PIGLeT: Language Grounding Through Neuro-Symbolic Interaction in a 3D World

Rowan Zellers[♠] **Ari Holtzman**[♠] **Matthew Peters**[♡]
Roozbeh Mottaghi[♡] **Aniruddha Kembhavi**[♡] **Ali Farhadi**[♠] **Yejin Choi**[♠♡]
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♡]Allen Institute for Artificial Intelligence
https://rowanzellers.com/piglet

Abstract

We propose PIGLeT: a model that learns physical commonsense knowledge through interaction, and then uses this knowledge to ground language. We factorize PIGLeT into a physical dynamics model, and a separate language model. Our dynamics model learns not just what objects *are* but also what they *do*: glass cups break when thrown, plastic ones don't. We then use it as the interface to our language model, giving us a unified model of linguistic form and grounded meaning. PIGLeT can read a sentence, simulate neurally what might happen next, and then communicate that result through a literal symbolic representation, or natural language.

Experimental results show that our model effectively learns world dynamics, along with how to communicate them. It is able to correctly forecast "what happens next" given an English sentence over 80% of the time, outperforming a 100x larger, text-to-text approach by over 10%. Likewise, its natural language summaries of physical interactions are also judged by humans as more accurate than LM alternatives. We present comprehensive analysis showing room for future work.

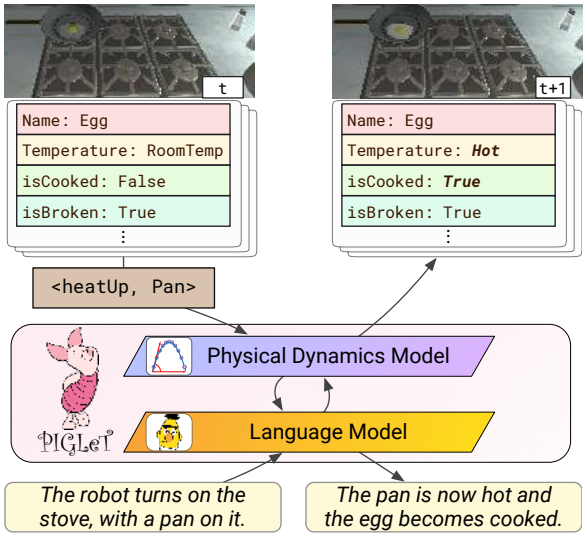

Figure 1: PIGLeT. Through physical interaction in a 3D world, we learn a model for what actions do to objects. We use our physical model as an interface for a language model, jointly modeling elements of language *form* and *meaning*. Given an action expressed symbolically or in English, PIGLeT can simulate what might happen next, expressing it symbolically or in English.

1 Introduction

As humans, our use of language is linked to the physical world. To process a sentence like "the robot turns on the stove, with a pan on it" (Figure 1) we might imagine a physical Pan object. This meaning representation in our heads can be seen as a part of our commonsense world knowledge, about what a Pan is and does. We might reasonably predict that the Pan will become Hot – and if there's an Egg on it, it would become cooked.

As humans, we learn such a commonsense world model through interaction. Young children learn to reason physically about basic objects by manipulating them: observing the properties they have, and how they change if an action is applied on them (Smith and Gasser, 2005). This process is hypothesized to be crucial to how children learn language: the names of these elementary objects become their first "real words" upon which other language is scaffolded (Yu and Smith, 2012).

In contrast, the dominant paradigm today is to train large language or vision models on *static data*, such as language and photos from the web. Yet such a setting is fundamentally limiting, as suggested empirically by psychologists' failed attempts to get kittens to learn passively (Held and Hein, 1963). More recently, though large Transformers have made initial progress on benchmarks, they also have frequently revealed biases in those same datasets, suggesting they might not be solving underlying tasks (Zellers et al., 2019b). This has been argued philosophically by a flurry of re-

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 2040–2050
August 1–6, 2021.

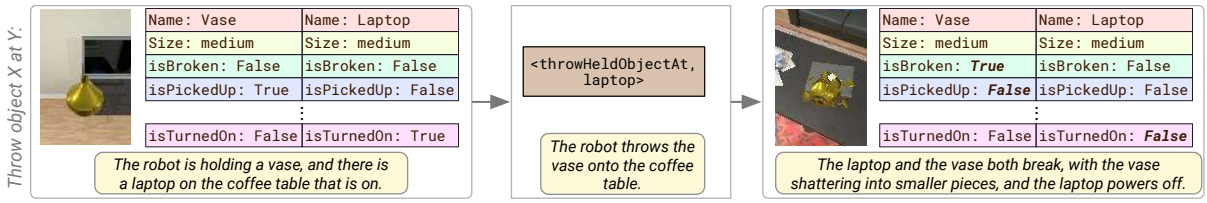

Figure 2: PIGPeN, a setting for few-shot language-world grounding. We collect data for 280k physical interactions in THOR, a 3D simulator with 20 actions and 125 object types, each with 42 attributes (e.g. `isBroken`). We annotate 2k interactions with English sentences describing the initial world state, the action, and the action result.

cent work arguing that no amount of language *form* could ever specify language *meaning* (McClelland et al., 2019; Bender and Koller, 2020; Bisk et al., 2020); connecting back to the Symbol Grounding Problem of Harnad (1990).

In this paper, we investigate an alternate strategy for learning physical commonsense through interaction, and then transferring that into language. We introduce a model named PIGLeT, short for **P**hysical **I**nteraction as **G**rounding for **L**anguag**e** **T**ransformers. We factorize an embodied agent into an explicit model of world dynamics, and a model of language form. We learn the dynamics model through *interaction*. Given an action `heatUp` applied to the `Pan` in Figure 1, the model learns that the `Egg` on the pan becomes `Hot` and `Cooked`, and that other attributes do not change.

We integrate our dynamics model with a pretrained language model, giving us a joint model of linguistic *form* and *meaning*. The combined PIGLeT can then reason about the physical dynamics implied by English sentences describing actions, predicting literally what might happen next. It can then communicate that result either symbolically or through natural language, generating a sentence like 'The egg becomes hot and cooked." Our separation between physical dynamics and language allows the model to learn about physical commonsense from the physical world itself, while also avoiding recurring problems of artifacts and biases that arise when we try to model physical world understanding solely through language.

We study this through a new environment and evaluation setup called PIGPeN, short for **P**hysical **I**nteraction **G**rounding **P**aire**d** with **N**atural Language. In PIGPeN, a model is given unlimited access to an environment for pretraining, but only 500 examples with paired English annotations. Models in our setup must additionally generalize to novel 'unseen' objects for which we intentionally do not provide paired language-environment supervision. We build this on top of the THOR environment (Kolve et al., 2017), a physics engine that enables agents to perform contextual interactions (Fig 2) on everyday objects.

Experiments confirm that PIGLeT performs well at grounding language with meaning. Given a sentence describing an action, our model predicts the resulting object states correctly over 80% of the time, outperforming even a 100x larger model (T5-11B) by over 10%. Likewise, its generated natural language is rated by humans as being more correct than equivalently-sized language models. Last, it can generalize in a 'zero-shot' way to objects that it has never read about before in language.

In summary, we make three key contributions. **First**, we introduce PIGLeT, a model decoupling physical and linguistic reasoning. **Second**, we introduce PIGPeN, to learn and evaluate the transfer of physical knowledge to the world of language. **Third**, we perform experiments and analysis suggesting promising avenues for future work.

2 PIGPeN: A Resource for Neuro-Symbolic Language Grounding

We introduce PIGPeN as a setting for learning and evaluating physically grounded language understanding. An overview is shown in Figure 2. The idea is that an agent gets access to an interactive 3D environment, where it can learn about the world through interaction – for example, that objects such as a `Vase` can become `Broken` if thrown. The goal for a model is to learn natural language *meaning* grounded in these interactions.

Task definition. Through interaction, an agent observes the interplay between objects $\boldsymbol{o} \in \mathcal{O}$ (represented by their attributes) and actions $\boldsymbol{a} \in \mathcal{A}$ through the following transition:

$$\underbrace{\{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_N\}}_{\vec{\boldsymbol{o}},\ \text{state pre-action}} \times \boldsymbol{a} \rightarrow \underbrace{\{\boldsymbol{o}'_1, \ldots, \boldsymbol{o}'_N\}}_{\vec{\boldsymbol{o}}',\ \text{state post-action}}. \quad (1)$$

Actions change the state of a subset of objects: turning on a `Faucet` affects a nearby `Sink`, but it will not change a `Mirror` on the wall.

To encourage learning from interaction, and not just language, an agent is given a small number of natural language annotations of transitions. We denote these sentences as $s_{\vec{o}}$, describing the state pre-action, s_a the action, and $s_{\vec{o}'}$ the state post-action respectively. During evaluation, an agent will sometimes encounter new objects o that were not part of the paired training data.

We evaluate the model's transfer in two ways:

a. PIGLeT-NLU. A model is given object states $\vec{o}$, and an English sentence s_a describing an action. It must predict the grounded object states $\vec{o}'$ that result after the action is taken.

b. PIGLeT-NLG. A model is given object states $\vec{o}$ and a literal action a. It must generate a sentence $s_{\vec{o}'}$ describing the state post-action.

We next describe our environment, feature representation, and language annotation process.

2.1 Environment: THOR

We use AI2-THOR as an environment for this task (Kolve et al., 2017). In THOR, a robotic agent can navigate around and perform rich contextual interactions with objects in a house. For instance, it can grab an `Apple`, slice it, put it in a `Fridge`, drop it, and so on. The state of the `Apple`, such as whether it is sliced or cold, changes accordingly; this is not possible in many other environments.

In this work, we use the underlying THOR simulator as a proxy for grounded meaning. Within THOR, it can be seen as a 'complete' meaning representation (Artzi et al., 2013), as it fully specifies the kind of grounding a model can expect in its perception within THOR.

Objects. The underlying THOR representation of each object o is in terms of 42 attributes; we provide a list in Appendix B. We treat these attributes as words specific to an attribute-level dictionary; for example, the temperature `Hot` is one of three possible values for an object's temperature; the others being `Cold` and `RoomTemp`.

Actions. An action a in THOR is a function that takes up to two objects as arguments. Actions are highly contextual, affecting not only the arguments but potentially other objects in the scene (Figure 2). We also treat action names as words in a dictionary.

Filtering out background objects. Most actions change the state of only a few objects, yet there can be many objects in a scene. We keep annotation and computation tractable by having models predict (and humans annotate) possible changes of at most two key objects in the scene. As knowing when an object *doesn't* change is also important, we include non-changing objects if fewer than two change.

Exploration. Any way of exploring the environment is valid for our task, however, we found that exploring *intentionally* was needed to yield good coverage of interesting states. Similar to prior work for instruction following (Shridhar et al., 2020), we designed an oracle to collect diverse and interesting trajectories $\{\vec{o}, a, \vec{o}'\}$. Our oracle randomly selects one of ten high level tasks, see Appendix B for the list. These in turn require randomly choosing objects in the scene; e.g. a `Vase` and a `Laptop` in Figure 2. We randomize the manner in which the oracle performs the task to discover diverse situations.

In total, we sampled 20k trajectories. From these we extracted 280k transitions (Eqn 1's) where at least one object changes state, for training.

2.2 Annotating Interactions with Language

2.2.1 Data Selection for Annotation

We select 2k action state-changes from trajectories held out from the training set. We select them while also balancing the distribution of action types to ensure broad coverage in the final dataset. We are also interested in a model's ability to generalize to new object categories – beyond what it has read about, or observed in a training set. We thus select 30 objects to be "unseen," and exclude these from paired environment-language training data. We sample 500 state transitions, containing only "seen" objects to be the training set; we use 500 for validation and 1000 for testing.

2.2.2 Natural Language Annotation

Workers on Mechanical Turk were shown an environment in THOR *before* and *after* a given action a. Each view contains the THOR attributes of the two key objects. Workers then wrote three English sentences, corresponding to $s_{\vec{o}}$, s_a, and $s_{\vec{o}'}$ respectively. Workers were instructed to write at a particular level of detail: enough so that a reader could infer "what happens next" from $s_{\vec{o}}$ and s_a, yet without mentioning redundant attributes.We provide more details in Appendix C.

3 Modeling PIGLeT

In this section, we describe our PIGLeT model. **First**, we learn a neural physical dynamics model

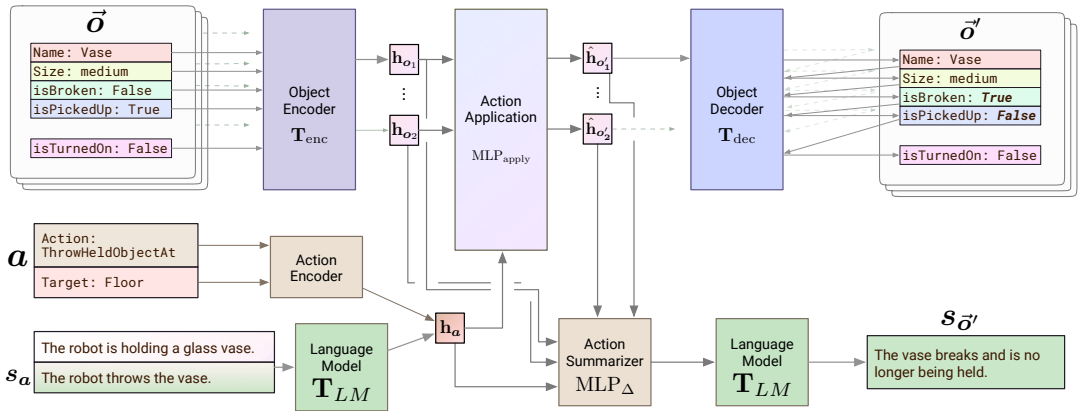

Figure 3: PIGLeT architecture. We pretrain a model of physical world dynamics by learning to transform objects $\vec{o}$ and actions a into new updated objects $\vec{o}'$. Our underlying world dynamics model – the encoder, the decoder, and the action application module, can augment a language model with grounded commonsense knowledge.

from interactions, and **second**, integrate with a pretrained model of language form.

3.1 Modeling Physical Dynamics

We take a neural, auto-encoder style approach to model world dynamics. An object o gets encoded as a vector $\mathbf{h}_o \in \mathbb{R}^{d_o}$. The model likewise encodes an action a as a vector $\mathbf{h}_a \in \mathbb{R}^{d_a}$, using it to manipulate the hidden states of all objects. The model can then decode any object hidden representation back into a symbolic form.

3.1.1 Object Encoder and Decoder

We use a Transformer (Vaswani et al., 2017) to encode objects into vectors $\mathbf{o} \in \mathbb{R}^{d_o}$, and then another to decode from this representation.

Encoder. Objects o are provided to the encoder as a set of attributes, with categories $c_1, ..., c_n$. Each attribute c has its own vocabulary and embedding $\mathbf{E}_c$. For each object o, we first embed all the attributes separately and feed the result into a Transformer encoder T_{enc}. This gives us (with position embeddings omitted for clarity):

$$\mathbf{h}_o = \mathbf{T}_{\text{enc}}\Big(\mathbf{E}_1(o_1), \ldots, \mathbf{E}_{c_n}(o_{c_n})\Big) \quad (2)$$

Decoder. We can then convert back into the original symbolic representation through a left-to-right Transformer decoder, which predicts attributes one-by-one from c_1 to c_n. This captures the inherent correlation between attributes, while making no independence assumptions, we discuss our ordering in Appendix A.2. The probability of predicting the next attribute $o_{c_{i+1}}$ is then given by:

$$p(o_{c_{i+1}}|\mathbf{h}_o, \boldsymbol{o}_{:c_i}) = \mathbf{T}_{\text{dec}}\Big(\mathbf{h}_o, \mathbf{E}_1(o_1), ..., \mathbf{E}_{c_i}(o_{c_i})\Big) \quad (3)$$

3.1.2 Modeling actions as functions

We treat actions a as functions that transform the state of all objects in the scene. Actions in our environment take at most two arguments, so we embed the action a and the names of its arguments, concatenate them, and pass the result through a multilayer perceptron; yielding a vector representation $\mathbf{h}_a$.

Applying Actions. We use the encoded action $\mathbf{h}_a$ to transform all objects in the scene, obtaining updated representations $\hat{\mathbf{h}}_{o'}$ for each one. We take a *global* approach, jointly transforming all objects. This takes into account that interactions are contextual: turning on a `Faucet` might fill up a `Cup` if and only if there is one beneath it.

Letting the observed objects in the interaction be o_1 and o_2, with encodings $\mathbf{h}_{o_1}$ and $\mathbf{h}_{o_2}$ respectively, we model the transformation via the following multilayer perceptron:

$$[\hat{\mathbf{h}}_{o'_1}, \hat{\mathbf{h}}_{o'_2}] = \text{MLP}_{\text{apply}}\Big([\mathbf{h}_a, \mathbf{h}_{o_1}, \mathbf{h}_{o_2}]\Big). \quad (4)$$

The result can be decoded into symbolic form using the object decoder (Equation 3).

3.1.3 Loss function and training

We train our dynamics model on $(\vec{o}, a, \vec{o}')$ transitions. The model primarily learns by running $\vec{o}, a$ through the model, predicting the updated output state $\hat{\mathbf{h}}_{o'}$, and minimizing the cross-entropy of generating attributes of the real changed object $\vec{o}'$. We also regularize the model by encoding objects $\vec{o}, \vec{o}'$ and having the model learn to reconstruct them. We weight all these cross-entropy losses equally. We discuss our architecture in Appendix A.1; it uses 3-layer Transformers, totalling 17M parameters.

3.2 Language Grounding

After pretraining our physical dynamics model, we integrate it with a Transformer Language Model (LM). In our framework, the role of the LM will be to both encode natural language sentences of actions into a hidden state approximating $\mathbf{h}_{\boldsymbol{a}}$, as well as summarizing the result of an interaction ($\vec{\boldsymbol{o}}$,$\boldsymbol{a}$,$\vec{\boldsymbol{o}}'$) in natural language.

Choice of LM. Our framework is compatible with any language model. However, to explore the impact of pretraining data on grounding later in this paper, we pretrain our own with an identical architecture to the smallest GPT2 (Radford et al. (2019); 117M). To handle both classification and generation well, we mask only part of the attention weights out, allowing the model to encode a "prefix" bidirectionally; it generates subsequent tokens left-to-right (Dong et al., 2019). We pretrain the model on Wikipedia and books; details in Appendix D.

We next discuss architectural details of performing the language transfer, along with optimization.

3.2.1 Transfer Architecture

English actions to vector form. Given a natural language description $\boldsymbol{s}_{\boldsymbol{a}}$ of an action $\boldsymbol{a}$, like "The robot throws the vase," for PIGLeT-NLU, our model will learn to parse this sentence into a neural representation $\mathbf{h}_{\boldsymbol{a}}$, so the dynamics model can simulate the result. We do this by encoding $\boldsymbol{s}_{\boldsymbol{a}}$ through our language model, $\mathbf{T}_{LM}$, with a learned linear transformation over the resulting (bidirectional) encoding. The resulting vector $\mathbf{h}_{\boldsymbol{s}_{\boldsymbol{a}}}$ can then be used by Equation 4.

Summarizing the result of an action. For PIGLeT-NLG, our model simulates the result of an action $\boldsymbol{a}$ neurally, resulting in a predicted hidden state $\hat{\mathbf{h}}_{\boldsymbol{o}}$ for each object in the scene $\boldsymbol{o}$. To write an English summary describing "what changed," we first learn a lightweight fused representation of the transition, aggregating the initial and final states, along with the action, through a multilayer perceptron. For each object $\boldsymbol{o}_i$ we have:

$$\mathbf{h}_{\Delta \boldsymbol{o}_i} = \text{MLP}_{\Delta}([\mathbf{h}_{\boldsymbol{o}_i}, \hat{\mathbf{h}}_{\boldsymbol{o}_i'}, \mathbf{h}_{\boldsymbol{a}}]). \quad (5)$$

We then use the sequence $[\mathbf{h}_{\Delta \boldsymbol{o}_1}, \mathbf{h}_{\Delta \boldsymbol{o}_2}]$ as bidirectional context for our our LM to decode from. Additionally, since our test set includes novel objects not seen in training, we provide the names of the objects as additional context for the LM generator (e.g. 'Vase, Laptop'); this allows a LM to copy those names over rather than hallucinate wrong ones. Importantly we only provide the surface-form names, **not** underlying information about these objects or their usage as with few-shot scenarios in the recent GPT-3 experiments (Brown et al., 2020) – necessitating that PIGLeT learns what these names *mean* through interaction.

3.2.2 Loss functions and training.

Modeling text generation allows us to incorporate a new loss function, that of minimizing the log-likelihood of generating each $\boldsymbol{s}_{\vec{\boldsymbol{o}}'}$ given previous words and the result of Equation 5:

$$p(s_{i+1}^{\text{post}} | \boldsymbol{s}_{\vec{\boldsymbol{o}}',1:i}) = \mathbf{T}_{\text{LM}}(\mathbf{h}_{\Delta \boldsymbol{o}_1}, \mathbf{h}_{\Delta \boldsymbol{o}_2}, \boldsymbol{s}_{\vec{\boldsymbol{o}}',1:i}). \quad (6)$$

We do the same for the object states $\boldsymbol{s}_{\vec{\boldsymbol{o}}}$ pre-action, using $\mathbf{h}_{\boldsymbol{o}_i}$ as the corresponding hidden states.

For PIGLeT-NLU, where no generation is needed, optimizing Equation 5 is not strictly necessary. However, as we will show later, it helps provide additional signal to the model, improving overall accuracy by several percentage points.

4 Experiments

We test our model's ability to encode language into a grounded form (PIGLeT-NLU), and decode that grounded form into language (PIGLeT-NLG).

4.1 PIGLeT-NLU Results.

We first evaluate models by their performance on PIGLeT-NLU: given objects $\vec{\boldsymbol{o}}$, and a sentence $\boldsymbol{s}_{\boldsymbol{a}}$ describing an action, a model must predict the resulting state of objects $\vec{\boldsymbol{o}}'$. We primarily evaluate models by accuracy; scoring how many objects for which they got all attributes correct. We compare with the following strong baselines:

a. No Change: this baseline copies the initial state of all objects $\vec{\boldsymbol{o}}$ as the final state $\vec{\boldsymbol{o}}'$.

b. GPT3-175B (Brown et al., 2020), a very large language model for 'few-shot' learning using a prompt. For GPT3, and other text-to-text models, we encode and decode the symbolic object states in a JSON-style dictionary format, discussed in Appendix A.4.

c. T5 (Raffel et al., 2019). With this model, we use the same 'text-to-text' format, however here we train it on the paired data from PIGLeT. We consider varying sizes of T5, from T5-Small – the closest in size to PIGLeT, up until T5-11B, roughly 100x the size.

d. (Alberti et al., 2019)-style. This paper originally proposed a model for VCR (Zellers et al.,

	Model	Accuracy (%)				Attribute-level accuracy (Test-Overall,%)				
		Val	Test			size	distance	mass	Temperature	isBroken
			Overall	Seen	Unseen	8-way	8-way	8-way	3-way	boolean
	No Change	27.4	25.5	29.9	24.0	83.2	84.1	96.3	86.0	94.8
text-to-text	GPT3-175B (Brown et al., 2020)	23.8	22.4	22.4	21.4	73.7	77.0	89.5	84.2	94.7
text-to-text	T5-11B (Raffel et al., 2019)	68.5	64.2	79.5	59.1	83.9	88.9	94.3	95.4	98.1
text-to-text	T5-3B	66.6	63.3	77.1	58.7	81.6	90.0	94.0	95.6	98.4
text-to-text	T5-Large	56.5	54.1	69.2	49.1	81.8	84.6	94.3	96.3	95.8
text-to-text	T5-Base	56.0	53.9	69.2	48.8	81.1	87.5	93.6	96.1	96.5
text-to-text	T5-Small	39.9	36.2	57.0	38.0	82.2	84.9	93.8	89.6	93.5
BERT style	Alberti et al.2019, Pretrained Dynamics	61.3	53.9	71.4	48.1	87.7	87.6	97.5	93.4	97.5
BERT style	Alberti et al.2019	9.7	6.8	16.2	3.7	53.4	43.6	84.0	88.1	95.1
BERT style	G&D2019, Pretrained Dynamics	43.8	35.3	60.9	26.9	83.0	86.9	94.0	93.7	97.4
BERT style	G&D2019	15.1	11.3	23.1	7.3	68.6	47.3	82.2	88.3	95.8
	PIGLeT	**81.8**	**81.1**	**83.8**	**80.2**	**92.3**	**91.9**	**99.2**	**99.8**	**99.0**

Table 1: **Overall results**. **Left**: we show the model accuracies at predicting all attributes of an object correctly. We compare PIGLeT with 'text-to-text' approaches that represent the object states as a string, along with BERT-style approaches with additional machinery to encode inputs or decode outputs. PIGLeT outperforms a T5 model 100x its size (11B params) and shows gains over the BERT-style models that also model action dynamics through a language transformer. **Right**: we show several attribute-level accuracies, along with the number of categories per attribute; PIGLeT outperforms baselines by over 4 points for some attributes such as size and distance.

2019a), where grounded visual information is fed into a BERT model as tokens; the transformer performs the grounded reasoning. We adapt it for our task by using our base LM and feeding in object representations from our pretrained object encoder, also as tokens. Our object decoder predicts the object, given the LM's pooled hidden state. This is "pretrained dynamics," we also consider a version without a randomly initialized dynamics model.

e. (Gupta and Durrett, 2019)-style. Thiso paper proposes using Transformers to model physical state, for tasks like entity tracking in recipes. Here, the authors propose decoding a physical state attribute (like `isCooked`) by feeding the model a label-specific `[CLS]` token, and then mapping the result through a hidden layer. We do this and use a similar object encoder as our (Alberti et al., 2019)-style baseline.

We discuss hyperparameters in Appendix A.3.

Results. From the results (Table 1), we can draw several patterns. Our model, PIGLeT performs best at getting all attributes correct; doing so over 80% on both validation and test sets, even for novel objects not seen during training. The next closest model is T5-11B, which scores 68% on validation. Though when evaluated on objects 'seen' during training it gets 77%, that number drops by over 18% for unseen objects. On the other hand, PIGLeT has a modest gap of 3%. This suggests that our approach is particularly effective at connecting unpaired language and world representations. At the other extreme, GPT3 does poorly in its 'few-shot' setting, suggesting that size is no replacement for grounded supervision.

PIGLeT also outperforms 'BERT style' approaches that control for the same language model architecture, but perform the physical reasoning inside the language transformer rather than as a separate model. Performance drops when the physical decoder must be learned from few paired examples (as in Gupta and Durrett (2019)); it drops even further when neither model is given access to our pretrained dynamics model, with both baselines then underperforming 'No Change.' This suggests that our approach of having a physical reasoning model *outside of* an LM is a good inductive bias.

Model	Accuracy (val;%)
PIGLeT, No Pretraining	10.4
PIGLeT, Non-global $\text{MLP}_{\text{apply}}$	72.0
PIGLeT, Global $\text{MLP}_{\text{apply}}$	78.5
PIGLeT, Global $\text{MLP}_{\text{apply}}$, Gen. loss (6)	81.8
PIGLeT, Symbols Only (Upper Bound)	89.3

Table 2: Ablation study on PIGPeN-NLU's validation set. Our model improves 6% by modeling global dynamics of all objects in the scene, versus applying actions to single objects in isolation. We improve another 3% by adding an auxiliary generation loss.

4.1.1 Ablation study

In Table 2 we present an ablation study of PIGLeT's components. Of note, by using a global representation of objects in the world (Equation 4), we get

over 6% improvement over a local representation where objects are manipulated independently. We get another 3% boost by adding a generation loss, suggesting that learning to generate summaries helps the model better connect the world to language. Last, we benchmark how much headroom there is on PIGPeN-NLU by evaluating model performance on a 'symbols only' version of the task, where the symbolic action $\boldsymbol{a}$ is given explicitly to our dynamics model. This upper bound is roughly 7% higher than PIGLeT, suggesting space for future work.

4.2 PIGPeN-NLG Results

Next, we turn to PIGPeN-NLG: given objects $\vec{\boldsymbol{o}}$, and the literal next action $\boldsymbol{a}$, a model must generate a sentence $\boldsymbol{s}_{\vec{\boldsymbol{o}}'}$ describing what will change in the scene. We compare with the following baselines:

a. T5. We use a T5 model that is given a JSON-style dictionary representation of both $\vec{\boldsymbol{o}}$ and $\boldsymbol{a}$, it is finetuned to generate summaries $\boldsymbol{s}_{\vec{\boldsymbol{o}}'}$.

b. LM Baseline. We feed our LM hidden states $\mathbf{h}_{\boldsymbol{o}}$ from our pretrained encoder, along with its representation of $\boldsymbol{a}$. The key difference between it and PIGLeT is that we do **not** allow it to simulate neurally what might happen next – $\text{MLP}_{\text{apply}}$ is never used here.

Size matters. Arguably the most important factor controlling the fluency of a language generator is its size (Kaplan et al., 2020). Since our LM could also be scaled up to arbitrary size, we control for size in our experiments and only consider models the size of GPT2-base (117M) or smaller; we thus compare against T5-small as T5-Base has 220M parameters. We discuss optimization and sampling hyperparameters in Appendix A.3.

Evaluation metrics. We evaluate models over the validation and test sets. We consider three main evaluation metrics: BLEU (Papineni et al., 2002) with two references, the recently proposed BERTScore (Zhang et al., 2020), and conduct a human evaluation. Humans rate both the fluency of post-action text, as well as its faithfulness to true action result, on a scale from -1 to 1.

Results. We show our results in Table 3. Of note, PIGLeT is competitive with T5 and significantly outperforms the pure LM baseline, which uses a pretrained encoder for object states, yet has the physical simulation piece $\text{MLP}_{\text{apply}}$ removed. This suggests that simulating world dynamics not only allows the model to predict what might happen next, it leads to more faithful generation as well.

Model	BLEU		BERTScore		Human (test; [-1, 1])	
	Val	Test	Val	Test	Fluency	Faithfulness
T5	46.6	43.4	82.2	81.0	0.82	0.15
LM Baseline	44.6	39.7	81.6	78.8	0.91	-0.13
PIGLeT	**49.0**	**43.9**	**83.6**	**81.3**	**0.92**	**0.22**
Human	44.5	45.6	82.6	83.3	0.94	0.71

Table 3: Text generation results on PIGPeN-NLG, showing models of roughly equivalent size (up to 117M parameters). Our PIGLeT outperforms the LM baseline (using the same architecture but omitting the physical reasoning component) by 4 BLEU points, 2 BERTScore F_1 points, and 0.35 points in a human evaluation of language faithfulness to the actual scene.

5 Analysis

5.1 Qualitative examples.

We show two qualitative examples in Figure 4, covering both PIGPeN-NLU as well as PIGPeN-NLG. In the first row, the robot empties a held `Mug` that is filled with water. PIGLeT gets the state, and generates a faithful sentence summarizing that the mug becomes empty. T5 struggles somewhat, emptying the water from both the `Mug` and the (irrelevant) `Sink`. It also generates text saying that the Sink becomes empty, instead of the Mug.

In the second row, PIGLeT correctly predicts the next object states, but its generated text is incomplete – it should also write that the mug becomes filled wtih Coffee. T5 makes the same mistake in generation, and it also underpredicts the state changes, omitting all changes to the `Mug`.

We suspect that T5 struggles here in part because `Mug` is an unseen object. T5 only experiences it through language-only pretraining, but this might not be enough for a fully grounded representation.

5.2 Representing novel words

The language models that perform best today are trained on massive datasets of text. However, this has unintended consequences (Bender et al., 2021) and it is unlike how children learn language, with children learning novel words from experience (Carey and Bartlett, 1978). The large scale of our pretraining datasets might allow models to learn to perform physical-commonsense like tasks for wrong reasons, overfitting to surface patterns rather than learning meaningful grounding.

We investigate the extent of this by training a 'zero-shot' version of our backbone LM on Wikipedia and books – the only difference is that

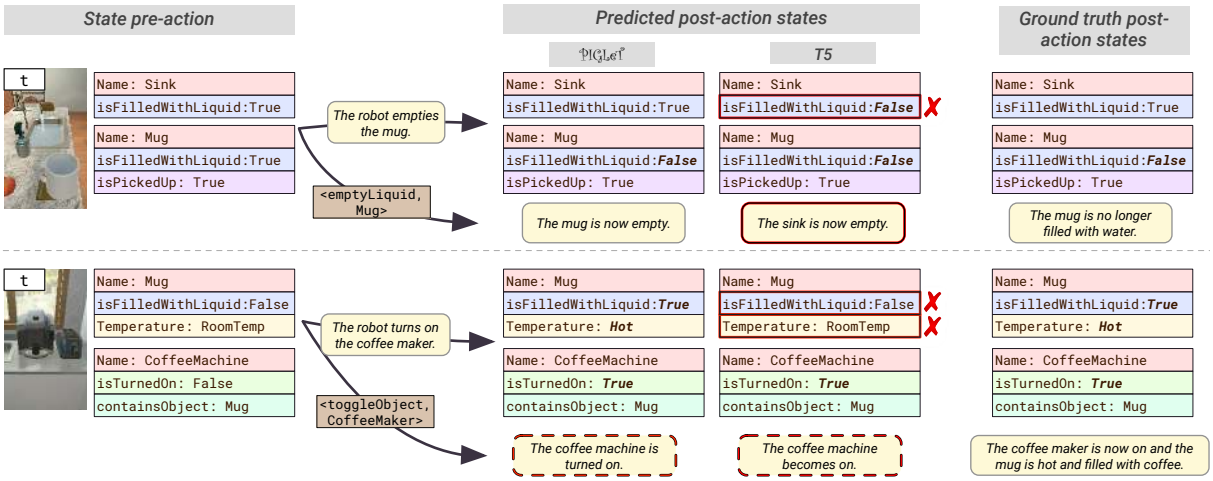

Figure 4: Qualitative examples. Our model PIGLeT reliably predicts what might happen next (like the Mug becoming empty in Row 1), in a structured and explicit way. However, it often struggles at generating sentences for unseen objects like Mug that are excluded from the training set. T5 struggles to predict these changes, for example, it seems to suggest that emptying the Mug causes all containers in the scene to become empty.

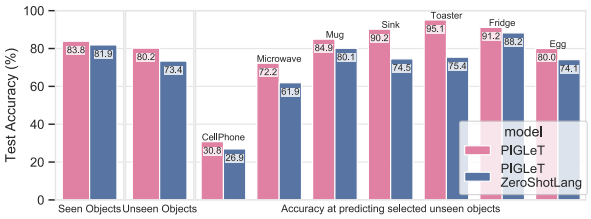

Figure 5: PIGPeN-NLU performance of a **zero-shot** PIGLeT, that was pretrained on Books and Wikipedia without reading any words of our 'unseen' objects like 'mug.' It outperforms a much bigger T5-11B overall, though is in turn beaten by PIGLeT on unseen objects like 'Sink' and 'Microwave.'

we explicitly **exclude** all mentioned sentences containing one of our "unseen" object categories. In this setting, not only must PIGLeT learn to ground words like 'mug,' it must do so without having seen the word 'mug' during pretraining. This is significant because we count over 20k instances of 'Mug' words (including morphology) in our dataset.

We show results in Figure 5. A version of PIGLeT with the zero-shot LM does surprisingly well – achieving 80% accuracy at predicting the state changes for "Mug" – despite never having been pretrained on one before. This even outperforms T5 at the overall task. Nevertheless, PIGLeT outperforms it by roughly 7% at unseen objects, with notable gains of over 10% on highly dynamic objects like Toasters and Sinks.

6 Related Work

Grounded commonsense reasoning. In this work, we study language grounding and commonsense reasoning at the representation and concept level. The aim is to train models that learn to acquire concepts more like humans, rather than performing well on a downstream task that (for humans) requires commonsense reasoning. Thus, this work is somewhat different versus other 3D embodied tasks like QA (Gordon et al., 2018; Das et al., 2018), along with past work for measuring such grounded commonsense reasoning, like SWAG, HellaSWAG, and VCR (Zellers et al., 2018, 2019b,a). The knowledge covered is different, as it is self-contained within THOR. While VCR, for instance, includes lots of visual situations about what people are doing, this paper focuses on learning the physical properties of objects.

Zero-shot generalization. There has been a lot of past work involved with learning 'zero-shot': often learning about the grounded world in language, and transferring that knowledge to vision. Techniques for this include looking at word embeddings (Frome et al., 2013) and dictionary definitions (Zellers and Choi, 2017). In this work, we propose the inverse. This approach was used to learn better word embeddings (Gupta et al., 2019) or semantic tuples (Yatskar et al., 2016), but we consider learning a component to be plugged into a deep Transformer language model.

Past work evaluating these types of zero-shot generalization have also looked into how well models can compose concepts in language together (Lake and Baroni, 2018; Ruis et al., 2020). Our work considers elements of compositionality through grounded transfer. For example, in

PIGPeN-NLG, models must generate sentences about the equivalent of dropping a 'dax', despite never having seen one before. However, our work is also contextual, in that the outcome of 'dropping a dax' might depend on external attributes (like how high we're dropping it from).

Structured Models for Attributes and Objects. The idea of modeling actions as functions that transform objects has been explored in the computer vision space (Wang et al., 2016). Past work has also built formal structured models for connecting vision and language (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013), we take a neural approach and connect today's best models of language *form* to similarly neural models of a simulated environment.

7 Conclusion

In this paper, we presented an approach PIGLeT for jointly modeling language form and meaning. We presented a testbed PIGPeN for evaluating our model, which performs well at grounding language to the (simulated) world.

Acknowledgments

We thank the reviewers for their helpful feedback, and the Mechanical Turk workers for doing a great job in annotating our data. Thanks also to Zak Stone and the Google Cloud TPU team for help with the computing infrastructure. This work was supported by the DARPA CwC program through ARO (W911NF-15-1-0543), the DARPA MCS program through NIWC Pacific (N66001-19-2-4031), and the Allen Institute for AI.

References

Chris Alberti, Jeffrey Ling, Michael Collins, and David Reitter. 2019. Fusion of detected objects in text for visual question answering. 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 2131–2140.

Yoav Artzi, Nicholas FitzGerald, and Luke S Zettlemoyer. 2013. Semantic parsing with combinatory categorial grammars. *ACL (Tutorial Abstracts)*, 3.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big. *Proceedings of FAccT*.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, et al. 2020. Experience grounds language. *arXiv preprint arXiv:2004.10151*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

S. Carey and E. Bartlett. 1978. Acquiring a single new word.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Aditya Gupta and Greg Durrett. 2019. Effective use of transformer networks for entity tracking. 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 759–769.

Tanmay Gupta, Alexander Schwing, and Derek Hoiem. 2019. Vico: Word embeddings from visual co-occurrences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7425–7434.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

Richard Held and Alan Hein. 1963. Movement-produced stimulation in the development of visually guided behavior. *Journal of comparative and physiological psychology*, 56(5):872.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1435–1442.

James L McClelland, Felix Hill, Maja Rudolph, Jason Baldridge, and Hinrich Schütze. 2019. Extending machine language models toward human-level language understanding. *arXiv preprint arXiv:1912.05877*.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. 2020. A benchmark for systematic generalization in grounded language understanding. *Advances in Neural Information Processing Systems*, 33.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4603–4611.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10740–10749.

Linda Smith and Michael Gasser. 2005. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010. Curran Associates Inc.

Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. 2016. Actions ˜ transformations. In *CVPR*.

Mark Yatskar, Vicente Ordonez, and Ali Farhadi. 2016. Stating the obvious: Extracting visual common sense knowledge. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 193–198.

Chen Yu and Linda B Smith. 2012. Embodied attention and word learning by toddlers. *Cognition*, 125(2):244–262.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019a. From recognition to cognition: Visual commonsense reasoning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical*

Methods in Natural Language Processing, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

Rowan Zellers and Yejin Choi. 2017. Zero-shot activity recognition with verb attribute induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 946–958.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019b. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Tianyi Zhang, V. Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *ArXiv*, abs/1904.09675.