Learning the Effects of Physical Actions in a Multi-modal Environment

Gautier Dagan, Frank Keller, Alex Lascarides
School of Informatics
University of Edinburgh, UK
gautier.dagan@ed.ac.uk, {keller, alex}@inf.ed.ac.uk

Abstract

Large Language Models (LLMs) handle physical commonsense information inadequately. As a result of being trained in a disembodied setting, LLMs often fail to predict an action's outcome in a given environment. However, predicting the effects of an action before it is executed is crucial in planning, where coherent sequences of actions are often needed to achieve a goal. Therefore, we introduce the multi-modal task of predicting the outcomes of actions solely from realistic sensory inputs (images and text). Next, we extend an LLM to model latent representations of objects to better predict action outcomes in an environment. We show that multi-modal models can capture physical commonsense when augmented with visual information. Finally, we evaluate our model's performance on novel actions and objects and find that combining modalities help models to generalize and learn physical commonsense reasoning better.

1 Introduction

Large Language Models (LLMs) are trained on large corpora of disembodied texts. They are typically pre-trained on a masked language modeling task: the model must predict a masked word in a text given its context. LLMs have achieved state-of-the-art performance on many NLP tasks (Devlin et al., 2019; Brown et al., 2020), but they can also fail on seemingly easy and obvious tasks and in unpredictable ways (McCoy et al., 2020; Bommasani et al., 2021). Commonsense knowledge is shared knowledge and is often so obvious that it is absent from the LLMs' training data: people don't mention what is already known to their interlocutors. This includes physical commonsense information, including how executed actions affect the physical attributes of objects; e.g., shape and weight (Forbes et al., 2019). Humans may learn such knowledge from their embodied environment. But LLMs, being trained on disembodied text, can make incorrect predictions about physical attributes and how these change when actions occur. For instance, when asked what the weight of a 150 grams potato after it is sliced, GPT-3 (Brown et al., 2020) incorrectly answers 75 grams (see Appendix A for the exact prompt). GPT-3 is an LLM with 175 billion parameters, and nonetheless its disembodied existence limits its physical commonsense estimates.

Zellers et al. (2021) inject physical commonsense information into LLMs via their model PIGLeT—a modified LLM that is trained on their PIGPeN simulated 3D environment dataset. PIGLeT estimates how an environment changes as a result of specific actions. In training and testing, the model uses ground-truth symbolic representations of the environment but not the images: it ignores visual sensory observations. These symbolic representations of objects in an environment are chosen to capture the possible effects of actions, and include attributes like `weight`, `size` and `temperature`. However, in an embodied situation, an agent needs to use visual perception to estimate its interpretation of the scene. Therefore, the symbolic representations should be treated as latent rather than observed.

We propose an alternative to the PIGLeT model, PIGLeT-Vis, which uses images directly as input into a multi-modal LLM to ground the model to its physical environment. We compare our approach to the original PIGLeT model and evaluate the generalization capabilities gained from using image inputs. At test time, our model foregoes symbolic labels: only the images and the name of the action are observed. Thus our model tackles a more challenging task than the original PIGLeT model in that it must not only predict the effect of actions but also (indirectly) estimate the symbolic representations of objects in the images. We also evaluate a model for predicting the effects of actions that trains on PIGPeN's images and their associated natural language (NL) descriptions, eliminating the need for

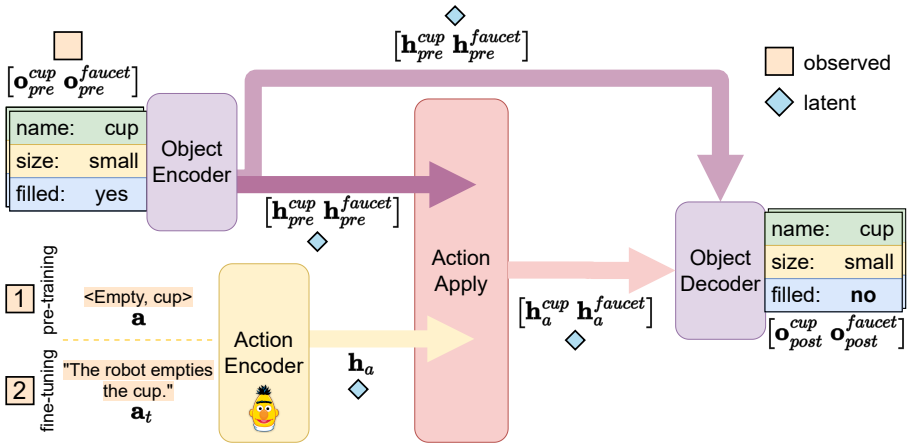


Figure 1: **Original PIGLeT Physical Dynamics Model (Zellers et al., 2021).** During pre-training the model receives as input the full symbolic representation of two objects ($\mathbf{o}^0_{pre}$ and $\mathbf{o}^1_{pre}$) before the action is taken and the symbolic representation of the action itself ($\mathbf{a}$) and is tasked with predicting the attributes of the objects after the action ($\mathbf{o}^0_{post}$ and $\mathbf{o}^1_{post}$). During fine-tuning, the action encoder is replaced by an LLM to process a natural language description of the action being taken and with what objects.

formal symbolic representations.

Our contributions are three-fold. First, we show that it is possible to predict the physical effects of actions from visual data. Second, we show that it is possible to learn the task on training data where formal symbolic representations, which are unobservable in real-world settings, are replaced with NL descriptions (which can be observed through natural interaction). Third, we evaluate all our models in a stricter zero-shot setup to promote ways to train agents that generalize. Overall our work paves the way for multi-modal models that learn the effects of actions in realistic environments.

2 Related Work

Commonsense reasoning has been highlighted as a potential weak point of LLMs in recent years (Shen and Kejriwal, 2021; Forbes et al., 2019; Bisk et al., 2020). Datasets such as PIGPeN (Zellers et al., 2021), commonsenseQA (Talmor et al., 2019), VCR (Zellers et al., 2019) and GD-VCR (Yin et al., 2021) help evaluate different aspects of commonsense reasoning in modern LLMs. In this paper, we focus on physical commonsense reasoning, which involves understanding the (often) unexpressed rules of the physical world.

Forbes et al. (2019) reported that neural representations found it challenging to infer the link between actions and what they imply about the attributes of objects. Accordingly, Zellers et al. (2019) introduced the Visual Commonsense Reasoning (VCR) task to test how images can inform question answering models that tackle commonsense information. Bisk et al. (2020) designed the PIQA benchmark to evaluate physical commonsense reasoning in LLMs through question answering. Sampat et al. (2021) proposed an extension to the CLEVR dataset, where an agent must reason and answer questions about a scene after a hypothetical action is taken.

Multiple approaches can improve the capabilities of LLMs in commonsense reasoning, such as using handcrafted knowledge graphs (Hwang et al., 2021) or leveraging simulated environments (Zellers et al., 2021). PIGLeT, in particular, combines a traditional LLM and a "Physical Dynamics" model to ground an LLM (Zellers et al., 2021). The Physical Dynamics model enhances the commonsense knowledge of an LLM by fine-tuning it, using trajectories sampled from a realistic environment (see Figure 1). Trajectories are an action and a pair of environment states (before and after the action) expressed in a formal symbolic representation. Zellers et al. (2021) found that fine-tuning LLMs with symbolic data from the simulated environment helped them outperform other models in physical commonsense reasoning tasks: in particular, predicting the effects of an action when executed in a particular state.

Image inputs offer a way to ground an LLM, as they only require general alignment with a text or symbolic input and do not require the comprehensive environment ground-truth labels that PIGLeT uses. Gao et al. (2018) used multi-modal web data to learn actions and their effects from images

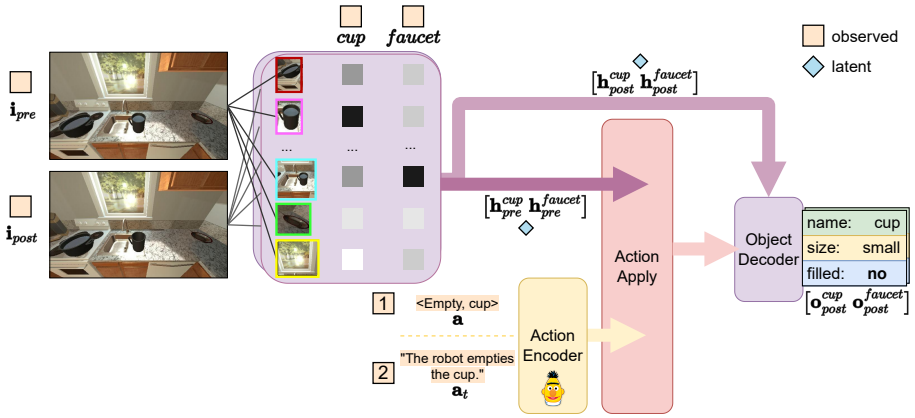


Figure 2: **PIGLeT-Vis.** We introduce PIGLeT-Vis, where we modify the PIGLeT architecture to replace its Symbolic Object Encoder with a vision component that makes use of images of the environment before and after an action is taken to predict the symbolic representation of objects post-action. We use an attention mechanism over the extracted bounding boxes to obtain a visual hidden representation of an object given its name. The only remaining symbolic inputs during pre-training are the action description and object names.

and corresponding text descriptions. Zellers et al. (2019) used an off-the-shelf ResNet50 model (He et al., 2016) to augment an existing BERT language model (Devlin et al., 2019) with vision capabilities. Transformer models such as UNITER (Chen et al., 2020), ERNIE-ViL (Yu et al., 2021), VisualBERT (Li et al., 2020), and ViLBert (Lu et al., 2019) have been applied to visual commonsense reasoning. These models use a joint transformer backbone for images and text and vary their pre-training objectives. However, most of these models are trained on static text-image pairs: they aren't designed to capture the dynamics of an environment, particularly how object attributes change with actions. Notably, recent work by Hanna et al. (2022) uses CLIP (Radford et al., 2021) and MOCA (Singh et al., 2021) embeddings to predict a post-action image given a set of possible images. In contrast, we focus on adapting an LLM with a vision-based component to predict the consequences of actions on the environment.

3 Method

We propose PIGLeT-Vis (Figure 2) for learning the effects of actions on objects from images. We use a pre-trained vision backbone, DETR (Carion et al., 2020), as a Vision Object Encoder and combine it with a RoBERTa LLM (Liu et al., 2019) as an Action Encoder. We experiment with different configurations of inputs to measure the impact of the various components of our architecture. In particular, we test a variation in which we remove the formal symbolic labels even in training, replacing them with NL text labels. To evaluate our models, we use the PIGPeN dataset (Zellers et al., 2021), which consists of a symbolic and visual representation of an environment before and after an action is taken. However, we filter PIGPeN to create a viable testing ground for visual grounding of physical actions and more accurately measure generalization capabilities of models.

3.1 Architecture

PIGLeT-Vis (shown in Figure 2) consists of separate components, which can combine multi-modal inputs in different ways. Through this modular approach, we can turn off specific components to evaluate how different inputs and model structures affect performance on the task. We test models with and without symbolic inputs and image inputs. For all components, we use a dropout of $p = 0.1$ in between layers and a default hidden layer size of $h = 64$.

3.1.1 Object Encoder

We reproduce Zellers et al. (2021), where all actions are assumed to involve two objects, $\mathbf{o}^0$ and $\mathbf{o}^1$, and the symbolic representation of objects are encoded in an Object Encoder model. The symbolic representation of an object before the action is represented by $\mathbf{o}_{pre}$. Both objects ($\mathbf{o}^0_{pre}$ and $\mathbf{o}^1_{pre}$) in the environment are described by a vector of 38 attributes, chosen on the basis that they are the kinds of physical attributes that are influenced by actions. They describe an object as small/large, cold/hot, empty/full, etc.

We first embed these symbolic object attributes

using an embedding layer $\mathbf{E}^{e \times h}$, where $e = 329$ is the total number of unique attributes and h is our hidden size. For an object k:

$$\hat{\mathbf{o}}^k_{pre} = \mathbf{E}(\mathbf{o}^k_{pre}) \quad (1)$$

The Object Encoder $\mathbf{O}_{encoder}$ takes in the embedded object attributes through a set of multi-head attention layers to encode the symbolic representation of each object. We use the default Pytorch implementation of the Transformer Encoder (Paszke et al., 2019) with three layers and 4 heads. The first encoded output of each object sequence is used for representing the entire object.

$$\mathbf{h}^k_{pre} = \mathbf{O}_{encoder}(\hat{\mathbf{o}}^k_{pre}) \quad (2)$$

3.1.2 Action Encoder

Actions are encoded either as a symbolic triplet ⟨action, action object, action receptacle⟩ or as an annotated text describing an action being taken (e.g., "robot empties the cup").

During pre-training, the Action Encoder $\mathbf{A}_{pretrain}$ uses an action embedding layer $\mathbf{E}'$ to embed the first dimension of the action, and reuses the object embedding layer $\mathbf{E}$ to embed the action object name a_o and action receptacle name a_r. The action embedding layer $\mathbf{E}'$ has dimensionality $10 \times h$ for the 10 distinct actions. The three embedded representations are summed and passed to the Action Encoder's linear layers to produce $\mathbf{h}_a$ (see equation 3). Similarly to Zellers et al. (2021), a `tanh` activation is applied after each linear layer.

$$\mathbf{h}_a = \mathbf{A}_{pretrain}(\mathbf{E}'(\mathbf{a}) + \mathbf{E}(a_o) + \mathbf{E}(a_r))) \quad (3)$$

When fine-tuning on the annotated dataset, the action input is text and therefore we switch out the Action Encoder $\mathbf{A}_{pretrain}$ for $\mathbf{A}_{finetune}$—our text-based Action Encoder. $\mathbf{A}_{finetune}$ uses a RoBERTa-base[1] model (Liu et al., 2019) to process a tokenized version of the text input $\mathbf{a}_t$. The first token ([`CLS`]) of the RoBERTa output layer is used to represent the action sequence and then passed through a linear layer to map the dimensionality of the hidden states from 256 to h.

$$\mathbf{h}_a = \mathbf{A}_{finetune}(\mathbf{a}_t) \quad (4)$$

[1]Implementation and pre-trained model weights are taken from the Huggingface library (Wolf et al., 2019).

3.1.3 Vision Object Encoder

The Vision Object Encoder takes in images ($\mathbf{i}_{pre}$ and $\mathbf{i}_{post}$) to provide a visual representation of each object k before and after ($\mathbf{h}^k_{pre}$ and $\mathbf{h}^k_{post}$). We use the DETR[1] (Carion et al., 2020) model as a backbone to predict N bounding boxes in a pair of images (pre- and post-action). As DETR is pre-trained on the COCO object detection dataset (Lin et al., 2014), its predicted object labels do not align with those in PIGPeN. Therefore, we instead learn a mapping between the predicted bounding box representations and the PIGPeN objects. For each image, we obtain a hidden representation $\mathbf{h}_b$ of dimensionality $N \times 256$ where $N = 100$.

We use an attention mechanism over the bounding boxes' hidden representation, conditioned on the object names. For a given object o^k, its conditional representation $\mathbf{h}^k_c$ is the encoded name of the object: $\mathbf{E}(o^k_{name})$. We can therefore obtain the attention score of a given object o^k and image $\mathbf{i}_m$ by calculating the alignment between the conditional representation $\mathbf{h}^k_c$ and the hidden representations of bounding boxes $\mathbf{h}_{b_m}$:

$$\mathbf{h}_{b_m} = \text{DETR}(\mathbf{i}_m) \quad (5)$$

$$\boldsymbol{\alpha}^k_m = \text{Softmax}\left(\sum_{i=1}^{h}(\mathbf{h}^k_c \mathbf{h}_{b_m})_i\right) \quad (6)$$

We obtain the final representation for a given object and image by multiplying our attention scores $\boldsymbol{\alpha}$ with the extracted output representation from DETR and summing along the bounding box axis:

$$\mathbf{h}^k_{o_m} = \mathbf{W}\left(\sum_{j=1}^{b}(\boldsymbol{\alpha}^k_m \mathbf{h}_{b_m})_j\right) \quad (7)$$

We use a final output layer $\mathbf{W}$ to decrease the dimensionality of $\mathbf{h}_o$ from the DETR dimensionality of 256 to h.

Through the Vision Object Encoder, we replace the previously symbolic inputs with images and can extract $[\mathbf{h}^0_{pre}\mathbf{h}^1_{pre}]$ and $[\mathbf{h}^0_{post}\mathbf{h}^1_{post}]$ from $\mathbf{i}_{pre}$ and $\mathbf{i}_{post}$ respectively. Note that we make the implicit assumption that $\mathbf{i}_{pre}$ and $\mathbf{i}_{post}$ contain the information necessary to predict object attributes of the objects post-action.

3.1.4 Action Apply

The Action Apply Model $\boldsymbol{\beta}$ is a simple fuse operation (concatenation in the hidden dimension) followed by three linear layers, which combine

the action representation $\mathbf{h}_a$ and an object representation of the scene pre-action $\mathbf{h}^k_{pre}$. The model outputs an object's representation $\mathbf{h}^k_a$, containing information conditioned all inputs:

$$\mathbf{h}^k_a = \boldsymbol{\beta}(\mathbf{h}_a, \mathbf{h}^k_{pre}) \tag{8}$$

3.2 Object Decoder

Finally, the Object Decoder is a transformer module that maps the object representations h_o from the pre-action state back to 38 symbolic attributes. It uses a default three layer Transformer Decoder (Paszke et al., 2019) that takes the hidden representation from the Action Apply $\mathbf{h}^k_a$ as an encoded memory state and $\mathbf{h}^k_{pre}$ as the source sequence to predicts a label for each attribute.

$$\dot{\mathbf{o}}^k_{post} = \mathbf{O}_{decoder}(\mathbf{h}^k_a, \mathbf{h}^k_{pre}) \tag{9}$$

When we use image inputs, we also have access to the post-action visual representation and can therefore use $\mathbf{h}^k_{pre} + \mathbf{h}^k_{post}$ instead of $\mathbf{h}^k_{pre}$.

The output has post-action object states $\dot{\mathbf{o}}^k_{post}$ which are compared to the ground truth $\mathbf{o}^k_{post}$ to calculate cross-entropy. As an additional loss, we also use the cross-entropy between $\dot{\mathbf{o}}^k_{pre}$ and $\mathbf{o}^k_{pre}$ by passing an empty $\mathbf{h}^k_a$ to force the Object Decoder to recreate the attributes in the pre-action state. We weight both losses equally.

3.3 Evaluation Metrics

Since our task involves predicting 38 attributes for two different objects per example, we follow Zellers et al. (2021) and report different types of accuracy metrics on the test set (after fine-tuning). We measure the overall accuracy by scoring how many objects have all attributes correctly predicted (exact match). Note that this is a high bar for a model where the symbolic representations are latent: to predict an object correctly, our model must first estimate its attributes before the action and then estimate whether and how these change given an action. So we also measure the attribute-level and action-level accuracies of each model, so as to explore which attributes and actions are more difficult to predict than others.

3.4 PIGPeN-Vis Dataset Split

To evaluate physical commonsense reasoning using PIGLeT-Vis, we filter PIGPeN (Zellers et al., 2021) to create a subset (PIGPeN-Vis) which we use for all our experiments. We motivate PIGPeN-Vis as a way to isolate the effects of adding our vision component, because while PIGPeN already has images, these images were not used in PIGLeT.

The PIGPeN dataset consists of trajectories of an environment before (pre) and after ($post$) an action is taken. Each trajectory contains representations of two distinct objects before and after. One of the objects is usually targeted by the action, while the other acts as a distractor. In addition, image pairs $(\mathbf{i}_{pre}, \mathbf{i}_{post})$ for each trajectory are provided, where each image is snapshot of the simulated photo-realistic 3D environment which contains the objects in view (see Appendix B for an example). Each image is an RGB image of dimensions 640×385.

The original dataset is separated into two distinct sets:

1. A *pre-training* set of $278,009$ trajectories, which includes the symbolic representations of objects **o** before and after a symbolic action **a** is taken. A separate validation set of $33,042$ examples is also included.

2. A *fine-tuning* set of $1,000$ trajectories which has been annotated to replace the symbolic action **a** with a textual representation $\mathbf{a_t}$ describing the action. Separate validation and test sets of 500 examples each are also included. All metrics are reported on the test set.

In PIGPeN, the object states $\mathbf{o}_{pre}$ and $\mathbf{o}_{post}$ contained 40 different attributes and 13 different actions **a**. Attributes range from intrinsic such as `name` or `moveable` to stateful such as `distance` or `isCooked`. In forming PIGPeN-Vis, we remove two attributes and three actions from the dataset to obtain 38 attributes and 10 possible actions (see Appendix B for more details).

3.4.1 Viewpoint and Action Filtering

Since the PIGPeN images were not generated with the goal of being used as input data, we identified several issues with the quality of certain scenes. A notable difficulty is that in some cases, the before and after images are not captured from the same camera angle or they have different lighting conditions. Changing orientations and lighting conditions makes it difficult to use an image pair $(\mathbf{i}_{pre}, \mathbf{i}_{post})$ to isolate the outcome of an action. Conversely, image pairs with too few perceivable differences also break our assumption that the changes in

the environment are perceivable. Therefore, we filter the dataset using pixel statistics to remove image pairs that have either large perceivable differences (likely due to changes in viewpoint) or small perceivable differences (where the action's results are not visually salient enough) (see Appendix B.2). We exclude 15.4% of the total dataset through visual filtering of the original dataset.

3.4.2 Zero-Shot Filtering

To evaluate the generalization capabilities gained from a vision component, we further filter the dataset to exclude a subset of training examples. Unlike the original PIGPeN dataset which only tested for zero-shot generalization at the level of the fine-tuning data, we remove all instances with selected specific objects or action-object pairs from all training and validation sets. To minimize the effect of removing examples from the dataset, we pick objects and action-object pairs with an already low number of samples in the training sets. In total, we exclude 14 objects and 27 action-object pairs, which amounts to less than 3% $(6,816$ samples) of the remaining training sets (see Appendix B.3). These zero-shot examples comprise around 10% of the test set.

After both filtering stages, PIGPeN-Vis contains a pre-training dataset of $232,625$ trajectories with a validation set of $26,823$, and a fine-tuning training set of 750 examples with a validation set of 367 examples and a test set of 398 examples.

3.5 Training Configurations

We evaluate the impact of the vision component on PIGPeN-Vis through five different setups:

- `base`: We implement a baseline model without symbolic object inputs. Our implementation removes the Object Encoder entirely, such that the model must predict the attributes of objects solely from knowing the action and the object names that it relates to. This model acts as a lower bound on the capabilities of the vision model: its performance would match the vision model if images are irrelevant to solving the task.

- `base+symbolic`: This is our implementation of the original Zellers et al. (2021) PIGLeT model, shown in Figure 1. This model acts as an upper bound on the capabilities of the vision model since it observes the true symbolic representations of objects before the action (which the vision model must estimate).

- `base+images`: This is our proposed PIGLeT-Vis, shown in Figure 2, where the Vision Object Encoder replaces the previously symbolic Object Encoder. This model leverages the before and after images of the environment as well as the name of the objects to extract representations of the object attributes.

- `base+symbolic+images`: We sum the hidden symbolic representations of objects with their visual representations in a unified model. Through this setup, we evaluate whether images can provide additional information to the already comprehensive symbolic representations.

- `base+images+text-labels`: We convert the symbolic representations of the labels for the object names and actions to their text label and encode them using a frozen LLM during pre-training. We use the same LLM to encode the text labels that we later use in the fine-tuning stage. This setup replaces all symbolic inputs from the pre-training stage to only language and image inputs.

Note that there are a few differences between the original Zellers et al. (2021) model and our implementation of `base+symbolic`. For instance, for simplicity, we opted to use an off-the-shelf RoBERTa-base (Liu et al., 2019) model instead of training our own custom GPT2 (Radford et al., 2019). Additionally, we also reduce the dimensionality of the PIGLeT layers from $h = 256$ to $h = 64$. We found that not only does this allow faster training times as it shrinks the Physical Dynamics model from 11.9 million parameters to 2 million parameters, it also improves the overall accuracy by a small margin $(+1.51\%)$.

We train each model for 80 epochs with a batch size of 256 using the Pytorch implementation of the Adam optimizer (Kingma and Ba, 2014) and a learning rate of 10^{-3} during pre-training and 10^{-5} during fine-tuning. We run each setup over 10 different seeds and report the average and standard deviation for each metric (see Appendix C.1 for more details).

	Accuracy ($\% \pm \sigma$)	
	Overall	Zero-Shot
base	21.23 ± 0.72	5.34 ± 2.77
base+symbolic (PIGLeT)	85.03 ± 0.45	39.04 ± 3.37
base+symbolic+images	86.01 ± 0.89	35.89 ± 3.47
base+images (PIGLeT-Vis)	45.47 ± 1.50	7.53 ± 2.60
base+images+text-labels	47.55 ± 2.10	8.90 ± 3.24

Table 1: Overall and zero-shot accuracies (PIGPeN-Vis)

4 Results and Discussion

We evaluate all models on our PIGPeN-Vis split and report the overall (exact match), zero-shot, action-level, and attribute-level accuracy results for all setups in Tables 1 and 2. For completeness, we also evaluate models on the original PIGPeN to contrast the effects of our filtering operations (see §3.4 and Appendix D) and find PIGPeN-Vis is a more challenging subset for all models.

The base model provides a low bar estimate of what is achievable using only the action encoder inputs. Unsurprisingly, the base model performs worst on overall accuracy, which demands an exact match of all attributes. It does relatively well on (individual) attribute-level accuracy, primarily because it predicts the most common attribute for each object. Some actions are also easier than others—for instance, the model reaches 27.38% accuracy on ToggleOn from only knowing the action and object names. This is likely because ToggleOn is constrained to a small set of objects and effects.

Our base+symbolic model obtains similar results to the original implementation by Zellers et al. (2021), with an overall accuracy of 85.03%. However, it performs much worse on the zero-shot split (39.04%) than the original PIGLeT model reported (80.2%) (Zellers et al., 2021). This disparity can be explained by the fact that the original zero-shot PIGPeN dataset was not a true zero-shot dataset, because the Physical Dynamics model was exposed to the "unseen" objects in its pre-training. The base+symbolic model provides a high bar estimate of what could be achievable if: (i) $\mathbf{i}_{pre}$ and $\mathbf{i}_{post}$ capture the symbolic environment; and (ii) the Vision Object Encoder can subsequently extract these features. However, as we will argue in Section 6, both (i) and (ii) are unrealistic given the constraints of both the dataset and the model.

Our base+images (PIGLeT-Vis) model scores 45.28% in overall accuracy but only 7.53% on the zero-shot set. Nevertheless, it outperforms the base model in overall accuracy ($p < 0.0001$) and in zero-shot accuracy ($p = 0.08$), which demonstrates that the images improve the prediction of the effects of actions. The base+images model also performs significantly better than base on difficult attribute-level accuracies such as distance ($p < 0.0001$). However, as before, accuracy on individual attributes benefits from the skewed distributions of their values and does not necessarily translate to high scores on predicting all 38 attributes correctly.

Utilizing both images and symbolic representations as inputs helps the base+symbolic+images model outperform purely symbolic inputs in overall accuracy, from 85.03% to 86.01% ($p < 0.01$). However, image inputs also decrease the model's zero-shot performance from 39.04% to 35.89%, although this isn't statistically significant ($p = 0.05$) due to high variance. We suspect that this high variance is caused by an increase in noise in the model resulting from adding images to the symbolic model. However, the overall picture is more complicated, as images can also provide gains on certain actions (e.g., PickUp accuracy increases from 80.48% to 86.14%) even though it causes a decrease in many other cases (e.g., ToggleOn).

Finally, when we utilize NL descriptions to replace the formal symbolic inputs (action name and object names), base+images+text-labels improves overall accuracy when compared to base+images from 45.47% to 47.55% ($p = 0.02$). Text inputs appear to improve zero-shot accuracy, but not by a statistically significant margin ($p = 0.31$). Accuracy also improves in most actions, for instance the Slice accuracy improves from 41.64% to 45.57% ($p = 0.03$). So the NL descriptions inform the task in a beneficial way, over and above the raw images. But encoding the labels as text rather than formal symbolic representations also adds noise.

Nevertheless, text labels improve accuracy on actions where the semantic information contained in the label provides a richer context to help generalize to similar objects. For instance, a "cup" and a "mug" are semantically close, and thus learning the effects of actions on a "cup" might help the model predict the same effects on a "mug" even if the word forms are different. In contrast, the formal symbolic representations treat the predicate symbols cup and mug as unrelated, and so don't benefit from the lexical relationships that the LLM captures. Fully removing the symbolic representations allows us to adapt our model

	Action Accuracy (%)				Attribute Accuracy (%)		
	Open	Pickup	ToggleOn	Slice	size	distance	temperature
base	8.33	10.96	27.38	22.13	73.78	51.01	95.91
base+symbolic (PIGLeT)	85.73	80.48	**96.90**	75.41	94.98	95.13	**99.85**
base+symbolic+images	**88.75**	**86.14**	92.86	**81.31**	**96.35**	**96.13**	99.59
base+images (PIGLeT-Vis)	20.83	33.49	70.24	41.64	87.03	76.62	96.10
base+images+text-labels	22.92	40.12	67.14	45.57	87.89	78.06	96.72

Table 2: Action and attribute specific accuracies for a subset of actions and attributes; for a comprehensive table with standard deviations see Appendix D. size and distance each have eight possible classes while temperature has three.

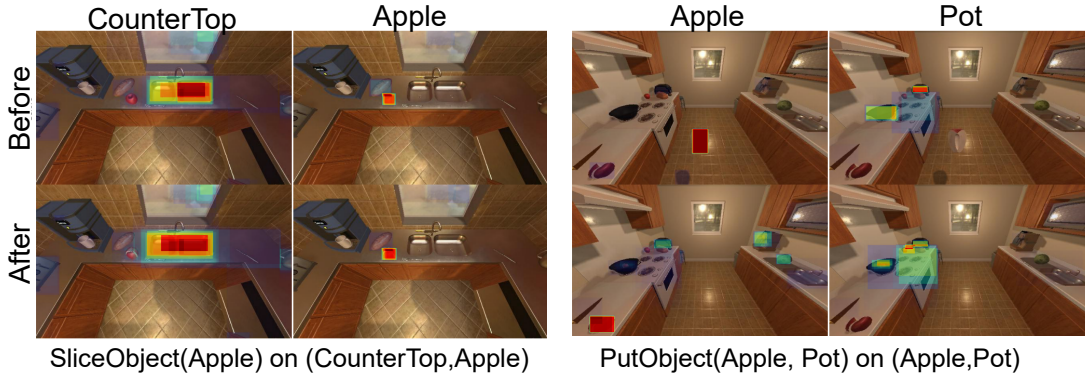


Figure 3: We visualize the attention of the Vision Object Encoder from a trained base+images model on two different actions and environments. The left grid focuses on the effect of Slice(Apple) on CounterTop and Apple, while the right grid focuses on the effects of Slice(Apple) on Apple and Pot objects.

to any possible unseen object during test time. base+images+text-labels is adaptable to general settings without knowing the symbolic mapping of objects and actions in the environment.

The results of both base+symbolic+images and base+images+text-labels make the case multimodal modeling of commonsense reasoning, as both language and images are complementary to generalize to unseen settings.

4.1 Qualitative Attention Maps

Visualizing attention is another benefit of a vision component, as we can see what the model focuses on and partially explain its predictions. Figure 3 shows two separate examples and corresponding attention maps. In the left example, base+images is tasked with predicting the attributes of CounterTop and Apple after the Slice action is applied on the Apple. In the right example, the Put action is applied on the Apple, and the model must predict the attributes of the Apple and the distractor object Pot. The two rows are the before and after images ($\mathbf{i}_{pre}$ and $\mathbf{i}_{post}$), and the two columns are the two objects used to condition the attention. The attention maps display the strength of the attention for each bounding box given an object name.

Both examples in Figure 3 show that the Vision Object Encoder can map known objects to relevant bounding boxes. The model successfully tracks the Apple in both cases by placing the most weight on the bounding box targeting the Apple. However, these examples also show the difficulty of this task—the environments are realistic and can be filled with more than one instance of an object.

5 Conclusion

In this paper, we tackle the task of predicting the effects of actions on objects' physical attributes. In contrast to (Zellers et al., 2021), our model does not treat the formal symbolic representation of the images as observed. Instead, PIGLeT-Vis supports inference when the inputs are images alone or images plus NL descriptions and a phrase denoting the action (e.g., "the robot empties the cup"). While PIGPeN offers challenges for applying a multimodal approach, our model can extract useful information from images, opening the door for generalizing learning physical commonsense to real-world data. Importantly, our PIGPeN-Vis split can be used to evaluate the zero-shot capabilities of different model configurations. Moreover, while base+symbolic still outperforms base+images, it does so without estimating the attributes of ob-

jects and thus solves a much easier but unrealistic task. Through `base+images+text-labels`, we show that, when replacing symbolic inputs, the best solution is to complement image inputs with NL descriptions to leverage information from both modalities. Finally, our results show the need to improve the generalization capabilities of multimodal models such that they can learn and adapt to unseen situations.

6 Limitations

There are several limitations to our approach that result directly from the inherent limitations of PIGPeN and our proposed Vision Object Encoder respectively.

PIGPeN was not originally designed for testing commonsense reasoning using images and contains numerous inconsistencies which cannot all be solved with the PIGPeN-Vis split obtained from filtering (Section 3.4.1). Given the presence of non-physically salient attributes such as `temperature`, images are not guaranteed to fully capture their symbolic representations. PIGPeN includes certain attributes which are not discernible from images, e.g., even humans would be unable to tell a hot plate from a cold plate from vision alone. The images in PIGPeN can also contain more than one object (e.g., more than one cup) without ever specifying which one the symbolic representation refers to. This causes difficulty for our approach because judging specific attributes such as distance is impossible if there are two cups at different distances from the viewpoint. Additionally, PIGPeN also discretizes continuous variables such as `distance` into categories which can be hard to disambiguate.

To approach the accuracy of `base+symbolic` with our vision component, we also need a vision representation from which to correctly estimate all latent attributes. Even if images are assumed to be perfect representations of the symbolic environment, the model still has to extract each of the 38 attributes correctly for both objects using only two images. It is possible (and likely) for the vision detection backbone to miss the target object entirely because it is not trained to detect the specific object in question. We see this effect in Figure 3, where the model falls back to using a bounding box around the sink area to describe the `CounterTop` object. The DETR vision model used to extract bounding boxes was pre-trained on the COCO dataset (Lin et al., 2014) which does not contain `CounterTop` as an object. PIGLeT-Vis is therefore ultimately limited by the capabilities of its vision backbone.

Ethics Statement

While this work does not introduce new data or involve human participants, we use the PIGPeN dataset which contains human-labelled data. The fine-tuning portion of the dataset was annotated through MTurk by Zellers et al. (2021) and they report following best practices (paying decent wages, providing feedback and using a qualification test) in their data collection. We filter and use a subset of PIGPeN and introduce methods to learn the effects of actions in a multimodal setting. We, therefore, believe that our work does not raise any ethical concerns.

Acknowledgements

This work was supported in part by the UKRI Centre for Doctoral Training in Natural Language Processing, funded by the UKRI (grant EP/S022481/1) at the University of Edinburgh, School of Informatics and School of Philosophy, Psychology & Language Sciences and by he UKRI- funded TAS Governance Node (grant number EP/V026607/1).

References

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *AAAI*.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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*, volume 33, page 1877–1901. Curran Associates, Inc.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey

Zagoruyko. 2020. End-to-end object detection with transformers. *CoRR*, abs/2005.12872.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Uniter: Universal image-text representation learning. In *ECCV*.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Maxwell Forbes, Ari Holtzman, and Yejin Choi. 2019. Do neural language representations learn physical commonsense? In *CogSci*.

Qiaozi Gao, Shaohua Yang, Joyce Chai, and Lucy Vanderwende. 2018. What action causes this? towards naive physical action-effect prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 934–945, Melbourne, Australia. Association for Computational Linguistics.

Michael Hanna, Federico Pedeni, Alessandro Suglia, Alberto Testoni, and Raffaella Bernardi. 2022. ACTthor: A controlled benchmark for embodied action understanding in simulated environments. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5597–5612, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

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)*, pages 770–778.

Jena D Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6384–6392.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2020. What does BERT with vision look at? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5265–5275, Online. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755. Springer International Publishing.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

R. Thomas McCoy, Junghyun Min, and Tal Linzen. 2020. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 217–227, Online. Association for Computational Linguistics.

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.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. *arXiv:2103.00020 [cs]*. ArXiv: 2103.00020.

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

Shailaja Keyur Sampat, Akshay Kumar, Yezhou Yang, and Chitta Baral. 2021. CLEVR_HYP: A challenge dataset and baselines for visual question answering with hypothetical actions over images. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3692–3709, Online. Association for Computational Linguistics.

Ke Shen and Mayank Kejriwal. 2021. On the generalization abilities of fine-tuned commonsense language representation models. In *Artificial Intelligence XXXVIII*, page 3–16. Springer International Publishing.

Kunal Pratap Singh, Suvaansh Bhambri, Byeonghwi Kim, Roozbeh Mottaghi, and Jonghyun Choi. 2021. Factorizing perception and policy for interactive instruction following. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1888–1897.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Da Yin, Liunian Harold Li, Ziniu Hu, Nanyun Peng, and Kai-Wei Chang. 2021. Broaden the Vision: Geo-Diverse Visual Commonsense Reasoning. In *EMNLP*.

Fei Yu, Jiji Tang, Weichong Yin, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2021. Ernie-vil: Knowledge enhanced vision-language representations through scene graph. In *AAAI*.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From recognition to cognition: Visual commonsense reasoning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 6713–6724. IEEE.

Rowan Zellers, Ari Holtzman, Matthew Peters, Roozbeh Mottaghi, Aniruddha Kembhavi, Ali Farhadi, and Yejin Choi. 2021. Piglet: Language grounding through neuro-symbolic interaction in a 3d world. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

A GPT-3 Example of Physical Reasoning

The weight of the potato is 150 grams.
The robot then slices the potato into thin slices.
The weight of the potato is now `75 grams.`

Figure 4: Example of incorrect physical commonsense by an LLM. When predicting what comes after the **input text**, the large 175 billion parameter GPT-3 (Brown et al., 2020) `predicts` that the weight of the potato halves after a slicing action is taken.

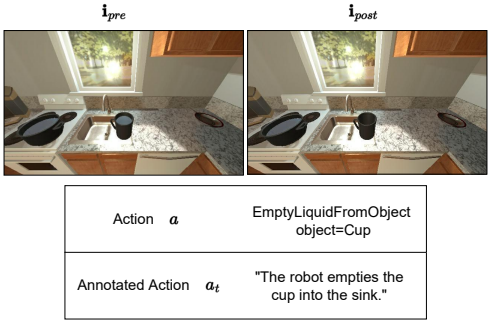


Figure 5: Image pair and actions for a selected PIGPeN example.

	pre		*post*	
	$\mathbf{o}_{pre}^{cup}$	$\mathbf{o}_{pre}^{faucet}$	$\mathbf{o}_{post}^{cup}$	$\mathbf{o}_{post}^{faucet}$
ObjectName	Cup	Faucet	Cup	Faucet
Contained Objects				
Is contained in...				
Mass	1 to 2lb	Massless	1 to 2lb	Massless
Size	small	medium	small	medium
Temperature	RoomTemp	RoomTemp	RoomTemp	RoomTemp
Distance	1 to 2ft	3 to 4 ft	1 to 2ft	3 to 4 ft
Breakable	Yes	No	Yes	No
Cookable	No	No	No	No
CanBecomeDirty	Yes	No	Yes	No
IsBroken	No	No	No	No
IsCooked	No	No	No	No
IsDirty	No	No	No	No
IsFilledWithLiquid	Yes	No	**No**	No
IsOpen	No	No	No	No
IsPickedUp	Yes	No	Yes	No
IsSliced	No	No	No	No
IsToggled	No	No	No	No
Moveable	No	No	No	No
Openable	No	No	No	No
Pickupable	Yes	No	Yes	No
CanHoldItems	Yes	No	Yes	No
Sliceable	No	No	No	No
Toggleable	No	Yes	No	Yes
Materials	Ceramic		Ceramic	

Table 3: Attributes for a selected PIGPeN example. The total number of attributes is 38 as the Materials attribute is a multi-hot encoding.

B PIGPeN-Vis

We select an example from PIGPeN to display in Figure 5 and Table 3.

From the original dataset, we remove two attributes (`isUsedUp` and `salientMaterials_Organic`) because they are unchanged in all examples. We also remove 3 actions (`ThrowObject10`, `ThrowObject100` and `ThrowObject1000`) which are all related to throwing an object across a certain distance. These actions account for only a small subset of the dataset and create inconsistent image pairs due to the agent's momentum being captured in the images. The angle of the camera changes as a result of `ThrowObject` and this breaks our assumption that the difference between $\mathbf{i}_{pre}$ and $\mathbf{i}_{post}$ solely reflects the effects of the action on the environment (and not on the viewer). We therefore reduce the total number of symbolic attributes per object to 38 and the number of possible actions to 10.

B.1 Attributes

The following 38 symbolic attributes are used to describe an object in PIGPeN:

`ObjectName, parentReceptacles, receptacleObjectIds, distance, mass,size, ObjectTemperature, breakable, cookable, dirtyable, isBroken, isCooked, isDirty, isFilledWithLiquid, isOpen, isPickedUp, isSliced, isToggled, moveable, openable, pickupable, receptacle, salientMaterials_Ceramic, salientMaterials_Fabric, salientMaterials_Food, salientMaterials_Glass, salientMaterials_Leather, salientMaterials_Metal, salientMaterials_Paper, salientMaterials_Plastic, salientMaterials_Rubber, salientMaterials_Soap, salientMaterials_Sponge, salientMaterials_Stone, salientMaterials_Wax, salientMaterials_Wood, sliceable, toggleable`

B.2 Filtering Statistics

We initially filter the PIGPeN dataset using two main strategies to remove images with too much or too little change between the pre and post images. In both cases, the goal is to remove pairs of images in which it would be impossible for a vision model to predict what has changed.

Images with too many changes are often images taken from different viewpoints or with different lighting conditions. We filter these images by looking at the number of pixels changed between $\mathbf{i}_{pre}$ and $\mathbf{i}_{post}$. We show the distribution of the number of pixels changed per image over the training dataset in Figure 6. Using this visualization we can clearly see a small peak at the extreme - where almost all the pixels in $\mathbf{i}_{post}$ are different from $\mathbf{i}_{pre}$. Note that since each image is an RGB image of dimensions 640×385, the max number of change is $640 \times 385 \times 3 = 739,200$ (we also compare pixels across color channels). We opt to remove all images with more than $400,000$ changes, which corresponds to around 6.2% of the training dataset.

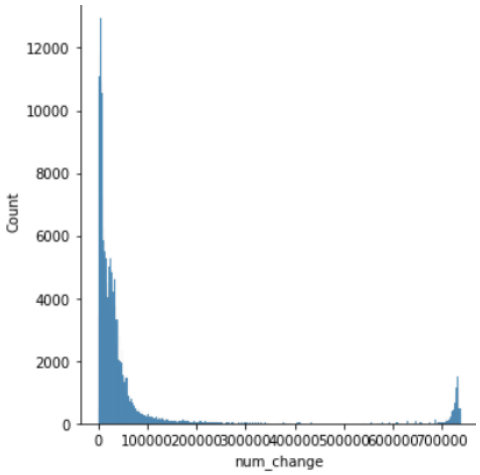


Figure 6: Distribution of the number of pixels changed per image in the PIGPeN dataset.

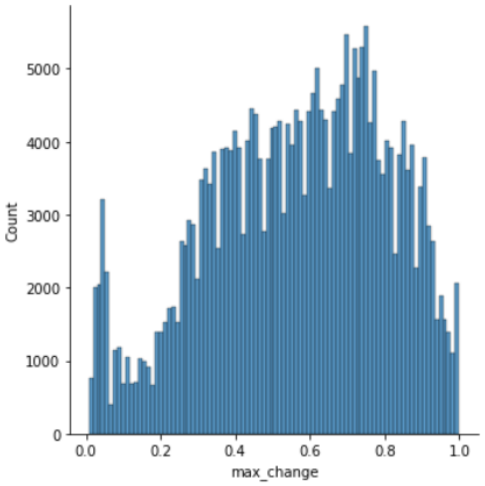


Figure 7: Distribution of the maximum pixel value changed per image in the PIGPeN dataset.

Images with too little change could be examples of where the action has no visual outcome and $\mathbf{i}_{pre}$ and $\mathbf{i}_{post}$ are indistinguishable. To filter these images we measure the maximum magnitude of change in each pixel and each color channel be-

tween the pairs of images. We visualize the max change across the training dataset in Figure 7. Here a low values implies almost no salient change, and as max change approaches zero - it becomes unlikely that a human would be able to perceive the difference between the pair of images. We opt for to keep images with a max change greater than 0.2 which corresponds to excluding 7.8% of the training dataset.

Filtering on the number of changed pixels lead to the exclusion of around 13.89% of the training dataset.

B.3 Zero-shot Filtering

We remove the following 14 objects from both the train and validation ($3,401$ examples total):

`HandTowel, Towel, Plunger, Watch, CD, SoapBottle, Pen, RemoteControl, SoapBar, Box, Bottle, CreditCard, Statue, KeyChain`

We remove the following 27 action-object pairs from both the train and validation ($3,278$ examples total):

`(CloseObject,Toilet), (DirtyObject,Pan), (DirtyObject,Pot), (EmptyLiquidFromObject,Bottle), (EmptyLiquidFromObject,Pot), (OpenObject,Toilet), (PickupObject,Box), (PickupObject,CellPhone), (PickupObject,CreditCard), (PickupObject,KeyChain), (PutObject,CD), (PutObject,CreditCard), (PutObject,HandTowel), (PutObject,Laptop), (PutObject,Lettuce), (PutObject,Pen), (PutObject,Plunger), (PutObject,Pot), (PutObject,RemoteControl), (PutObject,SoapBar), (PutObject,SoapBottle), (PutObject,Statue), (PutObject,ToiletPaper), (PutObject,Towel), (PutObject,Watch), (ToggleOff,CellPhone), (ToggleOff,Television)`

C Code Release and Training

Our full code, models, and PIGPeN-Vis split can be found at github.com/gautierdag/piglet-vis.

C.1 Additional Training Details

As previously mentioned, there are a few differences between the original Zellers et al. (2021) model and our implementation of `base+symbolic`. We use an off-the-shelf RoBERTa-base (Liu et al., 2019) model instead of a custom GPT2 (Radford et al., 2019). Additionally, we also reduce the dimensionality of the PIGLeT layers from $h = 256$ to $h = 64$. This shrinks the overall model (excluding the LLM) from 11.9 million parameters to less than 2 million parameters during pre-training and improves the overall accuracy by a small margin ($+1.51\%$). We do not run any other hyper-parameter search throughout our experiments and wherever possible use the same hyper-parameters as PIGLeT. We also reduce the batch size from 1024 to 256 because we use a mix of NVIDIA GTX 1080 and NVIDIA A100 GPUs and wish to keep batch size constant.

The `+images` models use the extracted representations from a frozen off-the-shelf DETR model (41.3 million parameters), however it is ran only once over all images as we cache its predictions. We do not use the "NO OBJECT" predictions from DETR, and simply pass all 100 bounding boxes representations to the attention mechanism. Since we do not have access to the true bounding boxes in PIGPeN, we do not fine-tune DETR and therefore ignore its prediction heads which have also been trained on COCO and mismatch our possible objects.

The `+symbolic` models use the Symbolic Object Encoder which is an additional $800,000$ parameters on its own. During fine-tuning all models use a RoBERTA-base model ($+120$ million parameters) in the Action Encoder. The `+text-label` model also uses the RoBERTA-base model during pre-training, but again this is frozen and its outputs are cached for the full dataset.

We pre-train each model for 80 epochs and fine-tune for 60 epochs. For all setups, pre-training takes between 1 to 2 hours and fine-tuning takes less than 1 hour on an NVIDIA A100 GPU. We use the Pytorch implementation of the Adam optimizer (Kingma and Ba, 2014) and a learning rate of 10^{-3} during pre-training and 10^{-5} during fine-tuning. We use early stopping on the validation loss with a patience of 10 epochs. We run each setup over 10 different seeds ($s \in [1, 2, ..., 10]$ and report the average and standard deviation for each metric.

D Accuracy Results

D.1 Comparing PIGPeN and PIGPeN-Vis

Table 4 compares the overall accuracy on the original PIGPeN dataset with our proposed PIGPeN-Vis split. We find that our PIGPeN-Vis split is consistently harder to solve than the original PIGPeN dataset. We explain the increased accuracy in the original dataset with the fact that some of the filtered out actions (see Appendix B) are easy to

	Overall Accuracy ($\% \pm \sigma$)		
	PIGPeN	PIGPeN-Vis	Δ
base	29.18 ± 0.34	21.23 ± 0.72	-7.95%
base+symbolic (PIGLeT)	86.39 ± 0.79	85.03 ± 0.45	-1.36%
base+symbolic+images	87.45 ± 0.66	86.01 ± 0.89	-1.44%
base+images (PIGLet-Vis)	49.13 ± 1.53	45.47 ± 1.50	-3.66%
base+images+text-labels	51.28 ± 1.68	47.55 ± 2.10	-3.73%

Table 4: Overall Accuracies comparing full PIGPeN with the PIGPeN-Vis split across 10 seeds.

	Overall Accuracy ($\% \pm \sigma$)	
	validation	test
base	23.85 ± 0.95	21.23 ± 0.72
base+symbolic (PIGLeT)	88.08 ± 0.50	85.03 ± 0.45
base+symbolic+images	$\mathbf{89.49 \pm 0.82}$	$\mathbf{86.01 \pm 0.89}$
base+images	50.73 ± 2.97	45.47 ± 1.50
base+images+text-labels	53.33 ± 3.15	47.55 ± 2.10

Table 5: Validation and test overall accuracies. Note the zero-shot accuracy is not calculated on the validation set since there are no unseen examples in the validation set to prevent leakage.

solve from knowing the object name and action: e.g., most of the images we exclude due to little salient changes are appliances like stoves being turned on or off. However, it is easy for a model to predict the post-condition attributes of a stove, which are mostly static, across all examples given an action such as ToggleOn, which always has the same effect.

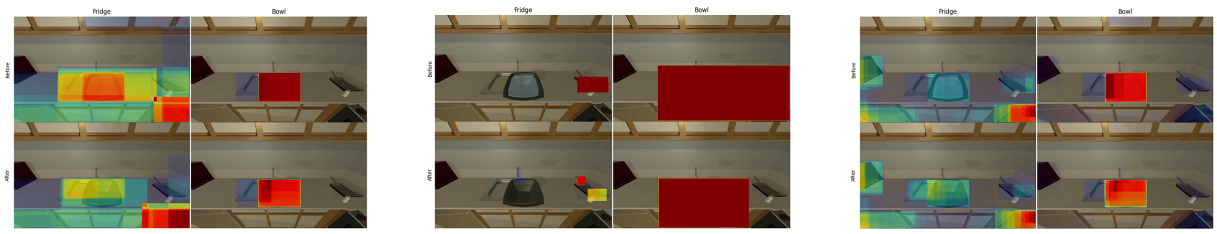
D.2 Complete Accuracy Results on PIGPeN-Vis

Table 5 shows the overall accuracies for both the test and validation sets. The full accuracy results for all actions in Table 6 and for all attributes in Table 7.

E Additional Attention Maps

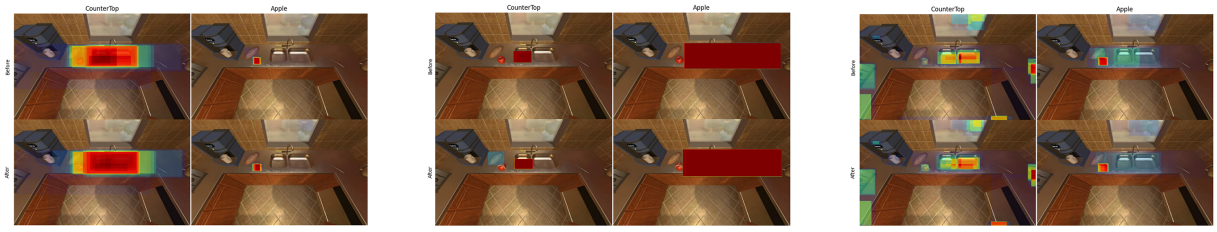
We plot additional attention visualizations for all three image models base+images, base+symbolic+images, and base+images+text-labels in Figures 8, Figures 9, and Figures 11. Since the DETR object detector remains frozen, all models have access to the same bounding boxes and bounding box representations. Qualitatively, we find that the attention weights of base+images and base+images+text-labels both learn to map to globally relevant bounding boxes given an objects. We also find the attention maps in base+images+text-labels to be less confident overall than base+images, likely due to the noise introduced by the semantic text inputs. As a result, base+images+text-labels makes less mistakes by not focusing too much attention to the wrong bounding box.

On the other hand, base+symbolic+images focuses on seemingly random bounding boxes. Since base+symbolic+images already receives the full representation of each objects, it does not learn to complement the object's representation with accurate visual information. While base+symbolic+images extracts 1% of additional overall accuracy from image inputs when compared to base+symbolic, it does so by falling back to vision for visually salient actions such as Pickup. base+symbolic+images focuses only a narrow set bounding boxes with overconfidence with no regard for whether or not the bounding box relates to the object. We posit that the model might use vision to better estimate more difficult attributes to predict such as distance in some contexts. Note Pickup is a salient action because when the agent in the environment picks an object up, the object is placed directly in the middle of its field of vision (as if the agent were holding the object in front of it).



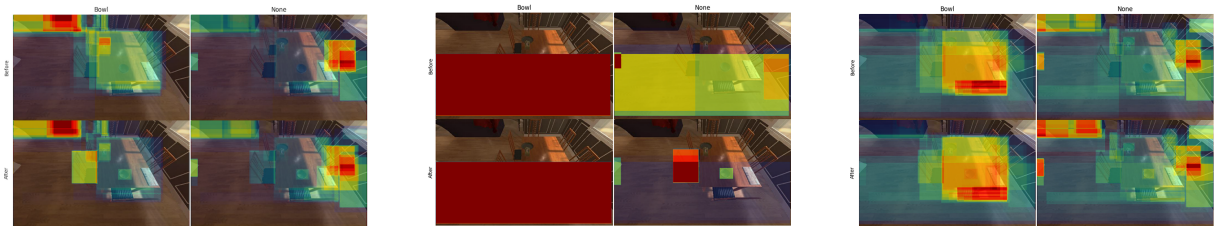
(a) base+images (b) base+symbolic+images (c) base+images+text-labels

Figure 8: Attention maps for the effects of the EmptyLiquid action on Bowl with objects Fridge and Bowl. The top row of each grid maps to the before environment and the bottom row maps to the after environment. The columns map to each respective object. The Fridge object appears in the lower left of the image, and is only correctly identified by base+images+text-labels, even though the model does place more weight to the bounding box of the stove (lower right).



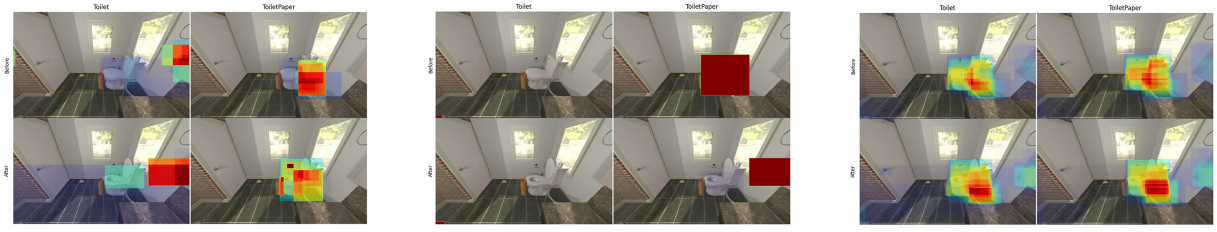
(a) base+images (b) base+symbolic+images (c) base+images+text-labels

Figure 9: Attention maps for the effects of the Slice action on Apple with objects CounterTop and Apple. The top row of each grid maps to the before environment and the bottom row maps to the after environment. The columns map to each respective object.



(a) base+images (b) base+symbolic+images (c) base+images+text-labels

Figure 10: Attention maps for the effects of the Dirty action on Bowl with objects Bowl and None. The top row of each grid maps to the before environment and the bottom row maps to the after environment. The columns map to each respective object. None can be an object in PIGPeN, but we do not predict its attributes and exclude it in all model predictions.



(a) base+images (b) base+symbolic+images (c) base+images+text-labels

Figure 11: Attention maps for the effects of the Open action on Toilet with objects Toilet and ToiletPaper. The top row of each grid maps to the before environment and the bottom row maps to the after environment. The columns map to each respective object. This particular example is an unseen combination of action and object that has been excluded from the training and validation set.

	Action Accuracy ($\% \pm \sigma$)				
	Close	Dirty	EmptyLiquid	HeatUpPan	Open
base	13.20 ± 1.06	17.71 ± 1.20	24.75 ± 5.75	36.33 ± 4.14	8.33 ± 1.84
base+symbolic	85.98 ± 1.77	94.00 ± 3.42	99.34 ± 1.15	100.00 ± 0.00	85.73 ± 0.99
base+symbolic+images	86.80 ± 3.29	90.29 ± 5.90	99.02 ± 2.07	99.17 ± 1.62	88.75 ± 3.02
base+images	27.42 ± 3.71	58.57 ± 2.78	69.34 ± 4.17	68.67 ± 3.75	20.83 ± 4.63
base+images+text-labels	28.87 ± 3.19	57.71 ± 3.24	70.16 ± 3.17	74.00 ± 3.16	22.92 ± 5.79
	Pickup	Put	Slice	ToggleOff	ToggleOn
base	10.96 ± 1.92	27.95 ± 1.19	22.13 ± 0.86	30.83 ± 3.39	27.38 ± 2.57
base+symbolic	80.48 ± 2.88	58.39 ± 1.94	75.41 ± 1.89	99.40 ± 0.84	96.90 ± 1.61
base+symbolic+images	86.14 ± 2.56	57.59 ± 2.31	81.31 ± 3.96	99.05 ± 0.75	92.86 ± 5.14
base+images	33.49 ± 3.45	34.91 ± 2.43	41.64 ± 3.80	71.43 ± 2.75	70.24 ± 16.00
base+images+text-labels	40.12 ± 2.61	38.30 ± 3.11	45.57 ± 3.85	69.05 ± 5.81	67.14 ± 16.53

Table 6: Full accuracy results table including the standard deviation over 10 seeds for all actions and setups.

	Attribute Accuracy ($\% \pm \sigma$)									
	Name	Temperature	attribute	breakable	cookable	dirtyable	distance	isBroken	isCooked	isDirty
base	99.66 ± 0.07	95.91 ± 0.41	96.12 ± 0.07	91.46 ± 0.36	99.95 ± 0.07	99.95 ± 0.10	51.01 ± 0.93	99.86 ± 0.00	98.60 ± 0.06	97.93 ± 0.19
base+symbolic	99.64 ± 0.12	99.85 ± 0.04	99.48 ± 0.03	99.84 ± 0.09	100.00 ± 0.00	100.00 ± 0.00	95.13 ± 0.35	100.00 ± 0.00	99.85 ± 0.04	99.71 ± 0.14
base+symbolic+images	99.62 ± 0.09	99.59 ± 0.27	99.48 ± 0.04	99.78 ± 0.10	100.00 ± 0.00	99.97 ± 0.09	96.13 ± 0.40	100.00 ± 0.00	99.85 ± 0.04	99.52 ± 0.32
base+images	97.34 ± 0.65	96.28 ± 0.74	97.25 ± 0.13	92.63 ± 0.75	99.91 ± 0.10	99.62 ± 0.20	76.90 ± 1.05	99.85 ± 0.05	98.68 ± 0.19	97.87 ± 0.34
base+images+text-labels	98.44 ± 0.35	96.05 ± 1.23	97.46 ± 0.13	93.19 ± 0.31	99.96 ± 0.09	99.93 ± 0.10	78.56 ± 1.16	99.84 ± 0.09	98.19 ± 0.84	97.78 ± 0.24
	isFilledWithLiquid	isOpen	isPickedUp	isSliced	isToggled	mass	moveable	openable	parentReceptacles	pickupable
base	96.79 ± 0.50	98.84 ± 0.23	94.83 ± 0.82	97.99 ± 0.09	98.36 ± 0.23	96.51 ± 0.15	99.90 ± 0.09	99.97 ± 0.06	87.44 ± 0.42	99.84 ± 0.09
base+symbolic	99.93 ± 0.12	98.95 ± 0.09	99.27 ± 0.31	100.00 ± 0.00	99.88 ± 0.12	99.33 ± 0.14	99.99 ± 0.04	99.97 ± 0.06	97.78 ± 0.47	99.90 ± 0.11
base+symbolic+images	99.84 ± 0.19	98.67 ± 0.38	98.96 ± 0.31	99.97 ± 0.06	99.74 ± 0.30	99.59 ± 0.09	100.00 ± 0.00	99.99 ± 0.04	97.26 ± 0.44	99.88 ± 0.10
base+images	96.88 ± 0.55	98.81 ± 0.97	97.43 ± 0.37	98.28 ± 0.30	97.92 ± 0.83	96.41 ± 0.41	99.79 ± 0.21	99.74 ± 0.20	91.05 ± 0.77	99.59 ± 0.17
base+images+text-labels	97.25 ± 0.45	98.11 ± 1.14	97.54 ± 0.53	98.34 ± 0.29	98.06 ± 0.55	96.74 ± 0.24	99.89 ± 0.09	99.95 ± 0.10	92.49 ± 0.69	99.70 ± 0.09
	receptacleIds	receptacle	Ceramic	Fabric	Food	Glass	Leather	Metal	Paper	Plastic
base	84.20 ± 0.61	99.85 ± 0.10	98.26 ± 0.17	99.55 ± 0.07	99.99 ± 0.04	98.91 ± 0.13	99.89 ± 0.06	98.69 ± 0.15	99.73 ± 0.00	98.30 ± 0.10
base+symbolic	96.36 ± 0.18	99.90 ± 0.09	100.00 ± 0.00	99.96 ± 0.07	100.00 ± 0.00	99.99 ± 0.04	100.00 ± 0.00	99.99 ± 0.04	100.00 ± 0.00	99.97 ± 0.06
base+symbolic+images	96.13 ± 0.30	99.92 ± 0.10	99.99 ± 0.04	99.85 ± 0.10	99.99 ± 0.04	99.97 ± 0.06	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.96 ± 0.07
base+images	82.87 ± 0.55	99.47 ± 0.21	99.03 ± 0.22	99.50 ± 0.19	99.92 ± 0.10	99.16 ± 0.21	99.97 ± 0.06	98.31 ± 0.37	99.67 ± 0.21	98.83 ± 0.31
base+images+text-labels	83.91 ± 0.56	99.69 ± 0.11	99.36 ± 0.19	99.44 ± 0.12	99.96 ± 0.09	99.37 ± 0.24	99.95 ± 0.10	98.69 ± 0.30	99.56 ± 0.19	99.08 ± 0.20
	Rubber	Soap	Sponge	Stone	Wax	Wood	size	sliceable	toggleable	
base	100.00 ± 0.00	99.99 ± 0.04	100.00 ± 0.00	99.34 ± 0.09	100.00 ± 0.00	99.51 ± 0.16	73.78 ± 0.29	98.02 ± 0.12	99.95 ± 0.07	
base+symbolic	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.99 ± 0.04	100.00 ± 0.00	99.99 ± 0.04	94.98 ± 0.19	100.00 ± 0.00	99.99 ± 0.04	
base+symbolic+images	99.97 ± 0.06	99.99 ± 0.04	100.00 ± 0.00	99.99 ± 0.04	100.00 ± 0.00	100.00 ± 0.00	96.35 ± 0.20	99.99 ± 0.04	99.96 ± 0.09	
base+images	99.88 ± 0.08	99.89 ± 0.11	99.92 ± 0.10	99.48 ± 0.14	99.92 ± 0.10	99.25 ± 0.22	87.03 ± 1.15	98.32 ± 0.32	99.81 ± 0.17	
base+images+text-labels	99.85 ± 0.08	99.92 ± 0.10	99.88 ± 0.14	99.60 ± 0.19	99.95 ± 0.07	99.37 ± 0.22	87.89 ± 1.11	98.32 ± 0.36	99.95 ± 0.07	

Table 7: Full accuracy results table including the standard deviation over 10 seeds for all attributes and setups.