously generated word and fed into the network, following Xu et al. (2015). We set the hidden size

of the LSTM to 512 and the Attention dimension to 128 dimensions. The images are encoded as above.

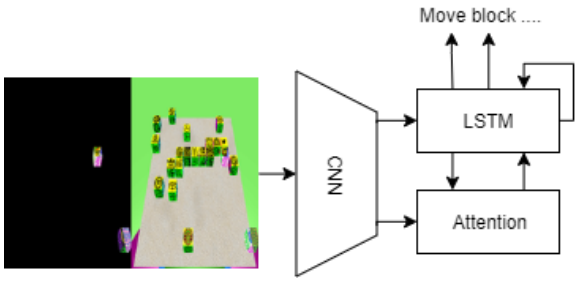


Figure 5: Architecture of the Show, Attend and Tell model applied to Blocks Dataset. The input image looks distorted, because we halve the original image widths for horizontal concatenation while keeping the heights.

Template Generator Given symbolically represented states, we can also use the template generator described above to generate instructions for the test set. The generated instructions are guaranteed to express correct information, at the cost of naturalness, and so can serve as an upper bound on semantic metrics.

4.3 Evaluation Metrics

In the test set, we have available both the symbolic representation of the states (and hence objectively know what the required change is) and the set of reference instructions E. We define metrics making use of either.

We use common metrics from caption generation: BLEU-4, measuring token overlap up to 4-grams (Papineni et al., 2002); CIDEr, measuring overlap based on the consensus of reference instructions (Vedantam et al., 2015), METEOR (Banerjee and Lavie, 2005), measuring unigram overlap with advanced normalization like stemming and synonym comparison, and ROUGE-L (Lin, 2004) which measures similarity based on longest common subsequences. We apply each individual metric by comparing the generated instruction against all available reference instructions for the respective image pair using the pycocoevalcap library.[4]

To better analyse task performance, where it matters that the blocks are correctly referred to, we also parse the generated instructions using our instruction parser, to extract what was mentioned as target block and as landmark. We can then compare these to either the objectively determined action parameters (we will call this variant *Ground Truth (GT)* below) or to those mentioned in the reference set (*Ref*). The target block will be identical in both, but as discussed above, there is some variance in which blocks were considered landmarks. For *GT*, we compare the landmarks against the three blocks closest to the target position (cf. 3.3.1).

$$S_{target} = \frac{\textit{Correct Targets}}{\textit{All Generated Targets}} \quad (1)$$

$$S_{landmarks} = \frac{\sum_i^N \frac{|G_i \cap P_i|}{|P_i|}}{N} \quad (2)$$

where N is the total number of predicted instructions. G_i is the set of all correct landmarks and P_i the set of all predicted landmarks.

4.4 Experiments

We train all of our models on the BLOCKS dataset using the original splits. Depending on task variant, we either provide a single template-generated instruction as training example, or the 9 human-generated ones from the original corpus. We train and evaluate separately for the *logo* and *digits* variants.

We train the models for a maximum of 50 epochs and early stopping with patience of 15 epochs. The parameters are optimized using the Adam Optimization Algorithm (Kingma and Ba, 2015). The CNN+LSTM is configured with hidden size and embedding size of 512 dimensions (except for the task variant using manually extracted feature vectors where the input size is 116 dimensions). The extended version using an attended image input at each time step t is configured with the same embedding and hidden size. Additionally, the attention dimensions are set to 128 and dropout with $p = 0.5$ is applied to the output of the LSTM before it is fed into the linear prediction layer.

5 Results and Discussion

All models reach a performance plateau within the maximum number of epochs. The results of the baselines are reported in Table 6 for experiments with the logo data. For the digit data, the performance in general is worse but supports the same conclusions (see Appendix C and D). This might be due to the image encoder working better with logos than numbers.

First, we present some samples of the generated instruction from the various models in table 5. They are randomly picked from the set of all generated

[4] https://github.com/salaniz/pycocoevalcap

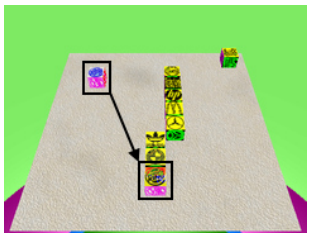
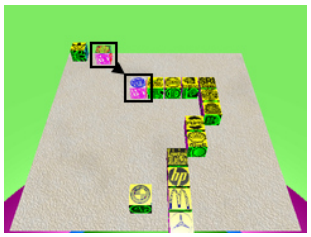
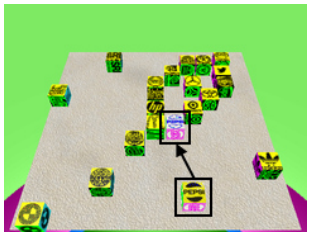
Image	Reference	Template	CNN+LSTM	CNN+LSTM+Att.
	move the burger king block directly south of the bmw block.	take block burger king and move it below block bmw	move the burger king block to the left of the coca cola block .	move the burger king block below the nvidia block.
	move the burger king cube directly next to the coca - cola cube touching it 's left side	take block burger king and move it to the left of block coca cola	move the adidas block to the left of the bmw block .	put the burger king block in the first open space to the left of the esso block.
	take the pepsi block from the bottom and move it to the right of and in contact with the mcdonald 's block in the center .	pick up block pepsi and move it to the right of block mcdonalds	move the shell block to the right of the mcdonalds block .	put the pepsi block in the first open space to the right of the mcdonalds block.

Table 5: Generated instructions by neural and template-based models for image pairs I_b (*add* and *subtract* modification combined) on logo data. The sample images provided here are not the ones used for prediction, because it would be much harder to see the individual blocks on those. Arrows and boxes only for visual aid.

instructions. Some of the generated instructions are not fully correct, e.g. the one in the first row of table 5 uses an incorrect reference block. This indicates the model's weakness in correctly understanding spatial relations. In the second and third sample, models use the correct relative position (left/right), so they were able to learn at least some of the spatial information. Next, we discuss the results in terms of the metrics presented in section 4.3.

NN-retrieve is the lower bound baseline and performs poorly. This indicates that the image encodings are not enough to retrieve appropriate instructions for an unseen image pair. The CNN+LSTM based models can improve on this lower bound. BLEU score jumps up by over 0.2 so the model is able to produce instructions that are more similar to the references than those NN-retrieve selects from the most similar pair. Target and landmark match improve as well, however only landmark by a bigger margin.

This model type improves further when presented with the pre-processed image (here, only *both* is shown; a more detailed discussion will be given below). It improves on all metrics, which suggests that the image component cannot on its own extract the information that the pre-processing makes available.

Adding an attention mechanism improves (CNN+LSTM+Att vs CNN+LSTM), but the gain of pre-processing the image (CNN+LSTM+Att+I_b) remains. Overall, that model achieves the best results (among the trained models). Interestingly, the improvement on the task-specific metrics *Target* and *Landmark* is not directly reflected by improvements on the string-based metrics. For example, on BLEU, the full model is substantially worse than the variant without attention, even though it is substantially better at naming the target block.

Generally, all generation models conditioned on images in Table 6 achieve a target accuracy that is rather unsatisfactory or even dramatically low (e.g. the CNN-LSTM model). Since instructions that do not mention the right target block have an extremely low chance of being communicatively successful when interacting with a user, none of the models can be considered ready to be tested in a dialogue set-up.

5.1 Model study

To investigate the contribution of the various component-tasks , we zoom into one of the mod-

Model	BLEU	METEOR	CIDEr	ROUGE-L	GT_T	GT_{LM}	Ref_T	Ref_{LM}
NN-retrieve	0.1057	0.0853	0.2567	0.0658	0.0559	0.0890	0.0726	0.1027
CNN+LSTM	0.3565	0.2764	0.2659	0.7114	0.0809	0.193	0.1098	0.2398
CNN+LSTM+I_b	**0.4238**	**0.3055**	0.6245	**0.724**1	0.3136	0.2857	0.3136	0.3071
CNN+LSTM+Att	0.3879	0.2812	0.2686	0.6505	0.1017	**0.3176**	0.113	0.2824
CNN+LSTM+Att+I_b	0.2766	0.1713	**0.6350**	0.3384	**0.5575**	0.2738	**0.5747**	**0.3214**
Template	0.3394	0.7528	0.4730	0.2632	1.0	1.0	1.0	0.9945

Table 6: Model performance on the BLOCKS data with logos with natural instructions. I_b denotes the *both* variant of the image. For the Template model, we compare each template instruction with the human references. In each column, the highest score (except those from template model) is marked in bold.

els, CNN+LSTM, and show its performance on the variant tasks defined above. (The attention model is not suited to work with the symbolic input representation and hence is not used here.)

Modification	Δ BLEU	Δ GT_T	Δ GT_{LM}
none	(0.3565)	(0.0809)	(0.193)
add	-0.026	0.0497	-0.0055
sub	-0.0583	0.0939	-0.0055
both	0.0673	0.2155	0.0387
state	0.2058	0.8287	0.6133
state + synthetic	0.5047	0.8950	0.7182

Table 7: Delta of CNN+LSTM performance with modified image inputs on logo data and natural instructions. (For *none*: actual value.)

As Table 7 shows, all image modifications provide useful additional information to the model, which improves its performance on naming the target block correctly. Interestingly, only the *both* variant (which was already listed in Table 6 above) leads to an improvement on the landmarks as well. Side-stepping the image processing (*state*) improves the output substantially, suggesting that the very simple image encoding that we use here, which served the image captioning task well in the original papers by Vinyals et al. (2014) and Xu et al. (2015), is not enough for this task. Finally, giving the model an easier generation target (*synthetic*) improves the performance to a degree that comes near that of the template generator. (This is perhaps not too surprising, but still good to see.)

As a final check, we report the performance of the template generation model (see Table 6). Compared to the CNN+LSTM model that is conditioned on state representations, it performs quite poorly on BLEU, as it captures very little of the variance of the natural data. However, on the task-specific metrics, it performs near optimal. (The difference to 1 on the landmark selection is due to the fact that the instruction givers sometimes picked differently from the scene parser, which is to be expected.)

6 Conclusion

We have introduced the task of *Instruction Giving from Before/After Images* (IG-BA), and shown that an existing dataset for instruction following can be used to train a model for this task. For the model, we used established architectures that have served the task of image captioning well. Through various analyses of the model, we have shown that their image processing capabilities seem to form a bottleneck; when provided with a pre-parsed scene representation, the quality of the generated instructions improves considerably. This suggests a clear route for future work, namely to improve the image processing capabilities, possibly along the lines of the recent, related work on *change captioning* (Park et al., 2019). The results that we have achieved are nevertheless encouraging that a performance can be achieved that can support our ultimate goal, which is to construct a system for interactive and collaborative instruction following in an assembly domain like shown in figure 1.

In comparison to the blocks data set, figures of assembly instructions are much more complex as they include changes of point of view, rotations on different axis and special informative pictograms (e.g. the cordless drill driver in figure 1). Additionally, the instructions may refer to previous steps or have more variability because of the less defined naming. For example, the figure above may requires the model to describe ”the screw with the long, smaller head must be screwed without a drill driver until its middle par hits the wood surface”. For future work, the language generation component of such system must be able to generate the instruction incrementally and describe the actions including advice of execution in the instruction.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *J. Artif. Int. Res.*, 55(1):409–442.

Yonatan Bisk, Daniel Marcu, and William Wong. 2016. Towards a dataset for human computer communication via grounded language acquisition. *AAAI Workshop - Technical Report*, WS-16-01 - WS-16-15:729–732.

D. Byron, Alexander Koller, J. Oberlander, Laura Stoia, and K. Striegnitz. 2007. Generating instructions in virtual environments (GIVE): A challenge and an evaluation testbed for NLG. *Proceedings of the Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation, Arlington*.

Thiago Castro Ferreira, Diego Moussallem, Ákos Kádár, Sander Wubben, and Emiel Krahmer. 2018. NeuralREG: An end-to-end approach to referring expression generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1969, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Saurabh Gupta, Ross B. Girshick, Margaret Mitchell, and C. Lawrence Zitnick. 2015. Exploring nearest neighbor approaches for image captioning. *CoRR*, abs/1505.04467.

Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Kristina Striegnitz. 2010. The GIVE-2 corpus of giving instructions in virtual environments. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Mehdi Ghanimifard and Simon Dobnik. 2019a. What a neural language model tells us about spatial relations. In *Proceedings of the Combined Workshop on Spatial Language Understanding (SpLU) and Grounded Communication for Robotics (RoboNLP)*, pages 71–81, Minneapolis, Minnesota. Association for Computational Linguistics.

Mehdi Ghanimifard and Simon Dobnik. 2019b. What goes into a word: generating image descriptions with top-down spatial knowledge. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 540–551, Tokyo, Japan. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December, pages 770–778. IEEE Computer Society.

MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. 2019. A comprehensive survey of deep learning for image captioning. *ACM Comput. Surv.*, 51(6).

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.

Arne Köhn, Julia Wichlacz, Christine Schäfer, Álvaro Torralba, Joerg Hoffmann, and Alexander Koller. 2020. MC-saar-instruct: a platform for Minecraft instruction giving agents. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 53–56, 1st virtual meeting. Association for Computational Linguistics.

Dimosthenis Kontogiorgos, Elena Sibirtseva, Andre Pereira, Gabriel Skantze, and Joakim Gustafson. 2018. Multimodal reference resolution in collaborative assembly tasks. In *Proceedings of the 4th International Workshop on Multimodal Analyses Enabling Artificial Agents in Human-Machine Interaction*, MA3HMI'18, page 38–42, New York, NY, USA. Association for Computing Machinery.

Emiel Krahmer and Kees van Deemter. 2012. Computational generation of referring expressions: A survey. *Comput. Linguist.*, 38(1):173–218.

Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. Technical report, New York University.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Vivien Mast, Diedrich Wolter, Alexander Klippel, Jan Oliver Wallgrün, and Thora Tenbrink. 2014. Boundaries and prototypes in categorizing direction. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8684 LNAI, pages 92–107.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. Listen, Attend, and Walk: Neural Mapping of Navigational Instructions to Action Sequences. In *AAAI'16: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2772–2778.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. 2018. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Aditya Mogadala, Marimuthu Kalimuthu, and Dietrich Klakow. 2019. Trends in integration of vision and language research: A survey of tasks, datasets, and methods. *CoRR*, abs/1907.09358.

Ariyo Oluwasanmi, Enoch Frimpong, Muhammad Umar Aftab, Edward Y. Baagyere, Zhiquang Qin, and Kifayat Ullah. 2019. Fully convolutional captionnet: Siamese difference captioning attention model. *IEEE Access*, 7:175929–175939.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Dong Huk Park, Trevor Darrell, and Anna Rohrbach. 2019. Robust change captioning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4623–4632.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575. IEEE Computer Society.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. Show and Tell: A Neural Image Caption Generator. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June-2015:3156–3164.

Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *32nd International Conference on Machine Learning, ICML 2015*, volume 3, pages 2048–2057. International Machine Learning Society (IMLS).

Sina Zarrieß and David Schlangen. 2018. Decoding strategies for neural referring expression generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 503–512, Tilburg University, The Netherlands. Association for Computational Linguistics.