Pento-DIARef: A Diagnostic Dataset for Learning the Incremental Algorithm for Referring Expression Generation from Examples

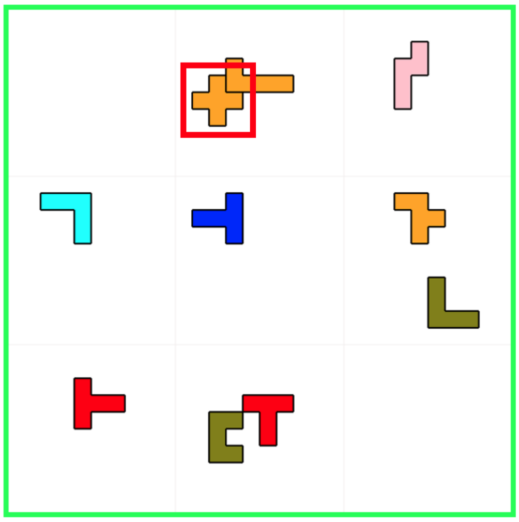
Philipp Sadler[1] and **David Schlangen**[1,2]
[1]CoLabPotsdam / Computational Linguistics
Department of Linguistics, University of Potsdam, Germany
[2]German Research Center for Artificial Intelligence (DFKI), Berlin, Germany
firstname.lastname@uni-potsdam.de

Abstract

NLP tasks are typically defined extensionally through datasets containing example instantiations (e.g., pairs of image i and text t), but motivated intensionally through capabilities invoked in verbal descriptions of the task (e.g., "t is a description of i, for which the content of i needs to be recognised and understood"). We present Pento-DIARef, a diagnostic dataset in a visual domain of puzzle pieces where referring expressions are generated by a well-known symbolic algorithm (the "Incremental Algorithm"), which itself is motivated by appeal to a hypothesised capability (eliminating distractors through application of Gricean maxims). Our question then is whether the extensional description (the dataset) is sufficient for a neural model to pick up the underlying regularity and exhibit this capability given the simple task definition of producing expressions from visual inputs. We find that a model supported by a vision detection step and a targeted data generation scheme achieves an almost perfect BLEU@1 score and sentence accuracy, whereas simpler baselines do not.

1 Introduction

Being able to effectively and efficiently *refer* to objects is a central component of human language competence (van Deemter, 2016). The computational task of referring expression generation (REG) goes beyond the production of image descriptions (as in image captioning), in that it is a *uniquely identifying* description that needs to be produced, given a specific situation. In the formulation of Krahmer and van Deemter (2012), the REG task involves reasoning over all relevant objects in a scene, in order to determine what would make a description uniquely identifying. Additionally, maxims of efficiency (Grice, 1967) predict that it is a *minimal* natural language expression that should be preferred. The *Incremental Algorithm* (IA) (Dale and Reiter, 1995) is a well-known classic symbolic algorithm that tries to realise these desiderata. For example given a reference target and various distractors as in Figure 1 (an example of the domain chosen in this paper (Pentomino, Golomb (1996); Zarrieß et al. (2016); Kennington and Schlangen (2017))), then the Incremental Algorithm (IA) produces "Take the X", achieving the desired uniquely identifying reference by mentioning only the shape and *not* also color and position.



(a) *"Take the orange X in the top"*
(b) *"Take the X"* (omit color and position)

Figure 1: An example board with a referring expression (b) as produced by the Incremental Algorithm (IA) (minimal wrt. a preference order); and an unnecessarily verbose reference (a) (that still is uniquely referring). The reference target is highlighted with bounding box; images regions separated by addition of lines.

Can such a reference strategy be learned by neural generation models from visual inputs alone? This is a question that is typically not systematically challenged in language generation from images (Kazemzadeh et al., 2014; Yu et al., 2016; Mao et al., 2016; Plummer et al., 2015; Luo and Shakhnarovich, 2017), as in natural scenes (such as in the RefCOCO dataset (Yu et al., 2016)), it has been shown that descriptions can be produced

Diagnostic Dataset	Task	Input	Condition	Output	Generalizability Testing
Wu et al. (2021)	Nav.	Symb. State	Text (Command)	Text (Actions)	Words, Phrases, Action Length
Liu et al. (2019)	REC	Image	Text (RE)	BBox	Color-Shapes
Pento-DIARef (Ours)	REG	Image	BBox	Text (IA-RE)	Color-Shapes, Positions, IA-REs

Table 1: The most relevant datasets in comparison to Pento-DIARef. In contrast to Liu et al. (2019) we study the task of REG (which avoids models to exploit language inputs) and add generalization tests for the output expressions.

based on the recognition of only parts of the image (Agrawal et al., 2016); our dataset is designed to make this impossible. Schlangen (2021) observed that in typical settings in the field of natural language processing, the connection between an underlying natural language capability and a learned model is only an indirect one. It rests on how well the dataset from which the model was induced does indeed exemplify the assumed underlying task—of which typically only a verbal description is given—and in turn on the extent to which the task represents the capability.

In this work we study how a intensionally defined task (in the distinction of Schlangen (2021)) for which a verbal and theoretically motivated description is given (through a symbolic algorithm) can be learned from its extensional exemplification. Our contention is that the use of synthetic data (Johnson et al., 2017; Liu et al., 2019; Lake and Baroni, 2018; Ruis et al., 2020; Wu et al., 2021) offers the opportunity to strengthen the link, insofar as guarantees can be given on the exemplification relation. More specifically, we choose the Incremental Algorithm (Dale and Reiter, 1995) for the data generation process, which itself comes with a motivation through recourse to underlying fundamental conversational capabilities (appeal to Gricean maxims, Grice (1967)). Our contributions are as follows:[1]

- We create a novel synthetic dataset, Pento-DIARef, of examples that pairs visual scenes with generated referring expressions;
- examine two variants of the dataset, representing two different ways to exemplify the underlying task;
- and evaluate an LSTM-based baseline (Mao et al., 2016), a transformer (Vaswani et al., 2017) and a modified version with region embeddings (Tan and Bansal, 2019) on them.

2 Related Work

Compositional Reasoning. Lake and Baroni (2018) introduced a systematic benchmark to test the generalization capabilities of recurrent neural networks through the use of compositional splits and found that these models fail "spectacularly". Ruis et al. (2020) extended the task of mapping text commands to actions (Navigation) by conditioning the learner additionally on a symbolic world state. Later (Wu et al., 2021) provided a curated dataset along with new dimensions for generalizability testing. Our work follows the idea of generalization testing through compositional datasets in language and vision settings where training examples are composed in such a way that the models are exposed towards all property values of objects, but not to all the possible combinations of them, so that they can be tested on unseen combinations. In contrast to their work we use images instead of symbolic world states as the input.

Diagnostic Datasets. For the generation of the synthetic data we took inspiration from Johnson et al. (2017) who created a "diagnostic dataset" for visual question answering to test for model limitations. They draw 3D objects on a 2D plane and systematically use templates to create questions about the objects to avoid biases that occur in "common" datasets. Later Liu et al. (2019) convert the questions to referring expressions to test systematically for referring expression comprehension (REC). They claim that the models' performance on the dataset proves that they "work as intended". In this work we study this aspect as well but on the mirroring task of REG which avoids models to exploit hints from the language inputs (Table 1).

Program Learners As a related idea Pi et al. (2022) suggest to train language models on text outputs of "executable programs" (which could be a symbolic algorithm). They focus on the pretraining paradigm and aim to induce reasoning capabilities into language models to enhance their usefulness for downstream tasks. Our work is more specifically focused on the question whether a neural model is able to learn the underlying capabilities that are exhibited by a symbolic algorithm in a vision and language domain. Same et al. (2022) showed that such rule-based algorithms are still a useful approach for REG in natural settings.

[1]The source code and datasets are made publicly available at `https://github.com/clp-research/pento-diaref`.

3 Pento-DIARef Task and Dataset

We present a **D**iagnostic dataset of IA **Ref**erences in a **Pento**mino domain (Pento-DIARef) that ties extensional and intensional definitions more closely together, insofar as the latter is the generative process creating the former (Schlangen, 2021). In this chapter we describe the task (§3.1) and how it is tied to the Incremental Algorithm (§3.2) via the generation process (§3.3) and present our compositional splits (§3.6) for generalization testing.

3.1 Task Description

Given as input an $x_i = (v_i, b_i)$, representing a Pentomino board v_i as in Figure 1 and a bounding box b_i (indicating the target piece), a model f has to produce a referring expression y_i (as it would be generated by IA) as shown in Figure 2. Formally, this can be described either as a classification task $\text{argmax}_{y_i} P(y_i|x_i)$ when y_i is considered a whole sentence or more generally as a conditional language modeling task $P(w_t|w_{<t}, x_i)$ with $y_i = \{w_0, ..., w_T\}$ where T is the length of the expression. We present models for both of these interpretations in Section 5.1.

3.2 The Incremental Algorithm (IA)

The Algorithm 1 , in the formulation of (Krahmer and van Deemter, 2012), is supposed to find the properties that uniquely identify an object among others given a preference over properties. To accomplish this the algorithm is given the property values $\mathcal{P}$ of distractors in M and of a referent r. Then the algorithm excludes distractors in several iterations until either M is empty or every property of r has been tested. During the exclusion process the algorithm computes the set of distractors that do *not* share a given property with the referent and stores the property in $\mathcal{D}$. These properties in $\mathcal{D}$ are the ones that distinguish the referent from the others and thus will be returned.

The algorithm has a meta-parameter $\mathcal{O}$, indicating the *preference order*, which determines the order in which the properties of the referent are tested against the distractors. In our domain, for example, when *color* is the most preferred property, the algorithm might return BLUE, if this property already excludes all distractors. When *shape* is the preferred property and all distractors do *not* share the shape T with the referent, T would be returned. Hence even when the referent and context are the same, different preference orders might lead to different expressions (Krahmer et al., 2012). We choose the preference order of *color, shape and position* for the algorithm; we leave experimenting with other orders to future work.

Algorithm 1 The IA on symbolic properties as based on the formulation by van Deemter (2016)

```
Require: A set of distractors $M$, a set of property
    values $\mathcal{P}$ of a referent $r$ and a linear preference
    order $\mathcal{O}$ over the property values $\mathcal{P}$
1: $\mathcal{D} \leftarrow \emptyset$
2: for $P$ in $\mathcal{O}(\mathcal{P})$ do
3:     $\mathcal{E} \leftarrow \{m \in M : \neg P(m)\}$
4:     if $\mathcal{E} \neq \emptyset$ then
5:         Add $P$ to $\mathcal{D}$
6:         Remove $\mathcal{E}$ from $M$
7: return $\mathcal{D}$
```

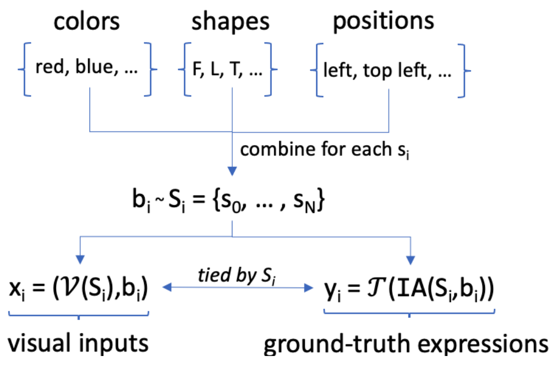


Figure 2: The general generation process for our synthetic datasets as defined by the task.

3.3 Data Generation

The inputs x_i for the task consist of two parts: the visual representation of the scene v_i and a bounding box around the target piece b_i. For the automatic generation of these inputs we make use of *symbolic board* representations $S_i = \{s_0, ..., s_N\}$ where N is the number of pieces on a board and s_i is a tuple of *color*, *shape* and *position* values e.g. (ORANGE,X,TOP). We define a mapping function $\mathcal{V}(S_i) \rightarrow v_i \in \mathbb{R}^{W \times H \times C}$ for rendering a board and sample uniformly from the symbols to select a target piece $b_i \sim S_i$ (for which we know the bounding box via $\mathcal{V}$). For simplicity, we use b_i to refer to the target bounding box in the visual domain or the target piece in the symbolic domain respectively.

As the ground-truth expressions we define $y_i = \{w_0, ..., w_T\}$ where T is the length of the expression and w_i is a word in the vocabulary. Again we make use of the symbolic piece representations (the same as above) to automatically generate the ground-truth by using the Incremental Algorithm. We apply the IA on the symbolic piece representations S_i and the target symbol b_i to select a set of property values $\mathcal{D}_i = \text{IA}(S_i, b_i)$ from the target b_i (Algorithm 1). These property values $\mathcal{D}_i$ are the shape, color or position values that are supposed to distinguish the target piece from other ones on the board. Finally, we define a mapping function $\mathcal{T}(\mathcal{D}_i) \rightarrow y_i$ to produce the ground-truth expression by filling the property values into pre-defined templates. The result of this process is a pairing of image and text, as you would find it for example in a captioning dataset (Johnson et al., 2016), albeit not collected from annotators but rather synthetically generated. In the following, we give more information on $\mathcal{V}$ and $\mathcal{T}$.

3.4 $\mathcal{V}$: Rendering the Pentomino boards

The symbolic piece representations in S_i are rendered as visual inputs v_i. We implement the rendering function $\mathcal{V}(S_i)$ that paints the symbolic pieces according to their shape and color values with black borders onto a board of 30 × 30 same-sized tiles. This underlying grid is projected onto 224 × 224 pixels. The exact tile coordinates of the pieces are determined by dividing the board into 9 distinct areas: one for each piece position value. To ensure that all pieces fit on the board, we allow maximal 2 pieces in a single area. We rotate and place the pieces one after the other into these areas by uniformly sampling tile coordinates that fall into the area that aligns to the piece position value. If two pieces collide during the placement, then we sample the coordinates again until they fit next to each other.

3.5 $\mathcal{T}$: Surface realization of IA outputs

The IA returns properties $\mathcal{D}_i$ of a target piece that distinguish it from other pieces. This list of properties is then transformed into a natural language expression. We define a mapping function $\mathcal{T}$ that inserts the property values into one of 7 different templates (Appendix C), for example "Take the *[color]* piece". We call these templates *expression types*. The mapping function selects the template based on the number of properties and the preference order: color, shape and then position. The word order in the templates is aligned with the preference order. We only use this order here to focus on the semantic correctness of the generation and leave mixing in additional variants like "Take the *[shape]* that is *[color]* in *[position]*" to future work. Altogether the property values and the templates lead to a vocabulary of 38 words—an extremely small vocabulary, which however as we are not targeting lexical complexity here is not a problem.

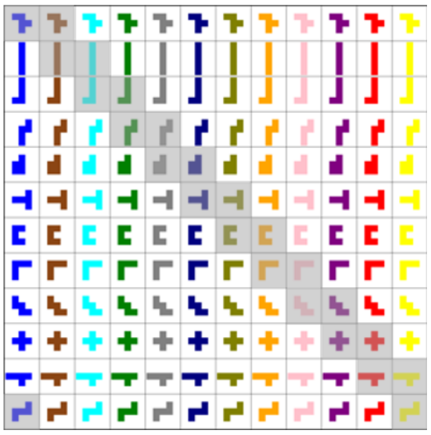


Figure 3: The shape and color combinations in grey are never seen during training, but only during evaluation.

3.6 Compositional Generalization

We make use of a synthetic dataset to guarantee the independence of properties and thus control, among other things, the compositionality of the learning task. There are 12 conventional names for the shapes which are roughly inspired by visual similarity to letters like *F*, *T*, *Y* etc. (Golomb, 1996). We sidestep the question of producing natural descriptions ("the one that looks like a boomerang") for the shapes and assume that these letter names can be produced. Furthermore, the pieces can appear in one of 12 different colors and the position can be approximated with 9 different spatial expressions (Appendix A.1, A.2). The permutation of colors, shapes and positions leads to $12 \cdot 12 \cdot 9 = 1296$ symbolic pieces to choose from for the composition of boards and the selection of targets. Now we create the training data from only $|S_{\text{train}}| = 840$ of the overall 1296 possible piece symbols and leave the remaining ones as a "holdout". These holdout pieces are specifically used to test the models' generalisation along three different dimensions, as in the following.

Piece appearances *(ho-color, 756 examples)*. The target piece shapes are combined with new colors with respect to the training set (Figure 3). For each of the 12 shapes we hold out 2 colors (val,test). Then we generate for each shape-color combination one board for each position and expression type.

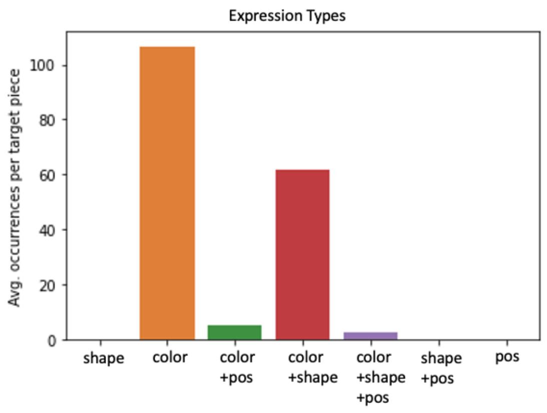


Figure 4: In the NAIVE dataset the synthesized examples highly prefer the generation of the two templates that only mention either the *(color)* or *(color,shape)* of a target piece. Some templates appear almost never [*(shape)*, *(shape,pos)*] or indeed never [*(pos)*].

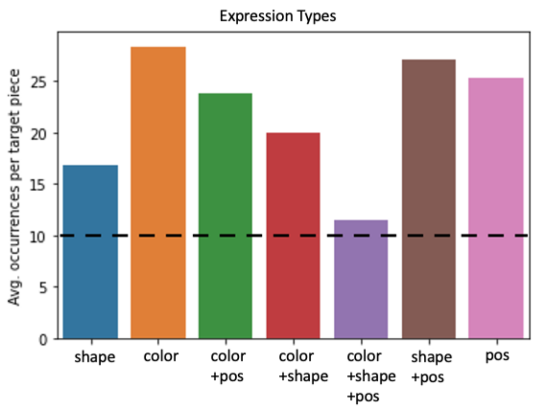


Figure 5: The DIDACT dataset is fairly balanced with respect to the expressions types. The figure shows the average count of expression types for each target piece after generating with ETOS. The dashed line indicate the distribution without the extra target selection.

Piece positions *(ho-pos, 840 examples).* In these examples the target pieces are shown at new positions with respect to the training set. For each of the pieces we hold out 2 positions (val,test). Then we generate for the holdout combinations one board for each expression type.

Expression types *(ho-uts, 840 examples).* We test that expression types are not attributed to specific pieces and show them in new contexts that leads to new expressions types wrt. the training set. For each of the pieces we hold out 2 expression types (val,test) and create corresponding boards.

4 What data is necessary to learn the IA?

The learned models have to generate an expression with exactly those property values (not more and not less) that the IA would produce. We hypothesise that learning the iterative set logic process and the preference order (either implicitly or explicitly) from text and visual inputs alone constitutes a challenging task for them, especially because given 12 shapes, 12 colors and 9 (discrete) positions for a piece (minus the combinations we excluded for holdout), then there are already around 20 billion possibilities to produce a board with 4 pieces on it.

Thus we make use of the fact that the generation process is fully under our control and directly ask what kind of data distribution is necessary to learn this task. We experiment with two different dataset variants: The first variant (§4.1) relies on an unconstrained sampling of symbolic pieces for each board while the second variant (§4.2) is designed to be more informative through a curated selection process.

4.1 NAIVE: Unconstrained Sampling

This process is meant to model the “naive” creation of a board by randomly sampling and placing pieces, as a person might do when setting up a board. We create these examples by randomly filling boards with pieces: First, we decide on the number of pieces that go on the board by sampling N from a uniform distribution over the integers 4–10. Then we sample uniformly with replacement $\{s_0, ..., s_N\}$ from the 840 symbolic pieces that are available for training. From the resulting symbolic board S_i we choose one piece, again uniform random, as the target piece b_{i_0}. Finally, we generate the input pairing (x_i, y_i) as described above.

We add one further constraint: We re-use the visual board v_i and pair it with 3 other target pieces $\{b_{i_1}, b_{i_2}, b_{i_3}\}$ chosen from S_i without replacement, so that a model *cannot* perform well by memorizing the (x_i, y_i) pairings alone, because then there are 4 identical visual boards with different targets that lead to (most likely) different expressions. This leads to 4 examples (x_j, y_j) per visual board with $x_j = (v_{i_0}, b_j)$. We repeat this procedure $42,000$ times which leads to $148,000$ training examples in total. The quantitative evaluation shown in Figure 4 reveals that here a model is most of the times confronted with expressions that only mention the *color* value or the *color and shape* of the target piece. The orange bar indicates that there are on average 100 examples (board and target) for each of the possible 840 target pieces where the color alone uniquely identifies the piece. So for around 84K samples in this dataset, a sentence like “Take the [blue, red, green,...] piece” would be correct.

4.2 DIDACT: Expression Oriented Sampling

The goal of the alternative sampling process is to ensure that examples of all *output* types are represented in the dataset, in a balanced way. We assume that this results in a more "didactic" dataset from which the underlying relation between input and desired output can more easily be induced. The idea is to directly choose the distractors of a target piece in such a way that the wanted expression type has to be produced. For example, when the target piece is (ORANGE, X, TOP) and the expression type is supposed to be *Take the [shape]*, then we construct a set of distractors where some share color and position, but none is of shape X. We call this approach *expression type oriented sampling* (ETOS) (details in Appendix B.1). This method allows us to confront the learner with all the possible expressions about the same amount of times. Thus each target piece is seen on 50 different boards resulting in $840 \cdot 50 = 42,000$ boards (Table 2).

Yet again we avoid that (x_i, y_i) pairs can be simply memorized and select as before 3 other pieces as the targets $\{b_{i_1}, b_{i_2}, b_{i_3}\}$ which leads to $148,000$ examples in total. The consequences of the extra target selection within this method are twofold: Firstly, the distribution is a bit shifted towards the NAIVE approach as shown in Figure 5 because we randomly select the target, and more importantly there might be now expressions produced that were actually intended for the holdout *(ho-uts)*. We remove such "unintended" examples from the training set so that there are $128,526$ examples for training (Table 3). Whereby the guarantees we can make for this "DIDACTic" dataset are that:

Target pieces appear with different distractors. Each target piece symbol for training appears on average in 153 contexts as a target.

On the same board occur different target pieces. We choose 3 additional pieces as targets apart from the one for which the board was initially intended.

Target pieces appear also on other boards. Each symbolic piece appears on average in $1,075$ contexts, which is more often than as a target.

Dataset / Num. of	TPS	pET	Bords per pET	Boards Total
NAIVE	840	7	-	42,000
DIDACT	840	5	10	42,000
ho-uts val	840	1	1	840
ho-uts test	840	1	1	840
ho-color val	108	7	1	756
ho-color test	108	7	1	756
ho-pos val	120	7	1	840
ho-pos test	120	7	1	840

Table 2: The number of target piece symbols (TPS) and possible expression types (pET) per TPS for the NAIVE, DIDACT and holdout datasets. Although the number of the resulting NAIVE and DIDACT boards is with $42,000$ the same, the boards are generated with different techniques: either with random uniform sampling (NAIVE; the number of boards per pET and TPS is not controlled for) or expression type oriented sampling (DIDACT; 10 boards for each pET and TPS).

	NAIVE dataset	DIDACT dataset
Number of Boards	42,000	42,000
TPS per Board	4	4
Number of Samples	168,000	168,000
Validation	10,000	10,000
Testing	10,000	10,000
Training	148,000	148,000
Filtered	128,526	-

Table 3: The number of samples for each dataset and training split. For each board we chose 4 target piece symbols (TPS) (incl. the originally intended target piece in the DIDACT dataset) resulting into $168,000$ samples for both datasets. From this overall samples we choose $10,000$ for validation and testing. In addition, we exclude the training samples of the DIDACT for which an expression is to be produced that should reserved for the *ho-uts* testing. This is not done for the NAIVE dataset because here we train on possibly all expression types.

5 Learning the Incremental Algorithm

Our goal in producing the collection of scenes was to ensure that a model f must indeed be based on features of the x_i that we care about (that is, which figure in the desired capability), namely the need to indeed compare the perceivable target piece and distractor properties. The IA (§3.2) achieves this by a hard-coded loop structure over symbols which (a) compares the objects (b) sticks to a preference order (c) preemptively stops when all distractors are excluded and (d) outputs the uniquely identifying properties (or all properties in ambiguous cases).

In the following, we present our neural models (§5.1) and the conducted experiments (§5.3) to test if neural language generators are indeed able to acquire such a "programatic" capability by the simple task definition of producing expressions from visual inputs. The generation models f will be trained on the basis of (x_i, y_i) pairs only. We train two common network architectures for this task of which one is an LSTM-based approach to REG proposed by Mao et al. (2016) for natural scenes and the other is a transformer (Vaswani et al., 2017). In addition, we propose a variant for processing the inputs along with a simple classifier-based baseline.

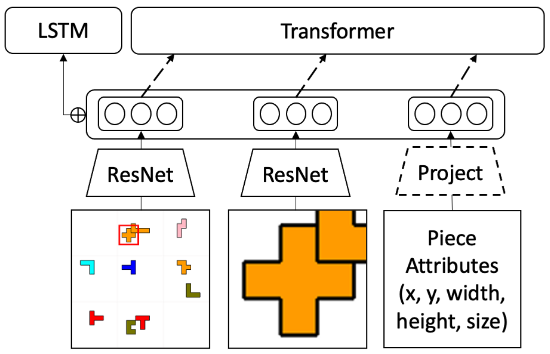


Figure 6: The encoding of the visual scene, target piece and its attributes as proposed by Mao et al. (2016). The dashed lines indicate the transformers' information flow.

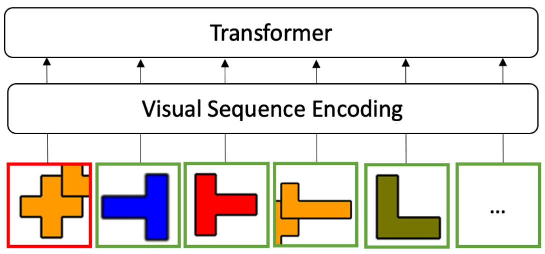


Figure 7: An exemplary visual input sequence for the Transformer+VSE model. The target is highlighted with a red and the distractors with a green border.

5.1 Models

LSTM. Mao et al. (2016), who present a model of REG in natural scene images, embed the scenes and the referent within them with a pre-trained VGG (Liu and Deng, 2015). We follow their procedure but use the 512 dimensional embeddings after global average pooling of a ResNet-34 (He et al., 2016) and fine-tune all of its layers because our images look very different to the ones from the pre-training on ImageNet (Deng et al., 2009). We cut out the target piece using the bounding box information. Then the piece snippet is dilated with 5 context pixels and up-scaled to the size of the board image. We additionally randomly shift the snippet by 0-5% of the pixels in either direction horizontally or vertically (fill-color is white). The target piece and board image embeddings are then concatenated together with five location and size features of the target. The resulting 1029-dimensional feature vector is fed to an LSTM at each time step to condition the language production (using greedy decoding). We reduce the word embedding dims to 512 because our vocabulary is very small and apply an Adam optimizer (Kingma and Ba, 2015).

Transformer. For comparison with Mao et al. (2016) we resize, augment and encode the target piece and visual board with a ResNet-34 in the same way as described before. Then the image embeddings are fed into the transformer (Vaswani et al., 2017) individually (not concatenated) as "visual words" together with the target piece attributes embedding as shown in Figure 6 to compute an intermediate representation of the inputs altogether. This "memory" embedding is then fed into the decoder to generate the RE using masked self-attention as in other machine translation tasks. For the variable length expressions we use a padding symbol and ignore prediction at padded positions during loss computation. We reduced the original capacity of the model to avoid overfitting and applied a learning rate scheduling strategy as described by Vaswani et al. (2017), using an AdamW optimizer (Loshchilov and Hutter, 2019).

Transformer+VSE. We assume that the transformer should be particularly capable of generating IA-like expressions because self-attention might allow it to learn the required piece-wise comparison operation. The self-attention mechanism has already been proven powerful for other image-related tasks (Li et al., 2020; Zhang et al., 2021; Jaegle et al., 2021). Therefore we follow Tan and Bansal (2019) and implement a *visual sequence encoding* (VSE) mechanism. For this we cut out each piece on the board to produce a sequence of piece snippets as shown in Figure 7 and project the visual features onto the models' input dimensions $\hat{f}_j$ and add region embeddings $\hat{p}_j$ to them that contain the snippets size and location information:[2]

$$\hat{f}_j = \text{LayerNorm}(W_F f_j + b_F) \quad (1)$$

$$\hat{p}_j = \text{LayerNorm}(W_P p_j + b_P) \quad (2)$$

$$v_j = (\hat{f}_j + \hat{p}_j + \hat{t}_j)/3 \quad (3)$$

To let a model distinguish between target and distractor "words" in the input sequence we add a type embedding $\hat{t}_j$, similar to word embeddings, and normalize. Furthermore, we have a variable amount of pieces on the board (between 4 and 10), but a transformer model assumes a fixed-size input sequence (per batch, during training). Thus we indicate "padding" pieces with a padding index in the sequence as implemented in PyTorch (Paszke et al., 2019) and use images with all zeros for them.

[2] b_F, b_P are bias terms of the linear projections

Model	Data	BLEU@1 (in %) ↑				Sentence-wise Acc. (in %) ↑			
		in-dist.	ho-color	ho-pos	ho-uts	in-dist.	ho-color	ho-pos	ho-uts
LSTM (Mao et al., 2016)	NAIVE	38	33	33	34	24	17	18	18
LSTM (Mao et al., 2016)	DIDACT	64	62	62	51	31	24	24	4
Transformer	NAIVE	27	23	23	23	21	14	14	15
Transformer	DIDACT	79	77	76	76	53	53	51	33
Transformer+VSE	NAIVE	59	53	57	54	29	22	22	25
Transformer+VSE	DIDACT	**97**	**97**	**97**	**97**	**91**	**91**	**91**	**92**
Classifier+VSE	NAIVE	32	25	28	28	27	15	19	22
Classifier+VSE	DIDACT	91	79	77	60	76	40	44	14

Table 4: The match rates for unigrams (BLEU@1) and whole sentences (SentA) on the test splits. The *in-dist.* samples are from the DIDACT dataset because these are more balanced with respect to the expression types.

Classifier+VSE. The representations of the VSE might already capture enough information to perform the task. Therefore we test this assumption by training a simple linear sentence classifier just on top of the concatenated embeddings. The classifier has to predict the correct sentence out of the 1,689 possible ones. This framing is similar to that often used in visual question answering (Hudson and Manning, 2019), where the possible answers are framed as classes in a classification task.

5.2 Metrics

We use the well known and commonly reported precision-based BLEU@1 metric for evaluation because this is simple metric for word matching when having only a single reference. In addition, we compute the *sentence-wise accuracy* (SentA) that indicates how often a prediction does exactly match the single reference so that the order of the words matters. As an example in Appendix E the model erroneously produces "Take the i top in the top left". We ignore the starting words "Take the" for the evaluation when they occur in both the prediction and the ground-truth, because then they are uninformative about the real performance.

5.3 Experiments

We perform separate training runs on both a NAIVE (§4.1) and DIDACT (§4.2) dataset for a maximum of 100 epochs and perform 10 validation runs during an epoch. Over all validation runs we save the three best performing models with respect to the BLEU@1 score using greedy decoding. We stop the training when the model does not improve anymore after 20 validation runs. For evaluation we choose the model with more epochs if the scores are the same. The training objective is to minimize the cross-entropy between the predicted and the ground-truth expression given by the Incremental Algorithm (IA).

6 Results and Discussion

NAIVE versus DIDACT. The results in Table 4 show that even the worst performing model trained on the DIDACT dataset (LSTM 31% in-dist) *is still performing better* than the best performing model trained on the NAIVE dataset (Transformer+VSE 29% in-dist) over all SentA scores (except ho-uts). This indicates that a well controlled data generation procedure is essential to perform well on this task, or conversely, that none of the learning algorithms can guess at the underlying minimality constraint from the unconstrained data alone. The SentA scores for the NAIVE-based models indicate that these often perform only about by chance (picking 1 of 7 templates leads to a score of 14%) on the compositional splits (highest 25% and avg. 18%). These splits contain all expression types in equal amounts and we find that these NAIVE models tend to produce only a few expression types.

Input triplets versus VSE. The results show that for both datasets a significant increase in performance is achieved by using VSE which includes a vision detection step. For the Transformer+VSE model the BLEU@1 scores double from 27% to 59% on the in-distribution test data. The simple Classifier+VSE model performs similarly well as the other models *without* VSE. This is reasonable because with VSE the visual encoder must *not* operate on two different image resolutions anymore: one for the (up-scaled) target piece and one for the whole context image. The VSE detection step "frees" capacities that would be necessary to correctly identify the content of the context image.

Classifier+VSE versus others. Almost all models struggle to perform well on both the compositional (<54% SentA) and the in-dist. test data (<77% SentA). Thus the Classifier+VSE establishes a relative high baseline on most of the test sets (76%/40%/44%) but only performs about by chance (14%) at the *ho-uts* data (which contains unseen expression types). The Transformer+VSE model is the only one that exceeds the high Classifier+VSE baseline by achieving almost perfect scores (91% SentA) over all categories when trained on the DIDACT data.

Effect of individual input features. We perform an ablation study to measure the impact of particular input features on the SentA scores. We do so by replacing the individual parts of the visual sequence encoding of our best model with noise sampled from a standard gaussian. We see that the visual embeddings are essential to generate the correct referring expression as the sentence-wise accuracy drops to 1% (Table 5). A similar performance drop is seen for the type embeddings where the accuracy is only 1-2%. A different impact is measured, when the region embeddings are replaced with random noise; here the accuracy is still around 40-44%. This is reasonable, because in only 4 of the 7 expression templates, the position (and therefore the region embeddings) are relevant.

Effect of DIDACTic training. We have a closer look on the tendencies of the models to produce certain expression types on the test data. For this we applied a parser to the predicted expressions of the models and counted the expression type occurrences. This provides insights, if a model tends to "overfit" on specific expression types. For example as the surface structure of the *color* expression types is seen in majority of cases during training, a model might simply try to produce *Take the [color] piece* and insert the referent color. We do not check for the correctness of the produced expressions here. The measures show that the LSTM model trained on the NAIVE dataset has converged on a behavior that produces in the majority of cases the *color* or *color+shape* expression type (Figure 8). This is reasonable as this is the majority class in the random sampling data. Only the DIDACT dataset let's them pick up on other expression types more regularly. The Transformer+VSE produces on the DIDACT test dataset rather balanced amounts of expression types (as these are given in the test data).

7 Conclusion and Future Work

In this work we presented the diagnostic dataset Pento-DIARef to study the question whether neural models can learn the RE production strategy of the Incremental Algorithm (IA). A symbolic algorithm

	Sentence-wise Acc. (in %) ↑			
Transformer+VSE	in-d.	ho-color	ho-pos	ho-uts
w/o visual emb.	1	1	1	1
w/o type emb.	2	1	2	2
w/o region emb.	44	42	41	40
full model	91	91	91	92

Table 5: The ablation study performed on the test dataset shows that the visual and type embeddings contain the crucial information for the RE generation.

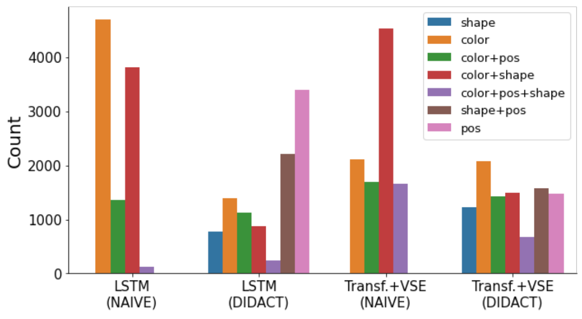


Figure 8: Produced expression types on the *in-dist.* split.

that is motivated by the appeal to the hypothesises capability of "elimination of distractors" (through the application of Gricean maxims).

We found through the better control on scene complexity that an unconstrained sampling method (NAIVE) does not provide enough information for a neural model to pick up on the underlying regularity and to exhibit the desired capability, while an output oriented sampling process (DIDACT) does. This indicates that the generalizability in this task and domain is not given by the capabilities of the learner alone but is strongly determined by the learning examples. We evaluated a classic LSTM-based model and a modern transformer (that have to process two different image resolutions) and observed that these still struggle even on the more informative dataset (DIDACT). We proposed a modification of the input processing that comes with a detection step (VSE) and observed that this leads to a strong baseline and allows the transformer to converge. This indicates that object detection is an essential requirement to perform well on this task.

In future work we want to evaluate more models on our diagnostic dataset to find potential weaknesses. An interesting question is whether a PLM (Brown et al., 2020) might have picked up such Gricean constraints and would be able to recognise their desirability from being prompted with only a few examples. We also plan to explore to what extent our best model is applicable to more realistic settings following *Sim-to-Real* approaches (Peng et al., 2018).

Limitations

Limits on visual variability and naturalness. The Pentomino domain can only serve as an abstraction for referring expression generations in visual domains. The amount of objects is limited to 12 different shapes and the number of colors is reduced to 12 as well. The positions are chosen to be discrete and absolute while real-world references might include spatial relations which we leave for further work. Furthermore, the pieces show no texture or naturalness, but are drawn with a solid color fill and a simple black border. Various lightning conditions that might impact a vision detection system are avoided. We left the evaluation of the proposed models on more realistic dataset for further work.

Limits on variability of the referring expressions. We only explored expressions that are generate by the Incremental Algorithm with one fix preference order of color, shape and position although we are aware of the fact that preference order might vary between subjects (Krahmer et al., 2012). Moreover, we choose a fix property value order (color is mentioned before shape is mentioned before position) for the realisation of the template's surface structure and left the exploration for a higher variability to further work.

Limits possible claims about human capabilities. As this work is on synthetic dataset created by an algorithm, any claims about human capabilities, and about a model's ability to acquire those, are only made indirectly, via the quality of the original algorithm.

Acknowledgements

We want to thank the anonymous reviewers for their comments. This work was funded by the *Deutsche Forschungsgemeinschaft* (DFG, German Research Foundation) – 423217434 ("RECOLAGE") grant.

References

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. 2016. Analyzing the Behavior of Visual Question Answering Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1955–1960, Austin, Texas. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Robert Dale and Ehud Reiter. 1995. Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science*, 19(2):233–263.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. ISSN: 1063-6919.

Solomon W. Golomb. 1996. *Polyominoes: Puzzles, Patterns, Problems, and Packings*. Princeton University Press.

Herbert Paul Grice. 1967. Logic and Conversation. In Paul Grice, editor, *Studies in the Way of Words*, pages 41–58. Harvard University Press.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Drew A. Hudson and Christopher D. Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6693–6702.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General Perception with Iterative Attention. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4651–4664. PMLR. ISSN: 2640-3498.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1988–1997. IEEE Computer Society.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. DenseCap: Fully Convolutional Localization Networks for Dense Captioning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*

(CVPR), pages 4565–4574, Las Vegas, NV, USA. IEEE.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 787–798, Doha, Qatar. Association for Computational Linguistics.

Casey Kennington and David Schlangen. 2017. A simple generative model of incremental reference resolution for situated dialogue. *Computer Speech & Language*, 41:43–67.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Emiel Krahmer, Ruud Koolen, and Mariët Theune. 2012. Is it that difficult to find a good preference order for the incremental algorithm? *Cognitive Science*, 36(5):837–841; discussion 842–845.

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

Brenden Lake and Marco Baroni. 2018. Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2873–2882. PMLR. ISSN: 2640-3498.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXX*, volume 12375 of *Lecture Notes in Computer Science*, pages 121–137. Springer.

Runtao Liu, Chenxi Liu, Yutong Bai, and Alan L. Yuille. 2019. Clevr-ref+: Diagnosing visual reasoning with referring expressions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4185–4194. Computer Vision Foundation / IEEE.

Shuying Liu and Weihong Deng. 2015. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734. ISSN: 2327-0985.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ruotian Luo and Gregory Shakhnarovich. 2017. Comprehension-guided referring expressions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3125–3134. IEEE Computer Society.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 11–20. IEEE Computer Society.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810. ISSN: 2577-087X.

Xinyu Pi, Qian Liu, Bei Chen, Morteza Ziyadi, Zeqi Lin, Yan Gao, Qiang Fu, Jian-Guang Lou, and Weizhu Chen. 2022. Reasoning like program executors. *CoRR*, abs/2201.11473.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2641–2649. ISSN: 2380-7504.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M. Lake. 2020. A benchmark for systematic generalization in grounded language understanding. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Fahime Same, Guanyi Chen, and Kees Van Deemter. 2022. Non-neural models matter: a re-evaluation of neural referring expression generation systems. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5554–5567, Dublin, Ireland. Association for Computational Linguistics.

David Schlangen. 2021. Targeting the benchmark: On methodology in current natural language processing research. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, pages 670–674. Association for Computational Linguistics.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

Kees van Deemter. 2016. *Computational Models of Referring*, chapter 4.6. The MIT Press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Zhengxuan Wu, Elisa Kreiss, Desmond C. Ong, and Christopher Potts. 2021. Reascan: Compositional reasoning in language grounding. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual.*

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. 2016. Modeling Context in Referring Expressions. In *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 69–85, Cham. Springer International Publishing.

Sina Zarrieß, Julian Hough, Casey Kennington, Ramesh R. Manuvinakurike, David DeVault, Raquel Fernández, and David Schlangen. 2016. Pentoref: A corpus of spoken references in task-oriented dialogues. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA).

Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. Vinvl: Revisiting visual representations in vision-language models. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 5579–5588. Computer Vision Foundation / IEEE.

A Experiment details

We trained each of our models on a single GeForce GTX 1080 Ti (11GB).

A.1 The vocabulary

The vocabulary includes the following 38 words:

- 12 shapes: F, I, L, N, P, T, U, V, W, X, Y, Z
- 12 colors: red, orange, yellow, green, blue, cyan, purple, brown, grey, pink, olive green, navy blue
- 6 position words: left, right, top, bottom, center (which are combined to e.g., right center or top left)
- 4 template words: Take, the, piece, at
- 4 special words: <s>, <e>, <pad>, <unk>

A.2 The piece colors (RGB-values)

Name	HEX	RGB
red	#ff0000	(255, 0, 0)
orange	#ffa500	(255, 165, 0)
yellow	#ffff00	(255, 255, 0)
green	#008000	(0, 128, 0)
blue	#0000ff	(0, 0, 255)
cyan	#00ffff	(0, 255, 255)
purple	#800080	(128, 0, 128)
brown	#8b4513	(139, 69, 19)
grey	#808080	(128, 128, 128)
pink	#ffc0cb	(255, 192, 203)
olive green	#808000	(128, 128, 0)
navy blue	#000080	(0, 0, 128)

Table 6: The colors for the Pentomino pieces. We also have 2 two-word colors: olive green and navy blue.

A.3 Uniform distribution of piece properties

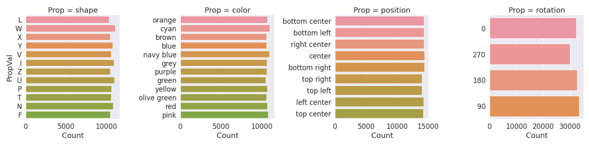


Figure 9: The occurrences of property values of target pieces in the DIDACT training data are almost uniform.

Our expression type oriented sampling strategy achieves an almost uniform distribution of piece color, shapes and positions (even rotations) as shown in Figure 9. We ignore the rotation property, but apply it to make the task harder. The model has to become invariant to the rotation.

B Data Generation

B.1 DIDACT dataset generation details

To construct this data, we iterate over all possible training symbols in S_{train} and set them as the target piece b_i directly (Table 2). Then we sample a symbolic board S_i (that includes the target) from the set of possible symbolic boards that lead to a wanted expression type u_i. For this we define the generator function $\mathcal{G}(u_i, b_i)$ that finds all possible symbolic boards that will be mapped by $\mathcal{T}$ so that $\{S_{u_i} | \mathcal{T}(\text{IA}(S_{u_i}, b_i)) \in \mathcal{Y}(u_i)\}$ where $\mathcal{Y}(u_i)$ is the collection of expressions that are represented by the template u_i, for example "Take the *[red, blue, green,...]* piece". In a sense $\mathcal{G}$ is the inverse of $\mathcal{T}$.

This method allows us to confront the learner with all the possible expressions about the same amount of times. We perform the example generation 10 times for each target piece and the according 5 training expression types (see §3.6 for holdouts). Finally, we generate the input pairing (x_i, y_i) as described in §3.1. Thus each target piece is seen on 50 different boards resulting in $840 \cdot 50 = 42,000$ boards. Yet again we avoid that (x_i, y_i) pairs can be learnt by heart and select as before 3 other pieces as the targets $\{b_{i_1}, b_{i_2}, b_{i_3}\}$ which leads to $148,000$ samples in total of which we filter the unintended ones (Table 3).

B.2 Holdout generation details

For the *ho-color* and *ho-pos* splits we additionally allow to choose distractors from the 840 symbolic pieces of the training split, because otherwise the distractor set of possible piece might become empty e.g. for the *ho-pos* split we have the target pieces only on a subset of possible positions, but need to place distractors in additional positions to produce all expression types.

C Expression Types

There are 3 expression types that are used when only a single property value of the target piece is returned by the Incremental Algorithm (IA):

- *Take the [color] piece*
- *Take the [shape]*
- *Take the piece at [position]*

Then there are 3 expression types that are selected when two properties are returned:

- *Take the [color] [shape]*
- *Take the [color] piece at [position]*
- *Take the [shape] at [position]*

And finally there is one expression type that lists all property values to identify a target piece:

- *Take the [color] [shape] at [position]*

In the following we exemplify the generated boards for each of the expression types.

C.1 Take the [color] piece

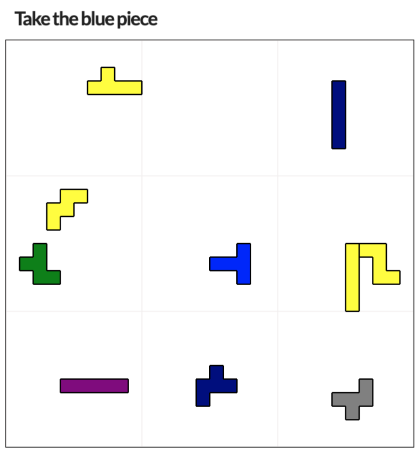


Figure 10: A sample board with the target piece `(T,blue,center)` for this expression type.

Mention the **color** excludes all. We add distractors with any shape or position, but a different color.

C.2 Take the [shape]

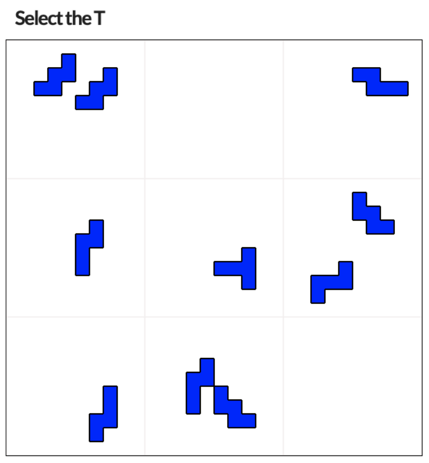


Figure 11: A sample board with the target piece `(T,blue,center)` for this expression type.

Mention the ~~color~~ does not exclude any. Mention the **shape** excludes all. We add distractors with the same color, but different shape and at any position.

C.3 Take the piece at [position]

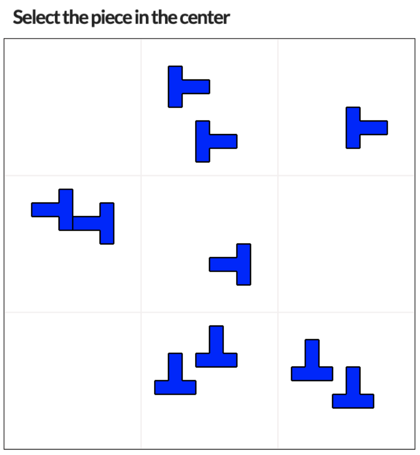


Figure 12: A sample board with the target piece (T,blue,center) for this expression type.

Mention the ~~color~~ does not exclude any. Mention the ~~shape~~ does not exclude any. Mention the **position** excludes all. We add distractors with the same color and shape, but at a different position.

C.3.1 Take the [color] [shape]

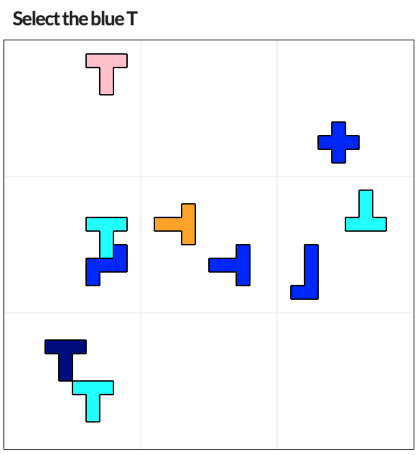


Figure 13: A sample board with the target piece (T,blue,center) for this expression type.

Mention the **color** excludes some, but not all. Mention the **shape** excludes the rest. We add some distractors with the same color (but different shape) and some distractors with the same shape (but different color) at any position.

C.3.2 Take the [color] piece at [position]

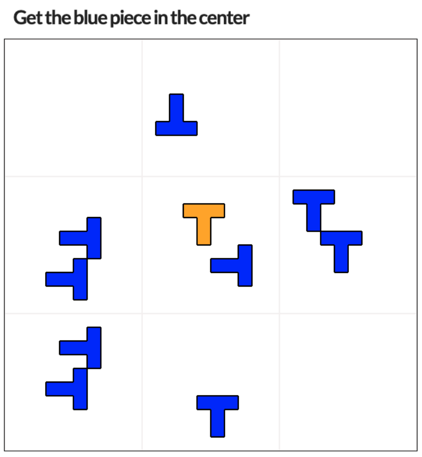


Figure 14: A sample board with the target piece (T,blue,center) for this expression type.

Mention the **color** excludes some, but not all. Mention the ~~shape~~ does not exclude any. Mention the **position** excludes the rest. We add some distractors with the same color (but different position) and some with the same position (but different color) and the same shape.

C.3.3 Take the [shape] at [position]

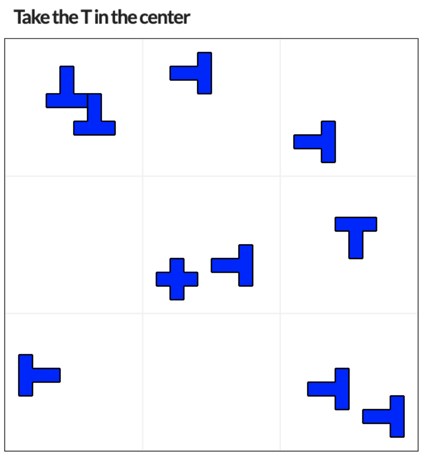


Figure 15: A sample board with the target piece (T,blue,center) for this expression type.

Mention the ~~color~~ does not exclude any. Mention the **shape** excludes some, but not all. Mention the **position** excludes the rest. We add distractors with the same color and some with the same shape (but different position) and some with the same position (but different shape).

C.3.4 Take the [color] [shape] at [position]

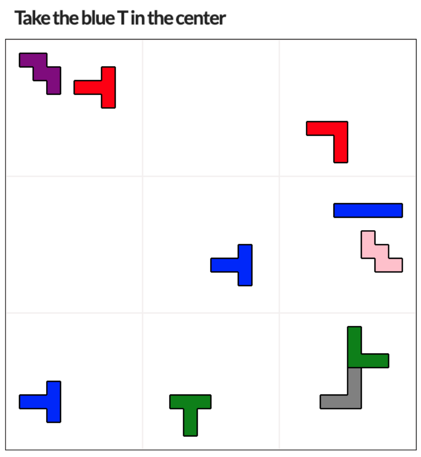


Figure 16: A sample board with the target piece (`T,blue,center`) for this expression type.

Mention the **color** excludes some, but not all. Mention the **shape** excludes some, but not all. Mention the **position** excludes the rest. We add one distractor that has the same color and shape (but a differen position) and one distractor that has the same color (but a different shape and position) and any other distractors. This requires at least 3 distractors.

D Model Details

D.1 LSTM

Parameters: $53,201,327$ (127 MB)
GPU RAM: $3,423$ MiB (Batch 24; VE)

lstm_hidden_size	1024
word_embedding_dim	512
visual_embedding_dim	512
dropout	0.5
lr	0.0003
l2	0.0001
gradient_clip_val	10

Table 7: LSTM hyperparameters

D.2 Classifier

Classes: $1,689$
Parameters: $30,234,718$ (120 MB)
GPU RAM: $11,063$ MiB (Batch 24; VSE)

d_model	512
visual_embedding_dim	512
lr	0.001
l2	0.01
layer_norm	0.00001
gradient_clip_val	10

Table 8: Linear model hyperparameters

D.3 Transformer

Parameters: $37,402,090$ (149 MB)
GPU RAM: $10,871$ MiB (Batch 24; VSE)

d_model	512
word_embedding_dim	512
visual_embedding_dim	512
nhead	4
num_encoder_layers	3
num_decoder_layers	3
dim_feedforward	1024
dropout	0.2
lr_initial	d_model$^{-0.5}$
l2	0.0001
layer_norm	0.00001
gradient_clip_val	10

Table 9: Transformer hyperparameters

E Error Analysis

Our best Transformer+VSE model predicts $1,119$ of $12,436$ evaluation expressions wrong meaning that the prediction does not match the reference perfectly. Here 425 errors (213 data, 77 ho-pos, 65 ho-color, 70 ho-uts) are expression predictions where the target piece is the one for which the board was initially designed for and 694 (all data) are cases where we picked an additional target randomly.

E.1 First-class errors

Error types	color	shape	pos	ungram.
data	4	5	180	24
ho-color	2	13	47	5
ho-pos	5	5	58	9
ho-uts	4	15	43	9

Table 10: The error types for the first-class errors

For the 425 first-class errors 213 of the errors are related to cases where the model mentions more properties of the target piece, although this would be unnecessary. In 47 cases the model produces an expressions that is not necessarily incorrect, but not grammatical.

E.2 Second-class errors

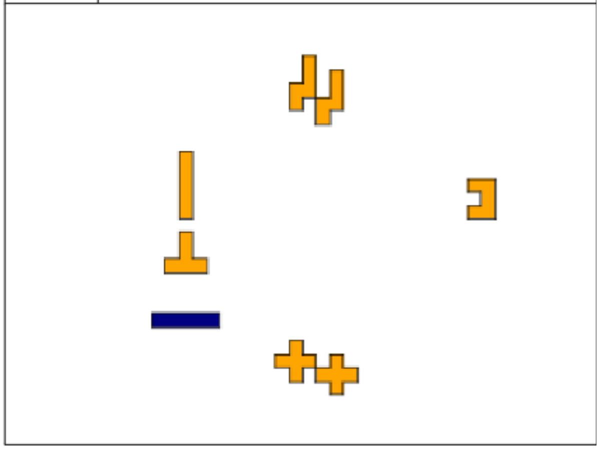
Error types	color	shape	pos	ungram.
data	30	56	524	90

Table 11: The error types for the second-class errors

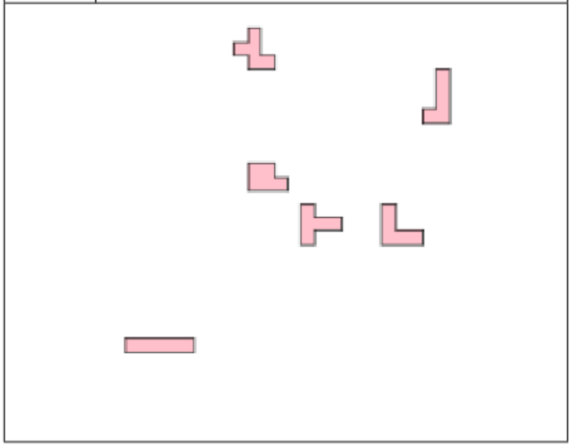
For the 694 second-class errors 179 of the errors are related to cases where the model mentions in addition the position, color or shape of the target piece, although this would be unnecessary. In 90 cases the model produces an expressions that is not necessarily incorrect, but not grammatical.

E.3 First-class error examples (intended target pieces)

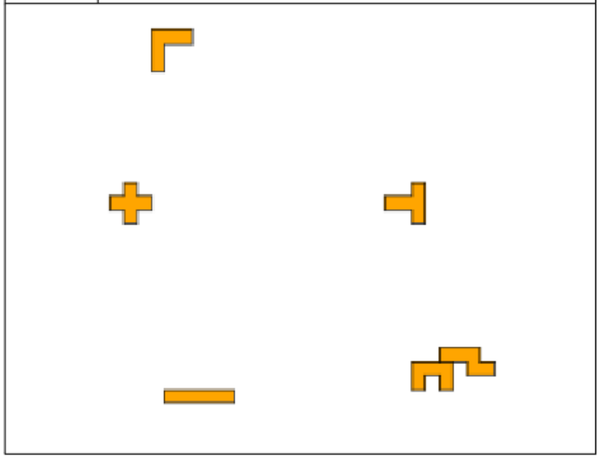
POSITION error	
Ref.	take the orange i
Pred.	take the orange i in the left center



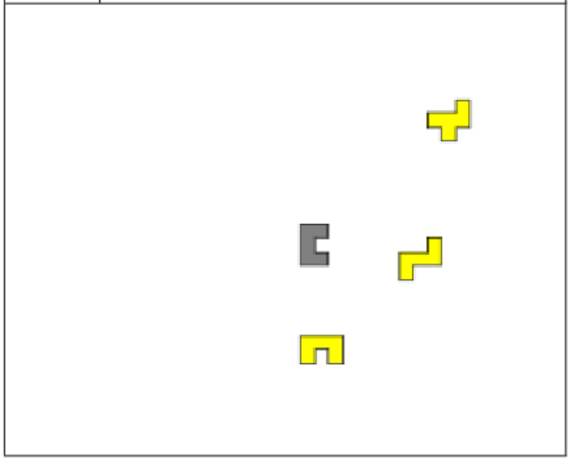
POSITION error	
Ref.	take the i
Pred.	take the i left center



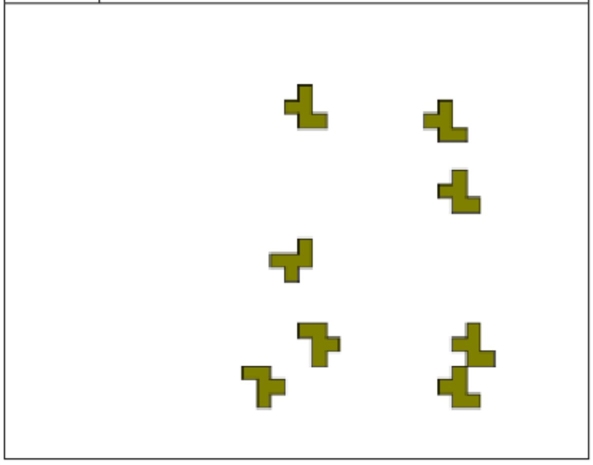
POSITION error	
Ref.	take the i
Pred.	take the i bottom in the bottom left



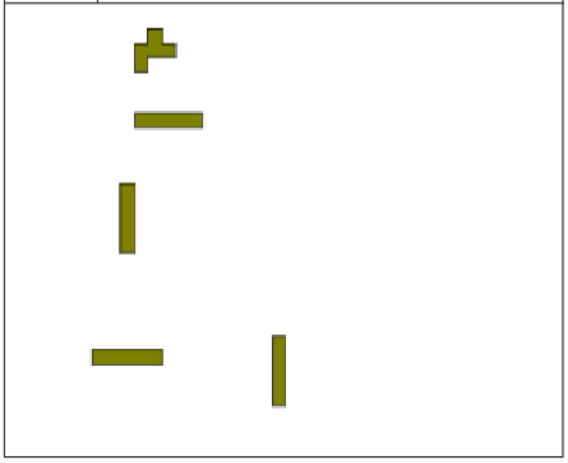
COLOR error	
Ref.	take the yellow u
Pred.	take the u



POSITION error	
Ref.	take the piece in the center
Pred.	take the piece in the bottom center

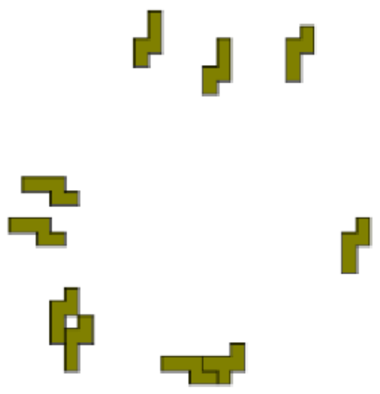


UNGRAM. error	
Ref.	take the i in the top left
Pred.	take the i top in the top left

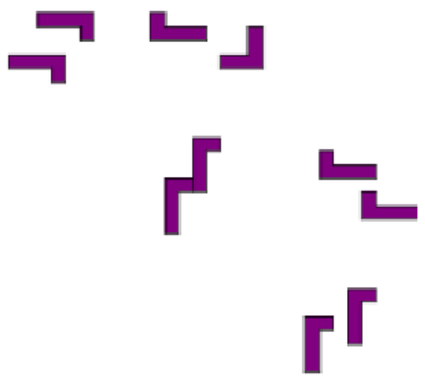


E.4 Second-class error examples (extra target pieces)

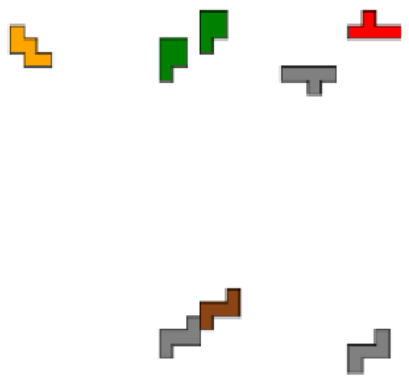
POSITION error	
Ref.	take the n in the right center
Pred.	take the n in the bottom right



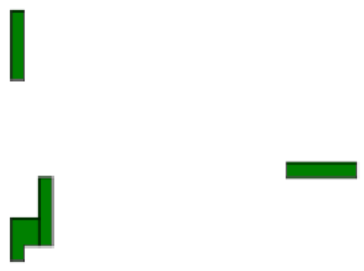
AMBIG. error	
Ref.	take the l in the bottom right
Pred.	take the piece in the bottom right



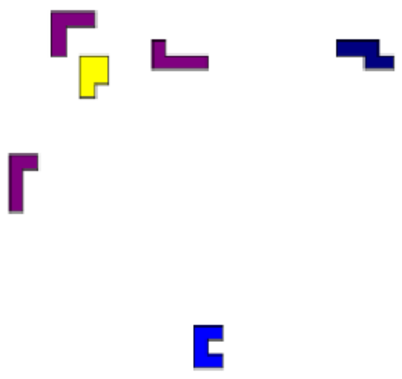
POSITION error	
Ref.	take the grey z in the bottom center
Pred.	take the grey z



UNGRAM. error	
Ref.	take the i in the bottom right
Pred.	take the i bottom right



SHAPE error	
Ref.	take the purple l in the left center
Pred.	take the purple piece in the left center



COLOR error	
Ref.	take the i in the bottom left
Pred.	take the i bottom in the bottom left