A Appendix

A.1 Price Tagging Game Experiment

Price Tagging Game

```
Below is an instruction that describes a task, paired with an input that provides further
context. Write a response that appropriately completes the request.

### Instruction:
Please say yes only if it costs between [X.XX] and [X.XX] dollars, otherwise no.

### Input:
[X.XX]

### Response:
[Model Output]
```

Figure 6: Instruction template for the Price Tagging game. We have three variables inside the prompt: lower bound amount, high bound amount as well as the input amount where each number has two decimal points but bounded with [0.00, 9.99].

Figure 6 shows our templates for our Price Tagging game Experiment. This format is enforced by the instruction tuning template of the Alpaca model.

A.2 Training Details

Training Setups During training for Boundless DAS, model weights are frozen except for the rotation matrix. To train the rotation matrix, we use `Adam` [26] as our optimizer with an effective batch size of 64 and a learning rate of 10^{-3}. We use a different learning rate 10^{-2} for our boundary indices for quicker convergence. We train our rotation matrix for 3 epochs and evaluate with a separate evaluation set for each 200 training steps, and we save the best rotation matrix evaluated on the evaluation set during training and test on another hold-out testing set. Our training set has 20K examples. Our in-training evaluation is limited to 200 examples for a quicker training time, and the hold-out testing dataset has 1K examples. These datasets are generated on the fly for different causal models guarded by our random seeds. The rotation matrix is trained by using the orthogonalized parameterization provided by the `torch` library[3]. For boundary learning, we anneal temperature from 50.0 to 0.10 with a step number equal to the total training step, and temperature steps with gradient backpropagation. Each alignment experiment is enabled with `bfloat16` and can fit within a single A100 GPU. Each alignment experiment takes about 1.5 hrs. To finish all of our experiments, it takes roughly 2 weeks of run-time with 2 × A100 nodes with 8 GPUs each. We run experiments with 3 distinct random seeds and report the best result using IIA as our metrics.

Vanilla Task Performance Remarks To ensure the model we are aligning behaviorally achieves good task performance, we evaluate the model task performance with 1K sampled triples of values more than 300 times. The Alpaca (7B) model achieves 85% for randomly drawn triples of numbers on average. We want to emphasize that it is really hard to find a good reasoning task with intermediate steps (e.g., tasks like "Return yes of it costs more than 2.50 dollars" do not require an intermediate step) that 7B-30B LLMs can solve. We test different reasoning tasks (e.g., math, logic, natural QA, code, etc..) with instruct-tuned LLaMA [49], Dolly[4], StableLM [3], Vicuna [9], GPT-J [53] and mostly struggled to find tasks they could do well.

A.3 Counterfactual Dataset Generation

To train Boundless DAS, we need to generate counterfactual datasets where each example contains a pair of inputs as base and source, an intervention indicator (i.e., what variable we are intervening),

---

[3] pytorch.org/docs/stable/generated/torch.nn.utils.parametrizations.orthogonal.html

[4] huggingface.co/databricks/dolly-v2-12b

and a counterfactual label. For instance, if we are training Boundless DAS for our Boundary Check (Left Only) high-level causal model, we may sample the base example with the bracket as [2.50, 7.50] and the input amount of 1.50 dollars, and the source example with the bracket as [3.50, 8.50] and the input amount of 9.50 dollars. If we intervene on the variable whether the input variable is higher than the lower bound, and we swap the variable from the second example into the first example, we will have a counterfactual label as “Yes”. For each high-level causal model, we follow the same steps to construct the counterfactual dataset.

A.4 Interchange Intervention Accuracy

Figure 7 to Figure 10 show breakdown of results presented in Table 1.

A.5 Pseudocode for Boundless DAS

Boundless DAS is a generic framework and can be implemented for models with different architectures. Here we provide a pseudocode snippet for a decoder-only model.

```python
def sigmoid_boundary_mask(population, boundary_x, boundary_y, temperature):
    return torch.sigmoid((population - boundary_x) / temperature) * \
        torch.sigmoid((boundary_y - population) / temperature)

class AlignableDecoderLLM(DecoderLLM):
    # overriding existing forward function
    def forward(hidden_states, source_hiddne_states):
        for idx, decoder_layer in enumerate(self.layers):
            if idx == self.aligning_layer_idx:
                # select aligning token reprs
                aligning_hidden_states = hidden_states[:,start_idx:end_idx]
                # rotate
                rotated_hidden_states = rotate(aligning_hidden_states)
                # interchange intervention with learned boundaries
                boundary_mask = sigmoid_boundary_mask(
                    torch.arange(0, self.hidden_size),
                    self.learned_boundaries,
                    self.temperature
                )
                rotated_hidden_states = (1. - boundary_mask) * rotated_hidden_states +
                    boundary_mask * source_hidden_states
                # unrotate
                hidden_states[:,start_idx:end_idx] = unrotate(rotated_hidden_states)
            hidden_states = self.layers[idx](hidden_states)
        return hidden_states
```

A simplified version of Boundless DAS with pseudocode for a generic decoder-only model.

A.6 Common Questions

In this section, we answer common questions that may be raised while reading this report.

> *Can Boundless DAS work with 175B models?*

Yes, it does. Taking the recently released Bloom-176B as an example, the hidden dimension is 14,336, which results in a rotation matrix containing 205M parameters. This is possible if the model is sharded and the rotation matrix is put into a single GPU for training. This is only possible given the fact that Boundless DAS is now implemented as a token-level alignment tool. If the alignment is for a whole layer (i.e., token representations are concatenated), it is currently impossible. Worth to note that training the rotation matrix consumes more resources than training a simple weight matrix, as the orthogonalization process in `torch` requires matrix exponential requiring to save multiple copies of the matrix.

> *Current paper limits to a set of tasks from the same family. How well will the findings in the paper generalize to other scenarios?*

Existing public models in 7B-30B regime are still not ineffective in solving reasoning puzzles that require multiple steps. We test different reasoning tasks (e.g., math, logic, natural QA, code, etc..) with instruct-tuned LLaMA [49], Dolly[5], StableLM [3], Vicuna [9], GPT-J [53] models that have 6B-30B parameters and is hard to find qualified reasoning tasks. Once larger LLMs are released and evaluated as having stronger reasoning abilities, Boundless DAS will be ready as an analysis tool. Although our Price Tagging game task is simple, we test with different variants of the task and show the efficacy of our method.

Is it possible to apply Boundless DAS over the whole layer representation?

It is possible for smaller models if we are learning a full-rank square rotation matrix. It is impossible for Alpaca now. Let's say we align 10 token representations together, we will have a total dimension size of 40,960. This leads to a rotation matrix containing 1.6B parameters. Given the current orthogonalization process in `torch`, it will require a GPU with memory larger than the current A100 GPU just to fit in a trainable rotation matrix.

Only limited high-level models are tested with Boundless DAS in the experiment.

There is a large number of high-level models to solve the task, and we only test a portion of them. We think it is sufficient to demonstrate the advantages of our pipeline. It would be interesting if humans can be in the loop of testing different hypotheses of the high-level causal model in future work. On the other hand, not all the high-level models are interesting to analyze. For instance, if two high-level models are equivalent (e.g., swapping variable names), it is not in our interest to find alignments.

Can Boundless DAS be applied to find circuits in a model?

Yes, it can. Circuit finding or inductive head searching in any Transformer model often involves using interventions at head level to find the mapping between activations generated on certain heads and high-level concepts. Previous work in this field usually rely on the assumption of localist representation (i.e., there exists a one-to-one mapping between a group of neurons and a high-level concept). This is often too ideal to be real in practice, as multiple concepts can easily be aligned with the same group of neurons or a group of neurons may map to multiple concepts. Boundless DAS actually breaks this assumption by specifically focusing on distributed alignment search. More importantly, Boundless DAS can be easily adapted into a head-wise alignment search where we add a shared rotation matrix on top of head representations of all the tokens.

---

[5] https://huggingface.co/databricks/dolly-v2-12b

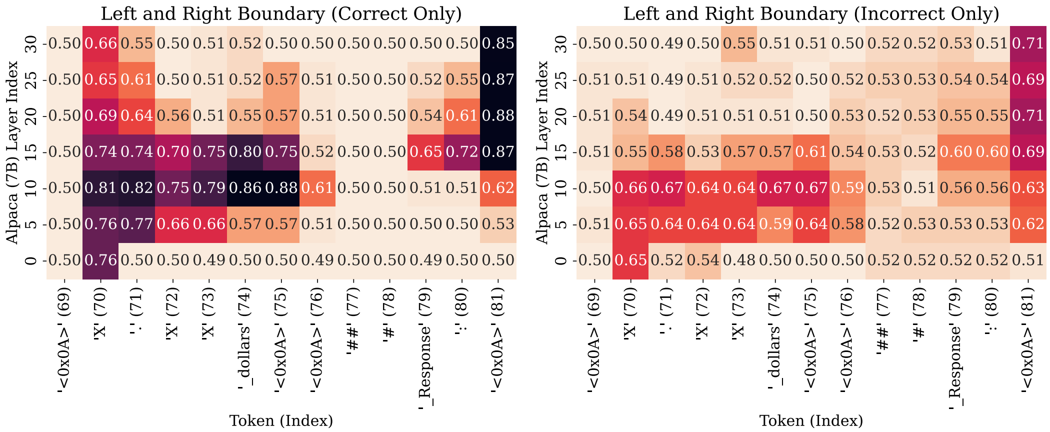

Figure 7: Interchange Intervention Accuracy (IIA) evaluated with correct input examples only as well as incorrect input examples only for our Left and Right Boundary causal model. The higher the number is, the more faithful the alignment is. We color each cell by scaling IIA using the model's task performance as the upper bound and a dummy classifier (predicting the most frequent label) as the lower bound.

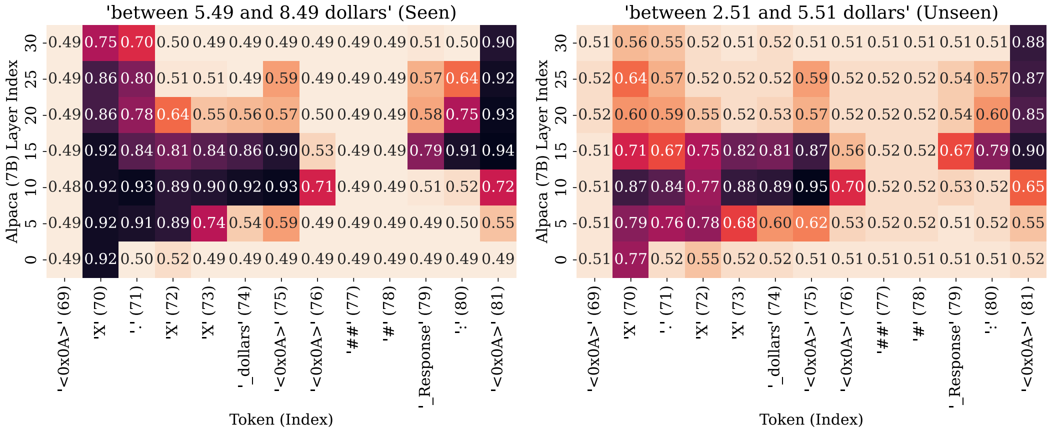

Figure 8: Interchange Intervention Accuracy (IIA) evaluated with different settings of brackets in the instruction. The seen setting is for training, and the unseen setting is for evaluation. The higher the number is, the more faithful the alignment is. We color each cell by scaling IIA using the model's task performance as the upper bound and a dummy classifier (predicting the most frequent label) as the lower bound.

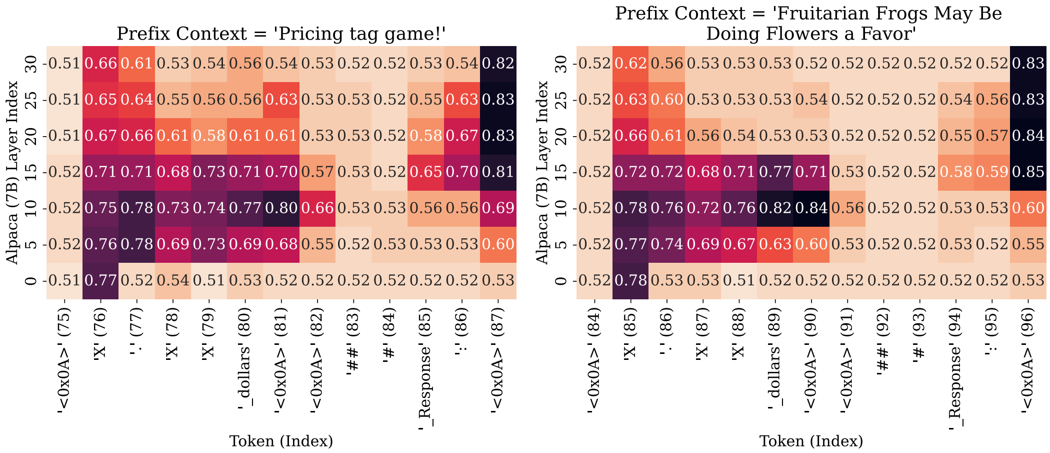

Figure 9: Interchange Intervention Accuracy (IIA) evaluated with different irrelevant contexts inserted as prefixes. The higher the number is, the more faithful the alignment is. We color each cell by scaling IIA using the model's task performance as the upper bound and a dummy classifier (predicting the most frequent label) as the lower bound.

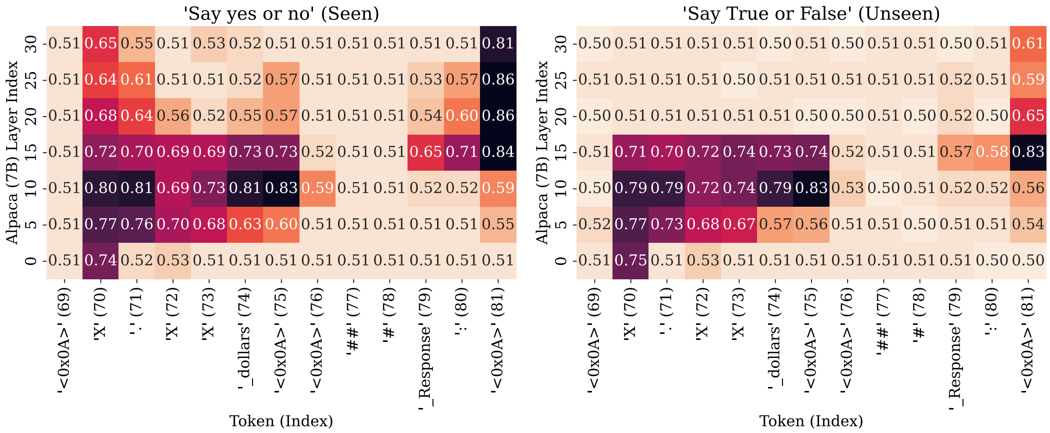

Figure 10: Interchange Intervention Accuracy (IIA) evaluated with different output formats. The seen setting is for training, and the unseen setting is for evaluation. The higher the number is, the more faithful the alignment is. We color each cell by scaling IIA using the model's task performance as the upper bound and a dummy classifier (predicting the most frequent label) as the lower bound.