The appendix includes the following sections:

- Sec A - Contributions
- Sec B - Limitations
- Sec C - More Related Works
- Sec D - Model Implementation Details
- Sec E - Pre-training Details
- Sec F - Tasks and Instruction Tuning
- Sec G - Experiment Details and Additional Results

## A. Contributions

Jiasen Lu, Christopher Clark, Sangho Lee, and Zichen Zhang collectively contributed to dataset construction, prompt development, and conducted numerous exploratory experiments for this project.

**Jiasen Lu** led and designed the main idea and scope of the project. Developed the majority of the model pipeline – image and audio tokenizer, main architecture, model stabilization, and training objective. Led and designed the pre-training and instruction tuning data pipelines. Conducted experiments with various model and data hyperparameters, oversaw the model training process, and wrote the paper. Coordinated with the whole team.

**Christopher Clark** co-led and designed the infrastructure, instruction tuning, and evaluation. Developed the dynamic packing system, modality processing pipeline, and classifier-free guidance for image and audio inference. Added the NLP and many V&L datasets, added many synthetic tasks, and built prompts for instruction-tuning tasks. Ran the evaluation in § 5.1 (NLP), 5.2, and 5.5 (detection, depth) and wrote the paper.

**Sangho Lee** core contribution to the pre-training data pipeline. Added all large-scale multimodal pretraining datasets, and video and audio instruction tuning datasets. Developed sample construction pipeline for pre-training. Helped with the model implementation – position encoding, perceiver resamplers, and model stabilization. Ran the evaluation in § 5.1 (audio), 5.3 (audio and image FID), 5.5 (video and audio understanding) and wrote parts of the paper.

**Zichen Zhang** core contribution to the instruction tuning data pipeline. Added many V&L, embodiment, video, audio, data augmentation, and all instruction tuning datasets. Built prompts for instruction tuning. Investigated the model architectures and training pipelines and stabilized the training. Ran the experiments in § 5.1 (image TIFA, SeedBench), 5.3 (image TIFA, action), 5.4, wrote parts of the paper, developed the model demo and project page.

**Savya Khosla** added 3D object detection, optical flow, and multi-point tracking datasets, ran the evaluation of 3D detection, and initiated the demo.

**Ryan Marten** added part of video and tracking datasets.

**Derek Hoiem** advised on the research direction.

**Aniruddha Kembhavi** advised on the research direction and evaluation, helped manage compute resources and wrote the paper.

## B. Limitation

- Due to memory constraints, we use the base versions of the ViT and AST models for image and audio features throughout the project. Using a larger version of these image and audio encoders could substantially improve performance.
- While our image generation is more faithful compared to SD-based methods, its quality doesn't match that of the stable diffusion model. Additionally, our audio generation is capped at approximately 4 seconds, which restricts the practical application of the audio outputs.
- Limited computational resources constrained our exploration of the model's hyperparameters. It's likely that using a significantly larger batch size could enhance the model's performance.
- Our model is much less reliable for modalities like depth, and video or when requiring more niche abilities like 3D object detection, *etc*. This is probably due to the limited variety of tasks we have in these areas.
- Improving the quality of our data could enhance the model's performance. However, despite considerable efforts, our human-written prompts still fall short in diversity. We notice a notable decrease in the model's performance when dealing with new instruction tasks, as opposed to those it was trained on.

## C. More Related Work

Overall, this shows a strong trend towards expanding the number of supported tasks and modalities. UNIFIED-IO 2 pushes this trend to its limit, including the capabilities of these prior works with few exceptions and the ability to generate outputs in more modalities. Recently, CoDi [174] also achieved similar any-to-any generation capabilities by using multiple independently trained diffusion models and aligning their embedding spaces. UNIFIED-IO 2 has stronger language abilities and can perform well on many more tasks.

A notable feature of UNIFIED-IO 2 is that the model is trained from scratch instead of being initialized with a pre-trained LLM. Prior works [114, 186, 188, 192] following this approach are typically not designed to produce complex generations like free-form text responses, images or sounds, or follow text instructions. Compared to recent general-purpose multimodals models [81, 143, 210], UNIFIED-IO 2 has a significantly broader scope of tasks and outputs. Training from scratch means that the method can be reproduced without a costly preliminary stage of language model pre-training and is a more natural fit for how

humans learn modalities simultaneously through their co-occurrences, not one at a time.

## D. Model Implementation Details

In this section, we present the implementation details of our model.

### D.1. Detailed of Unified Task Representation

First, we provide details about how different modalities are represented in our model.

**Text representation.** The Byte Pair Encoding (BPE) vocabulary size is 32000. Similar to [147], we add 200 additional special tokens to indicated masked spans when denoising. We further add 10 special tokens that can be used to reference the image, audio, and history input in the text. Two special tokens are to indicate the ⟨Image_Input⟩ and ⟨Audio_Input⟩, and 8 special tokens represent individual elements in the image and audio history inputs, both of which have a maximum of 4 frames. We use a maximum of 512 input and output tokens.

**Sparse structures representation.** We use an additional 1000 special tokens to represent all continuous values, such as points, boxes, camera transformation, and 3D cuboids. Points are represented with $[y, x]$ coordinates and boxes with $[y_1, x_1, y_2, x_2]$ coordinates with values normalized by the image size. Camera transformations are represented as polar angle $\theta$, azimuth angle $\phi$, and distance $r$. 1000 special tokens to represent discretized angle from $-\pi$ to $\pi$. Following [16], 3D cuboids are represented with 12 parameters including projected center $[u, v]$, virtual depth $z$, log-normalized box dimension $[\bar{w}, \bar{h}, \bar{l}]$, and continuous allocentric rotation $\boldsymbol{p}$.

- $[u, v]$ represent the projected 3D center on the image plane relative to the 2D RoI
- $z \in \mathbb{R}_+$ is the object's center depth in meters.
- $[\bar{w}, \bar{h}, \bar{l}] \in \mathbb{R}_+$ are the log-normalized physical box dimensions in meters.
- $\boldsymbol{p} \in \mathbb{R}^6$ is the continuous 6D allocentric rotation.

For 3D cuboid detection, we use prompts to indicate the target format, such as "*Locate all objects in 3D using projected 3D center, virtual depth, log-normalized box size, and rotation in the image.*"

**Action representation.** For embodied navigation tasks, the discrete action space is directly represented as texts, *e.g.* "`forward`", "`left`", "`right`", "`stop`". For object manipulation tasks, the action is represented differently based on the robots. Overall, the positional change (*e.g.* ($\Delta$`PosX`, $\Delta$`PosY`, $\Delta$`PosZ`)), rotational change (*e.g.* ($\Delta$`RotX`, $\Delta$`RotY`, $\Delta$`RotZ`)), and gripper open or close are discretized using the same 1000 special tokens, and we use the text prompt to indicate the input and target format. For tasks that require multi-step planning (*e.g.* VIMA [87]), the actions are represented as human-readable texts with the indication of steps, skills used (*e.g.* pick, place, or push), and discretized positional and rotational parameters. Figure 9 provides a detailed illustration of the robot tasks.

| | | |
|---|---|---|
| Audio Input | Sample rate | 16000 Hz |
| | FFT hop length | 256 samples |
| | FFT window size | 1024 |
| | Mel bins | 128 |
| | Subsegment length | 256 hops, (≈4.08 sec) |
| | Mel Spectrogram size | 128 mels × 256 hops |
| | fmin | 0 |
| | fmax | 8000 |
| | AST patch size | 16 |
| | token size | 8 × 16 |
| | Pretrain sub-sample | 64 |
| | Final size | 64 or 128 tokens |
| Image Input | ViT patch size | 16 |
| | Pretraining size | 384 × 384 |
| | Token size | 24 × 24 |
| | Pretrain sub-sample | 288 |
| | Final size | 288 or 576 tokens |
| Text | Seq length | 512 |
| | Final size | 512 tokens |
| Image History | ViT patch size | 16 |
| | Pretraining size | 256 × 256 |
| | Token size | 16 × 16 |
| | Pretrain sub-sample | 128 |
| | Max num segments | 4 |
| | Latent size | 32 |
| | Final size | 32, 64, 96, 128 tokens |
| Audio History | AST patch size | 16 |
| | Pretraining size | 128 × 256 |
| | Token size | 8 × 16 |
| | Pretrain sub-sample | 64 |
| | Max num segments | 4 |
| | Latent size | 16 |
| | Final size | 16, 32, 48, 64 tokens |

Table 7. Input representations details

**Images representation.** Images are encoded with a pre-trained ViT [45]. We use the ViT-B checkpoint trained on LAION 2B dataset[2]. For image inputs, we use a maximum length of 576 tokens (*i.e.* 24 × 24 patch encoding from a 384 × 384 image). We concatenate features from the second and second-last layers of the ViT to capture both low and high-level visual information. To generate the image, we encode these images as discrete tokens [49]. Different from UNIFIED-IO [123], which uses the VQ-GAN trained on ImageNet [41] to convert 256×256 resolution image into 16 × 16 tokens, we use the VQ-GAN trained on the Open

[2] https://github.com/mlfoundations/open_clip

Images dataset [103] with a compression ratio of 8 and a vocabulary size of 16384[3]. This converts $256 \times 256$ resolution image into $32 \times 32$ tokens. We also compare the VQ-GAN tokenizer with the ViT-VQGAN [208] and MoVQ [223]. We empirically find VQ-GAN leads to best generation results.

**Dense structures representation.** To handle this modality, we convert per-pixel labels into RGB images. For depth, we construct a grayscale image by normalizing the depth map. For surface normal estimation, we convert the $x/y/z$ orientations into $r/g/b$ values. For segmentation, we train UNIFIED-IO 2 to predict a single black-and-white mask for a particular object specified by a class and a bounding box. Instance segmentation (as done in GRIT [66]) can then be performed by first performing localization for the target class and then performing segmentation for each detected box. UNIFIED-IO instead trains the model to produce an image with a randomly selected color for each instance. We found this makes post-processing difficult since output images sometimes do not exactly follow the color scheme, and the model could struggle with images with many different instances.

**Audio representation.** This modality encodes a 4.08-second segment of audio. We take the waveform sampled at 16000 Hz and convert it to a log-mel-scaled spectrogram. We compute the spectrogram for an entire audio segment (4.08 seconds) simultaneously. Each window involves 1024 samples and 256 samples 'hops' between windows. The resulting spectrogram has a size of 128 mel bins with 256 windows. We chose these hyperparameters largely around efficiency. We then encode this with a pre-trained AST [57] with the patch size of $16 \times 16$, hence a total of 128 tokens.

To generate audio, we use ViT-VQGAN [208] to convert the spectrograms into discrete tokens. Since the authors of [208] did not release the source code or any pre-trained models, we implement and train our own version of ViT-VQGAN with $8 \times 8$ patch size that encodes a $256 \times 128$ spectrogram into 512 tokens with a codebook size of 8196. The model is trained with the audio on AudioSet [54], ACAV100M [105], and YT-Temporal-1B [215] datasets. After getting the log-mel-scaled spectrograms, we use HiFi-GAN[4] [98] vocoder to decode the spectrograms back to waveforms. We train the HiFi-GAN using the same parameters shown in Table 7. We trained the model on a mixture of AudioSet and LJSpeech [85] to cover natural sound and human voice.

**History representation.** Images and audio inputs in this history are first encoded in the same way as image and audio inputs. We then use a perceiver resampler [5] to further compress the image and audio features and produce a fixed number of visual outputs (32) and audio outputs (16) to reduce the total sequence length of the model. As shown in Table 7, we consider a maximum of 4 images and audio segments. In our experiments, we test with two different variants of perceiver implementations: 1) a small group of latent embeddings query each frame/segment individually [5, 9], 2) a large group of latent embeddings query all history at once. While the second implementation can finely represent the referenced image and audio, the first can preserve better temporal information. Thus, our final implementation uses the first one.

[3] https://github.com/CompVis/taming-transformers

[4] https://github.com/jik876/hifi-gan

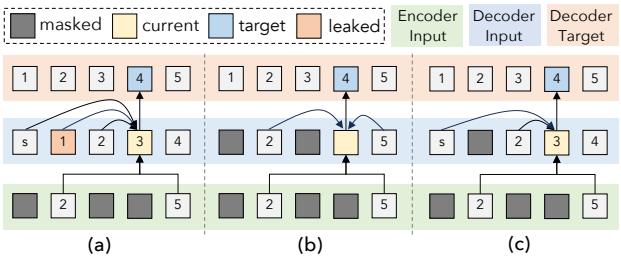



Figure 3. Different training paradigms in masked image modeling: (a) autoregressive, (b) mask auto-encoder, (c) autoregressive with dynamic masking. Our proposed paradigms can maintain causal generation while avoiding information leaks in the decoder.

## D.2. 2D Rotary Embedding

We use a rotary position encoding to model the relative location of input sequences [169]. We chose this primarily because we did not want to use absolute (additive) position embeddings, which would have to be added to the inputs of each encoder, and also wanted to be consistent with the LLaMA [177] position encoding.

The rotary encoding uses no parameters and instead uses a kernel trick to allow the model to recover relative distances between key and query elements in a transformer's attention head. For text, we apply rotary encoding at each layer of the network. For other modalities, we extend RoPE to two-dimensional cases by splitting each of the query and key embeddings of transformer attention heads in half and apply separate rotary embeddings constructed by each of the two coordinates to the halves.

We treat each token (image, audio, image history, and audio history) as having a 2-dimensional position corresponding to 1) $h, w$ coordinates in the image or audio spectrogram, 2) $(t, l)$ where $t$ and $l$ represent the indices of frame and perceiver latent vector in the image or audio history, respectively. Different from [215], which uses a 4-dimensional position to represent all the inputs, we use a combination of learnable segment (modality) embeddings and rotary encoding.

| | | L | XL | XXL |
|---|---|---|---|---|
| Transformer | Params | 1.1B | 3.2B | 6.8B |
| | Vocab size | | 33280 | |
| | Image vocab size | | 16512 | |
| | Audio vocab size | | 8320 | |
| | Model dims | 1024 | 2048 | 3072 |
| | MLP dims | 2816 | 5120 | 8192 |
| | encoder layer | | 24 | |
| | decoder layer | | 24 | |
| | Heads | 16 | 16 | 24 |
| | MLP activations | | silu, linear | |
| | Logits_via_embedding | | True | |
| | Dropout | | 0 | |
| Image Resampler | Latents size | | 32 | |
| | Model dims | 768 | 1024 | 1024 |
| | Heads | 12 | 16 | 16 |
| | Head Dims | | 64 | |
| | Number layer | | 2 | |
| | MLP Dims | 2048 | 4096 | 4096 |
| | MLP activations | | gelu | |
| Audio Resampler | Latents size | | 16 | |
| | Model dims | 768 | 1024 | 1024 |
| | Heads | 12 | 16 | 16 |
| | Head Dims | | 64 | |
| | Number layer | | 2 | |
| | MLP Dims | 2048 | 4096 | 4096 |
| | MLP activations | | gelu | |
| ViT | Patch size | | 16 | |
| | Model dims | | 768 | |
| | Heads | | 12 | |
| | Head Dims | | 64 | |
| | Number layer | | 11 | |
| | MLP Dims | | 3072 | |
| | MLP activations | | gelu | |
| AST | Patch size | | 16 | |
| | Model dims | | 768 | |
| | Heads | | 12 | |
| | Head Dims | | 64 | |
| | Number layer | | 11 | |
| | MLP Dims | | 2048 | |
| | MLP activations | | gelu | |

Table 8. Model Hyperparameters

### D.3. Autoregressive with Dynamic Masking

One problem with image and audio masked denoising in an autoregressive manner is an information leak on the decoder side; see Figure 3 (a). The current decoder's input token (3) is conditioned on enocoder's information (2, 5) and all previous tokens (s $\rightarrow$ 2) to predict target (4). As a result, the predicted token will be conditioned on 1 even though it was masked in the encoder since it appears in the decoder, which will simplify the task and harm representation learning. Simply masking the token in the decoder, as shown in Figure 3 (b), avoids this information leakage but causes the generation and de-noising tasks to interfere with one another. For example, we found that joint training with generation (50% MAE and 50% causal modeling) significantly reduced image generation performance. Our solution is to mask the token in the decoder except when predicting that token, as shown in Figure 3 (c), which does not interfere with causal prediction whilst mostly eliminating data leakage. For image and audio generation, we also use row, column, and conv-shaped masked sparse attention [148] in the decoder.

### D.4. Dynamic Packing

Here, we describe the dynamic packing algorithm in more detail. As is standard practice, when batching together inputs, we pad input tensors to a maximum length and use attention masked to prevent the transformer from attending to padding elements. This, however, is highly inefficient in our multi-modal setting because many modalities are not present in most examples, which results in a huge amount of padding. For example, if one example in the batch has an image output, every other example must be padded with 1024 target image tokens, even if their output is in a different modality.

One solution is to arrange batches so that each batch contains examples with similar numbers of tokens in each modality. This is, however, complicated to do in practice since (1) our data does not fit in RAM, so we cannot easily sort and group data this way, especially if needing to match tokens across five input and three output modalities and (2) our coding framework, JAX [15], does not support variable length tensors when constructing the execution graph which makes handling variable lengths between batches extremely difficult.

Instead, we use *packing*, a process where the tokens of multiple examples are packed into a single sequence, and the attentions are masked to prevent the transformer from cross-attending between examples. Packing is often done as a pre-processing step when handling text, but this does not work in our setup since some parts of our network cannot operate on packed data (*e.g.*, the VAE or image ViT). Instead, we start with an unpacked batch of examples, run these components first, and then dynamically pack the resulting tokens in a backdrop-compatible way before running the transformer. To run efficiently on TPUs we pack examples using matrix multiplication with carefully constructed one-hot matrices.

To account for all modalities, the maximum sequence length our transformer needs to take as input is 1152, and the maximum target length is 2048. When packing, we can generally pack two examples into an input sequence of 864 and a target sequence of 1280, which gives a roughly 4x

speed up due to reduced sequence length and the ability to process two examples simultaneously. When streaming data, packing cannot be done reliably. For example, if two consecutive examples have an image output, they cannot be packed since they will total over 1280 output tokens. To handle this, we use a heuristic algorithm to re-arrange data as it is being streamed. The algorithm keeps a small pool of examples in memory. Given a new example, it pairs it with the largest example in the pool it can be packed with and outputs both as a pair. If no such example exists, it adds the example to the pool. If the pool reaches a maximum size of 10, the largest example is emitted and processed without being packed with another example. We find this occurs less than 0.1% of the time during training.

### D.5. Full Model Details

In Table 8, we present the full hyperparameters of our model. During pre-training, we train the UIO-$2_{\text{L}}$, UIO-$2_{\text{XL}}$, and UIO-$2_{\text{XXL}}$ with a batch size of 512 due to memory limit. We sub-sample 50% of the image, audio, and history inputs patches. The total packing length is 864 for the encoder and 1280 for the decoder. During instruction tuning, we train all of our models with a batch size 256 due to computing constraints. We sub-sample 87.5% of the image, audio, and history input patches. The total packing length is 1024 for pretraining and 1280 for instruction tuning. 8-way in-layer parallelism and 64-way data parallelism were used to scale up to the 7B model training.

We pre-train and fine-tune for 1.5 million steps with an effective batch size of 512. This results in training on approximately 1 trillion tokens in each stage. During pre-training, we keep at most 50% of the image patches in the image history or image encoder, as is common practice with MAE pre-training [71]. We use up to four images/segments in image/audio history.

### D.6. Optimizer

We use Adafactor [164] as our optimizer with a linear warm-up for the first 5,000 steps and a learning rate decay of $1/\sqrt{k}$. We train with $\beta_1 = 0.9$ and $\beta_2 = 1.0 - k^{-0.8}$, where $k$ is the step number. We use global norm gradient clipping with a threshold of 1.0 and find that this is crucial to stabilized training. Table 1 gives the details of our different models. For all models, we train 3.0M steps – 1.5M for pre-training and 1.5M for instruction tuning, respectively. More details in Appendix D.5.

### D.7. Training Stability

An example of training instability as more modalities are added is shown in Figure 5. As shown in (a) and (b), training only on image generation (green curve) results in stable loss and gradient norm convergence. Introducing a combination of image and text tasks (orange curve) slightly increases the gradient norm compared to a single modality, but remains stable. However, the subsequent inclusion of the video modality (blue curve) leads to an unrestrained escalation in the gradient norm. When an XXL version of this model is trained on all modalities, as shown in Figure 5 (c) and (d), the loss explodes after 350k steps, and the next token prediction accuracy significantly drops at 400k steps. Figure 6 shows that the pre-training loss for our model is stable despite the heterogeneity of input and output modalities.

## E. Pre-Training Details

In this section, we provide additional details about the data UNIFIED-IO 2 is pre-trained on. The datasets we use for pre-training are listed in Table 9. Unless otherwise specified, we use the pre-training objective described in Section 3.3, where one of the present modalities is randomly selected as the target. We sample data to ensure all the output modalities are well represented and to balance how often our various corpora are used based on their size. The distribution is shown in Figure 4.

### E.1. Data Sources

**Text.** Our data follows the mixture used by MPT-7B [176].
**Image & Text.** Image & text paired data comes from various unsupervised corpora, shown in Table 9. For LAION data, we only generate images from image/text pairs from LAION aesthetic, which contains higher quality images, while we generate text for image/text pairs from LAION 400M. We also only keep images from LAION if they are marked as being unlikely to be NSFW in the LAION metadata. Web images is a dataset of images we download and focuses on icons and stylized images.
**Video.** We gather a total of 180M short videos from various sources. During training, we pick a random sequence of up to five frames from the video. The first four will be encoded with an image/audio history encoder, while the fifth frame will be encoded with the image/audio encoder. The text matching these frames is encoded with a text encoder along with marker tokens to show where each frame occurred as stated in D.1, or, if the dataset only includes a single caption that is not aligned with individual frames, the entire caption is encoded instead. The text, audio, or image modality can be selected as the target modality. As usual, other modalities are randomly masked, and the target modality is randomly masked or injected with noise in the input. Note we have sub-sampled data from many of these corpora to keep the dataset size more manageable, and sometimes due to broken video links.
**Interleaved Image & Text.** We primarily use OBELICS [104], which contains paragraphs and images interleaved together. For each document, we randomly select an image or a paragraph as the target and use up to

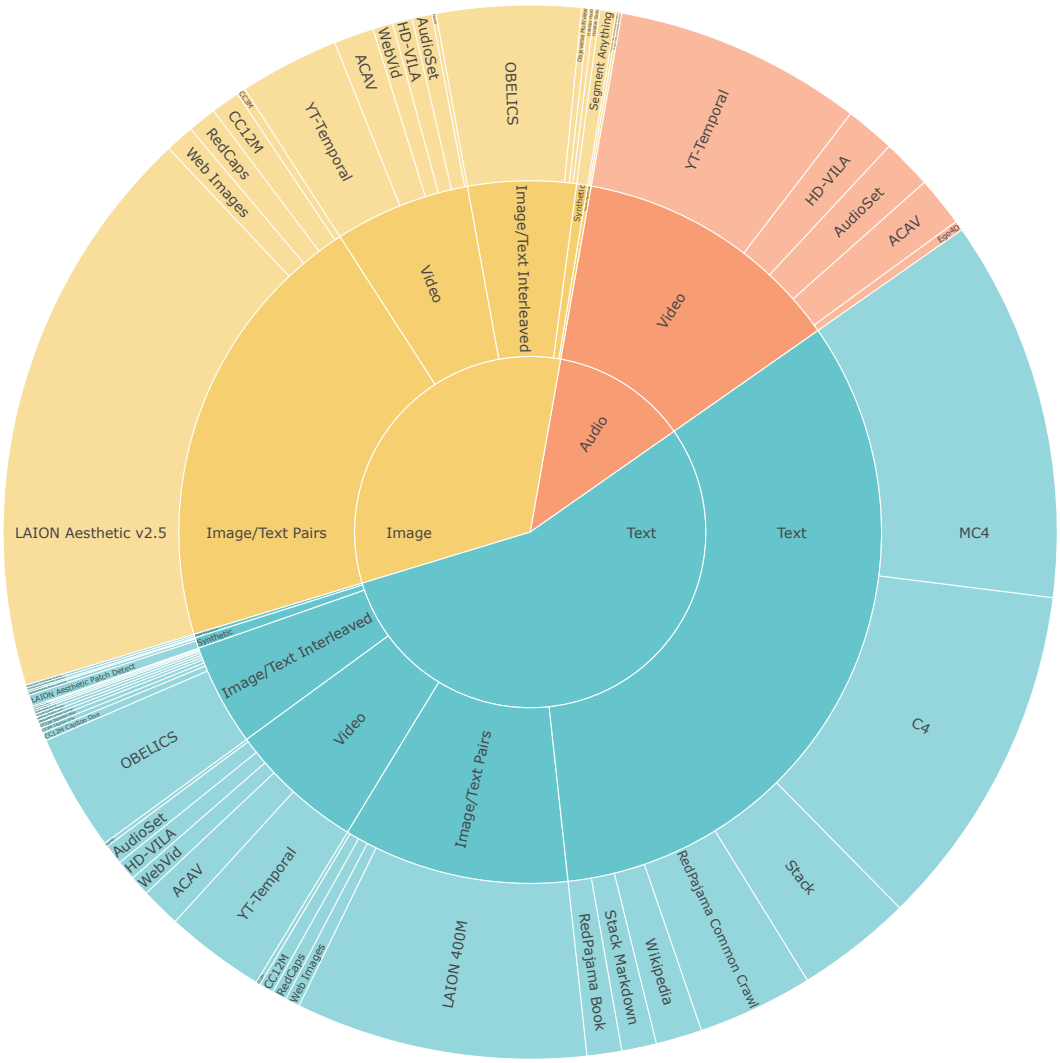



Figure 4. Pre-training data distribution, segments proportional to sampling rates. The inner section shows the target modality, the middle section shows the type of data, and the third shows particular datasets.

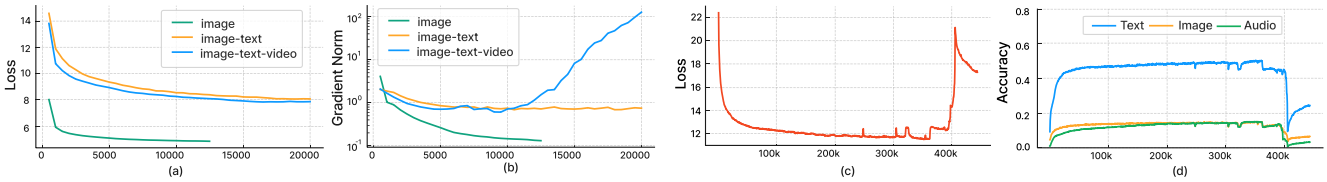



Figure 5. **Left**: Training loss (a) and gradient norms (b) on different modality mixtures. **Right**: Training loss (c) and next token prediction accuracy (d) of UIO-$2_{\text{XXL}}$on all modalities. Results were obtained before applying the proposed architectural improvements.

the previous four (if the target is an image) or five (if the target is a paragraph) images as context. The last image is encoded with the image encoder, and the remaining images are encoded in the image history. The text matching those images is concatenated and interjected with marker tokens to indicate where the images in the image history or image input occur. We either do de-noising, where a noisy version of the target is included in the input, or generation, where the target is not part of the input, although we always include both the text and image input modalities.

In addition, we construct interleaved data by interleaving multiple images and captions from several image/text pair corpora. The images are encoded as the image input and/or the image history, and matching text is constructed

| | Size | Rate | Text | Sparse | Dense | Image | Audio | ImageH | AudioH | Text | Sparse | Dense | Image | Audio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Text** | 6.6b | 33.0 | ✓ | - | - | - | - | - | - | ✓ | - | - | - | - |
| MC4 [201] | 5.0b | 11.7 | ✓ | - | - | - | - | - | - | ✓ | - | - | - | - |
| C4 [68] | 266m | 10.6 | ✓ | - | - | - | - | - | - | ✓ | - | - | - | - |
| Stack [95] | 147m | 3.55 | ✓ | - | - | - | - | - | - | ✓ | - | - | - | - |
| RedPajama CC [32] | 1.2b | 3.55 | ✓ | - | - | - | - | - | - | ✓ | - | - | - | - |
| Wikipedia | 6.8m | 1.42 | ✓ | - | - | - | - | - | - | ✓ | - | - | - | - |
| RedPajama Book [32] | 13m | 1.06 | ✓ | - | - | - | - | - | - | ✓ | - | - | - | - |
| Stack-Markdown [95] | 34m | 1.06 | ✓ | - | - | - | - | - | - | ✓ | - | - | - | - |
| **Image/Text** | 970m | 31.3 | ✓ | - | - | ✓ | - | - | - | ✓ | - | - | ✓ | - |
| LAION Aesthetics v2.5 [158] | 491m | 17.7 | ✓ | - | - | ✓ | - | - | - | - | - | - | ✓ | - |
| LAION-400M [159] | 346m | 8.95 | ✓ | - | - | ✓ | - | - | - | ✓ | - | - | - | - |
| CC12M [23] | 11m | 1.48 | ✓ | - | - | ✓ | - | - | - | ✓ | - | - | ✓ | - |
| RedCaps [42] | 12m | 1.39 | ✓ | - | - | ✓ | - | - | - | ✓ | - | - | ✓ | - |
| Web Images | 107m | 1.33 | ✓ | - | - | ✓ | - | - | - | ✓ | - | - | ✓ | - |
| CC3M [163] | 3.0m | 0.49 | ✓ | - | - | ✓ | - | - | - | ✓ | - | - | ✓ | - |
| **Video** | 181m | 25.0 | ✓ | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | ✓ |
| YT-Temporal [215] | 146m | 13.7 | ✓ | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | ✓ |
| ACAV [105] | 17m | 3.98 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | ✓ | ✓ |
| HD-VILA [200] | 7.1m | 2.75 | ✓ | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | ✓ |
| AudioSet [54] | 1.7m | 2.75 | ✓ | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | ✓ |
| WebVid [13] | 9.2m | 1.23 | ✓ | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | - |
| Ego4D [60] | 0.7m | 0.55 | ✓ | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | ✓ |
| **Interleaved Image/Text** | 157m | 8.70 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | ✓ | - |
| OBELICS [104] | 131m | 8.00 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | ✓ | - |
| CC12M Interleaved | 11m | 0.35 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | - | - |
| CC3M Interleaved | 3.0m | 0.21 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | - | - |
| RedCaps Interleaved | 12m | 0.14 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | - | - |
| **Multi-View** | 3.4m | 0.67 | ✓ | - | - | ✓ | - | ✓ | - | - | ✓ | - | ✓ | - |
| CroCo Habitat [157, 194] | 2.6m | 0.33 | ✓ | - | - | ✓ | - | ✓ | - | - | - | - | ✓ | - |
| Objaverse [40] | 0.8m | 0.33 | ✓ | - | - | ✓ | - | ✓ | - | - | ✓ | - | ✓ | - |
| **Agent Trajectories** | 1.3m | 0.33 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | ✓ | - |
| ProcTHOR [39] | 0.7m | 0.17 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | ✓ | - |
| Habitat [157] | 0.6m | 0.17 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | ✓ | - |
| **Synthetic** | 504m | 1.00 | ✓ | ✓ | - | ✓ | - | - | - | - | ✓ | ✓ | - | - |
| Segment Anything [94] | 1.1m | 0.50 | ✓ | ✓ | - | ✓ | - | - | - | - | - | ✓ | - | - |
| Laion Aesthetics Patches | 491m | 0.45 | ✓ | - | - | ✓ | - | - | - | - | ✓ | - | - | - |
| RedCaps Patches | 12m | 0.05 | ✓ | - | - | ✓ | - | - | - | - | ✓ | - | - | - |
| **All** | 8.5b | 100 | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 9. Datasets used for pre-training, rate shows the sampling percentage during pre-training and size shows the approximate number of examples if iterating through the data once.

by specifying the caption for one, or all, of these images using special tokens to mark which image each caption refers to. For this task, we only target the text modality, and train the model to either (1) de-noise the caption of a single image, (2) generate a caption for a single image that is specified in an input prompt using a marker token or (3) generate a sequence of marker tokens and captions that describe each input image. This task aims to ensure the model learns the semantics of the images in the history and understands the marker tokens.

**Multi-View.** We train on the cross-view completion task from CroCo [194], where the model must complete a heavily noised image using an image of the same scene, but from a slightly different angle, as context. The noised input is encoded as an image and the second image is encoded through the image history encoder. In addition, we generate data using Objaverse [40] objects by capturing multiple views of the object in 3D, and either specify the camera coordinates in the input text and train the model to generate a new image matching new camera coordinates, or train the model to pre-

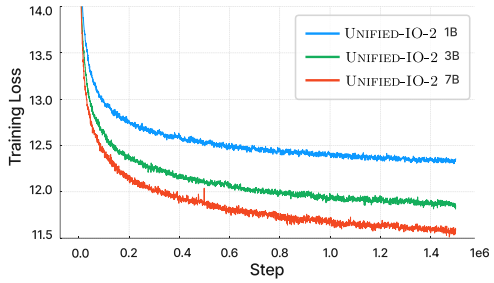



Figure 6. Training loss curves for the three models, which are pretrained with dynamic packing and a batch size of 512.

dict how the camera has moved between different images. We further augment the view synthesis task by providing in-context examples. For example, by giving one or more examples of the views and transformations in the image history, the model predicts the new view from the new camera transformation specified by the prompt. Both tasks aim to improve the model's 3D understanding during pre-training.

**Agent Trajectory.** We use scripted shortest path trajectories in ProcTHOR [39] and human-collected demonstrations in Habitat [149, 157]. While the original datasets are for object navigation with relatively long episode lengths, we only subsample from the last few frames for image history and image input such that mostly the target object is within the observation. The task is randomly selected from 1) generating the next visual observation frame as the target image, 2) predicting the next positional observation coordinates as the text target, and 3) predicting the next action as the text target. 1) requires inferring from the image and image history input and the last action specified in the text input, 2) further requires the location information, and 3) is based on the target object name and visual observations for the next action prediction.

**Synthetic.** We add two synthetic tasks. First, we use the automatically annotated data from Segment Anything [94]. We give the model either a set of points or a bounding box as input and train it to generate a segmentation mask as output. Second, we add artificial patches of various shapes and colors to images from other unsupervised datasets and train the model to output their locations in order to train the model to generate sparse coordinates as output. We additionally train the model to output the total number of patches on the image to pre-train its counting abilities.

### E.2. Pretraining Objective Example

Figure 8 shows an example of our unsupervised pretraining objective using a video that contains a sequence of image frames, the corresponding audio, and a text transcript. The pre-training sample is constructed by following the procedure: 1. select the target modality; 2. select which other input modalities to keep; 3. select the objective; 4. generate the random input mask depending on the task of denoising or generation; 5. add a prefix token indicating the task.

## F. Instruction Tuning Details

In this section, we provide additional details about the instruction tuning data and individual tasks UNIFIED-IO 2 supports. An overview of the instruction tuning data is shown in Table 10. We show a visualization including individual datasets in Figure 7. We sample broad categories of tasks evenly and then generally sample individual datasets in proportion to the square root of their size, although with some minor hand-engineered adjustments to downweight noisy datasets or upweight very rare tasks.

**Natural Language [25.0%].** For natural language, we use the mixture from FlanV2 [122] and various other instruction following datasets [33, 142]. In addition, we continue pre-training on our unsupervised NLP mixture to help prevent the model from forgetting information learned from pre-training during the extensive instruction tuning stage.

**Image Generation [17.6%].** For text-to-image generation, we use the same image & text pairs we used during pre-training. We also include data from [102, 103, 115] that provide better caption quality. We additionally train the model to generate images through view synthesis [40, 194], image editing [18, 217], segmentation-based image generation [123] and inpainting [123].

**Audio Generation [7.5%].** This includes text-to-audio datasets with audio in the wild [47, 93, 131], music [2], and human speech [85]. We also add pre-training data with the task of predicting the next audio clip in a video. More specifically, we divide the audio into segments and then generate one of them given both the text and previous segments as input.

**Image Understanding [17.8%].** We include various data sources from visual question answering [6], image tagging [41], region classification [102], and datasets with open-ended chat-like responses [119, 220]. We also include the multimodal instruction tuning datasets $M^3$IT [112] and MIMIC-IT [107].

**Video Understanding [10.6%].** We include data sources from video captioning [190, 199], video tagging [35, 111, 168], and video question answering [196, 198]. We also use examples from $M^3$IT [112] and MIMIC-IT [107] for video instruction following.

**Audio Understanding [2.5%].** We include data sources from audio tagging [24, 54], and audio captioning [47, 93]. We also include data from video action classification [7] with audio in the dataset.

**Image Sparse Labelling [7.25%].** These tasks require outputting sparse coordinates based on an input image. We mainly consider object detection [115], referring expression [91], 3D detection [16], camera pose prediction [40], text detection [183] and human keypoints [115].


Figure 7. Sunburst chart of our instruction-tuning mixtures, sections are proportional to sampling rates.

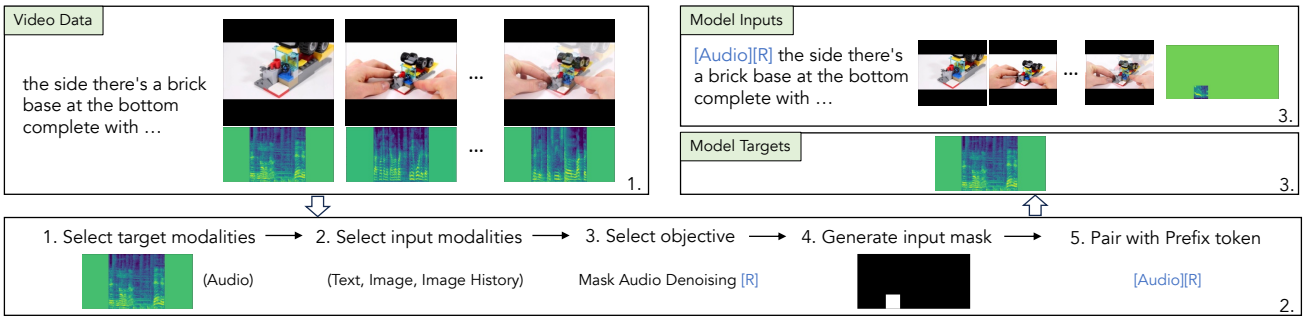



Figure 8. Construction of training samples from video data for the model's input and target. Given the video, we first extract the video frames and the corresponding audio spectrograms and transcript. Then, the data passes through a random selection process to determine the target modality, input modalities, training objective, input mask *etc*. The model's final input and target are shown in the top right.

**Image Dense Labelling [4.06%].** We do several image labeling tasks, including surface normal estimation [78, 204], depth estimation [138], and optical flow [21, 44]. We also train our models on various segmentation tasks, including

| | Size | Rate | Datasets | Text | Sparse | Dense | Image | Audio | ImageH | AudioH | Text | Sparse | Dense | Image | Audio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Image Generation** | 506m | 17.6 | 21 | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ | ✓ | - | - | ✓ | - |
| Image from Text | 497m | 10.6 | 5 | ✓ | - | - | - | - | - | - | - | - | - | ✓ | - |
| Controllable Image Editing | 3.0m | 2.92 | 4 | ✓ | - | ✓ | ✓ | - | ✓ | - | - | - | - | ✓ | - |
| Image Editing | 1.1m | 1.66 | 3 | ✓ | - | - | ✓ | - | - | - | - | - | - | ✓ | - |
| Next Frame Generation | 24k | 0.96 | 2 | ✓ | ✓ | - | - | - | ✓ | ✓ | - | - | - | ✓ | - |
| Image Inpainting | 1.0m | 0.79 | 3 | ✓ | ✓ | - | ✓ | - | - | - | - | - | - | ✓ | - |
| View Synthesis | 4.2m | 0.60 | 4 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | ✓ | - |
| **Audio Generation** | 164m | 7.50 | 9 | ✓ | - | - | ✓ | ✓ | ✓ | ✓ | - | - | - | - | ✓ |
| Audio from Text | 19m | 5.62 | 8 | ✓ | - | - | - | - | - | ✓ | - | - | - | - | ✓ |
| Audio from Video | 145m | 1.88 | 1 | ✓ | - | - | ✓ | ✓ | ✓ | ✓ | - | - | - | - | ✓ |
| **Image Understanding** | 53m | 17.8 | 73 | ✓ | ✓ | - | ✓ | - | ✓ | - | ✓ | - | - | - | - |
| VQA | 5.8m | 6.23 | 31 | ✓ | - | - | ✓ | - | - | - | ✓ | - | - | - | - |
| Image Captioning | 32m | 4.25 | 14 | ✓ | - | - | ✓ | - | - | - | ✓ | - | - | - | - |
| Region Classification | 6.1m | 2.41 | 4 | ✓ | ✓ | - | ✓ | - | - | - | ✓ | - | - | - | - |
| Image Tagging | 3.8m | 2.38 | 8 | ✓ | - | - | ✓ | - | - | - | ✓ | - | - | - | - |
| Relationship Prediction | 0.8m | 1.41 | 6 | ✓ | ✓ | - | ✓ | - | - | - | ✓ | - | - | - | - |
| Region Captioning | 3.5m | 0.60 | 1 | ✓ | ✓ | - | ✓ | - | - | - | ✓ | - | - | - | - |
| Image Instruction Following | 0.4m | 0.37 | 6 | ✓ | - | - | ✓ | - | - | - | ✓ | - | - | - | - |
| Image Pair QA | 0.1m | 0.17 | 3 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | - | - |
| **Image Sparse Labelling** | 13m | 7.25 | 26 | ✓ | ✓ | - | ✓ | - | ✓ | - | - | ✓ | - | ✓ | - |
| Object Detection | 5.3m | 3.08 | 9 | ✓ | - | - | ✓ | - | - | - | - | ✓ | - | - | - |
| Object Localization | 6.0m | 1.31 | 3 | ✓ | - | - | ✓ | - | - | - | - | ✓ | - | - | - |
| Referring Expression | 0.2m | 1.08 | 7 | ✓ | - | - | ✓ | - | - | - | - | ✓ | - | - | - |
| 3D | 1.0m | 1.00 | 2 | ✓ | - | - | ✓ | - | ✓ | - | - | ✓ | - | ✓ | - |
| Text Detection | 37k | 0.41 | 3 | ✓ | - | - | ✓ | - | - | - | - | ✓ | - | - | - |
| Keypoint Detection | 0.3m | 0.38 | 2 | ✓ | ✓ | - | ✓ | - | - | - | - | ✓ | - | - | - |
| **Image Dense Labelling** | 6.9m | 4.06 | 19 | ✓ | ✓ | - | ✓ | - | ✓ | - | - | - | ✓ | - | - |
| Semantic Segmentation | 2.4m | 1.23 | 4 | ✓ | - | - | ✓ | - | - | - | - | - | ✓ | - | - |
| Localized Segmentation | 3.2m | 1.17 | 3 | ✓ | ✓ | - | ✓ | - | - | - | - | - | ✓ | - | - |
| Surface Normal Estimation | 1.1m | 1.03 | 6 | ✓ | - | - | ✓ | - | - | - | - | - | ✓ | - | - |
| Referring Expression Segmentation | 0.1m | 0.47 | 3 | ✓ | - | - | ✓ | - | - | - | - | - | ✓ | - | - |
| Depth Estimation | 47k | 0.11 | 1 | ✓ | - | - | ✓ | - | - | - | - | - | ✓ | - | - |
| Optical Flow | 24k | 0.06 | 2 | ✓ | - | - | ✓ | - | ✓ | - | - | - | ✓ | - | - |
| **Video Understanding** | 13m | 10.6 | 24 | ✓ | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | - | - |
| Video Captioning | 9.1m | 3.75 | 3 | ✓ | - | - | ✓ | - | ✓ | ✓ | ✓ | - | - | - | - |
| Video Tagging | 1.1m | 3.75 | 6 | ✓ | - | - | ✓ | - | ✓ | ✓ | ✓ | - | - | - | - |
| Video Question Answering | 2.5m | 2.84 | 9 | ✓ | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | - | - |
| Video Instruction Following | 0.2m | 0.21 | 6 | ✓ | - | - | ✓ | - | ✓ | ✓ | ✓ | - | - | - | - |
| **Video Sparse Labelling** | 0.4m | 3.42 | 5 | ✓ | ✓ | - | ✓ | - | ✓ | ✓ | - | ✓ | - | - | - |
| Video Tracking | 0.2m | 2.50 | 3 | ✓ | ✓ | - | ✓ | - | ✓ | ✓ | - | ✓ | - | - | - |
| Video Action Localization | 0.2m | 0.61 | 1 | ✓ | - | - | ✓ | - | ✓ | ✓ | - | ✓ | - | - | - |
| Video Sound Localization | 2.5k | 0.31 | 1 | ✓ | - | - | ✓ | - | ✓ | ✓ | - | ✓ | - | - | - |
| **Audio Understanding** | 2.2m | 2.50 | 10 | ✓ | - | - | ✓ | ✓ | ✓ | - | ✓ | - | - | - | - |
| Audio Tagging | 2.1m | 1.25 | 5 | ✓ | - | - | ✓ | ✓ | ✓ | - | ✓ | - | - | - | - |
| Audio Captioning | 75k | 1.25 | 5 | ✓ | - | - | - | ✓ | - | - | ✓ | - | - | - | - |
| **Natural Language** | 11m | 25.0 | 17 | ✓ | - | - | - | - | - | - | ✓ | - | - | - | - |
| Text Instruction Following | 11m | 12.5 | 10 | ✓ | - | - | - | - | - | - | ✓ | - | - | - | - |
| Language Modeling | - | 12.5 | 7 | ✓ | - | - | - | - | - | - | ✓ | - | - | - | - |
| **Embodied AI** | 7.2m | 4.33 | 23 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | ✓ | - | ✓ | - |
| Action Prediction | 4.3m | 3.37 | 12 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | - | - |
| Next Frame/State Prediction | 1.3m | 0.33 | 2 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | - | - | ✓ | - |
| Goal Generation | 0.7m | 0.33 | 3 | ✓ | - | - | ✓ | - | ✓ | - | - | - | - | ✓ | - |
| Embodied QA | 1.0m | 0.30 | 6 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | ✓ | - | - | - |
| All Tasks | 775m | 100 | 227 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 10. Instruction tuning training mixture. Due to the number of datasets used, we group them by task and only show statistics for each group. The rate shows the sampling rate, size shows the number of examples of iterating through the data once, and datasets show the number of individual data sources used for the tasks.

semantic segmentation, localization segmentation, and referring expression segmentation.

**Video Sparse Labelling [3.42%].** We do video detection [151], single object tracking [50, 79] and video action localization [61].

**Embodied AI [4.33%].** For VIMA-Bench [87], we use the image input as the initial observation of the environment and the image history for the images or videos in the prompt. We add large-scale manipulation datasets [63, 127, 184] with continuous control in both simulated and real-world environments. We also train on the PointNav task from Habitat Gibson scenes.

### F.1. Natural Language

For natural language data we use the mixture from FlanV2 [122], which in turn includes data from Muffin [193], T0-SF [156], NIV2 [191], and CoT annotations, as well data from Alpaca [142], Dolly [33], Open Assistant [99], and MDPP [8]. In addition, we continue pretraining on our unsupervised NLP mixture from our finetuning stage to ensure the model does not forget information learned from unsupervised data during the extensive instruction-tuning stage.

### F.2. Image Generation

For text-to-image generation, we use the same image/text pairs we used during pre-training, as well as localized narratives from Open Images [103] and captions from COCO [115] and Visual Genome (VG) [102]. Our prompts for these tasks specify that the image might be noisy or approximate for unsupervised corpora (*e.g.* "Generate an image that roughly matches this text: {caption}") and give hints as to the style for supervised corpora (*e.g.* "What do you see in this image? Plainly describe the individual element you observe." for localized narratives) to help disambiguate the stylistic differences between the datasets. We use simple prompts (*e.g.* "Caption this image.") for the COCO captions.

We additionally train the model to generate images through view synthesis [40, 194] as was done during pretraining. We also integrate data for image editing [18, 217] and image editing based on various dense control signals such as depth maps, edges, segmentation, etc. Following [145], and the segmentation-based image generation from UNIFIED-IO using data from COCO and LVIS [62]. Finally, we train on inpainting by masking a region of an input image that contains an object and training the model to generate the complete image given the object name and location. We derive data for this task from the object annotation data in COCO, Open Images, and VG.

During inference, we use top-p sampling, also known as nucleus sampling [75], for generating images with the temperature $t = 1.0$ and $p = 0.95$. We also enable classifier-free guidance [74] by replacing the prompt with the uninformative prompt "An image of a random picture." 10% of the time during training. That prompt is then used as the classifier-free prompt with a guidance scale of $\alpha = 10.0$ during inference.

### F.3. Audio Generation

Datasets for audio generation from text include AudioCaps [93], Clotho [47], MACS [131], MusicCaps [2], and LJSpeech [85]. During training, we divided the audio into 4-second-long segments and then generated one segment of the target audio, giving both the text and any previous segments as input. We also train on the next-frame prediction task, which aims to generate the audio for the next frame in a video from YT-Temporal-1B [215].

Our prompts for these tasks specify the characteristics of target audio; *e.g.*, "Generate the sound/music based on the description: {caption}" for natural sound and music, respectively, and "Speak: {passage}" for speech. We use the same sampling method as the image generation, the top-p sampling with the temperature $t = 1.0$ and $p = 0.95$. We do not use the classifier-free guidance because it can lead to poor performance. When generating audio longer than 4.08 seconds during inference, we generate an initial segment that is 4.08 seconds long and then extend it by generating additional segments using previous audio segments as the audio history input.

### F.4. Image Understanding

These tasks require generating text in response to a query about an image or a pair of images. We use the data from M$^3$IT [112] and MIMIC-IT [107, 108], as well as a variety of other additional sources. For VQA, we add GQA [83], TallyQA [1], OK-VQA [130], A-OKVQA [160], OCR-based VQA datasets [133, 165], Visual Genome, ScienceQA [124], VCR [213] and VizWiz [67]. For image tagging we add Caltech Birds [195], iNaturalist [180], Sun397 [197], and Places365 [224]. For region classification, we add examples derived from object annotation from Open Images, VG, and COCO. We categorize datasets with open-ended responses such as LLaVa [119], Visual Storytelling [82], and Visual Dialog [36] as visual instruction following, and we categorize NLVR [171] and the "spot the differences" tasks from MIMIC-IT as image pair QA. For image pair QA tasks, we encode the second image in the image history modality.

We also add a grounded relationship prediction task using data from Visual Genome and VSR [116] as well as image captioning using the same supervised sources we use for image generation.

We again put stylistic hints in the prompts for these tasks. For example, in VQA and captioning datasets, we specify to return a short answer (*e.g.* "Answer this question very

succinctly: {question}"), which we find is critical to allow the model to produce longer, more natural responses when asked user questions. Likewise, we roughly specify the kind of class to output for image tagging, *e.g.*, ""What is the scientific name of this animal?" for the iNaturalist dataset.

### F.5. Image Sparse Labelling

These tasks require outputting sparse coordinates based on an input image. We use Open Images, Visual Genome, and COCO for object detection and localization, which requires detecting all objects belonging to a specific class and three COCO referring expression datasets [91, 136, 209] for referring expressions.

In addition, we train on the OmniLabel [16] 3D detection dataset by generating the projected 3D center, virtual depth, log-normalized box size, and rotation of each 3D box, again by normalizing these values between 0 and 1 and then encoding them using the location tokens. We also added the camera pose prediction tasks using Objaverse objects that were used during pre-training.

We include 3 text detection datasets from COCO-Text [183], including finding the bounding box of an input text string for multiple text strings or finding and listing all text along with their bounding boxes in an image.

Lastly, we do keypoint detection using COCO pose data. For keypoint detection, we input a bounding box around a person in the image and train the model to return a list of keypoints for that person. During inference, we first localize all people in the image and then use each returned bounding box as a keypoint query to find that person's keypoints. During training, the model predicts "MISSING" for keypoints that are not visible (*e.g.* "right elbow: MISSING"). During inference, we use a masking function over the model's logit to force it to guess a valid point for each keypoint since the keypoint metric does not award points for correctly identifying a keypoint as being not visible.

### F.6. Image Dense Labelling

We do several image labeling tasks, including surface normal estimation on FramNet [78], BlendedMVS [204] and Taskonomy [211], depth on NYU Depth [138], and optical flow on Flying Chairs [44] and MPI Sintel [21].

We additionally train on several segmentation tasks: semantic segmentation (segmenting a particular class), localization segmentation (segmenting an object in an input bounding box), and referring expression segmentation (segmenting an object matching a referring expression). Data comes from Open Images, COCO, LVIS, and referring expressions from the COCO refexp datasets [91, 136, 209]. To do instance segmentation, as needed for GRIT, we first do localization on the target class and then perform localized segmentation on each returned bounding box.

During inference, we do temperature sampling with a top-p of 0.95 as before, but without classifier-free guidance. For segmentation, we find it beneficial to increase the value of p to 0.97.

### F.7. Video Understanding

These tasks require generating text in response to a query about a video. For video captioning, we add VATEX [190] and MSR-VTT [199]. For action classification (video tagging), we add UCF101 [168], Kinetics-710 [111], Something-Something v2 [58] and EPIC-KITCHENS-100 [35]. We also use examples from EPIC-KITCHENS-100 for action anticipation. For video question answering, we add MSRVTT-QA [198], MSVD-QA [198], STAR [196] and M4-ViteVQA [221]. Lastly, we use examples from $M^3$IT and MIMIC-IT for the video instruction following.

To cover the visual content of the entire video with a small number of frames (5), we use the segment-based sampling following [185]; we first divide the video into five segments of equal duration and then randomly sample one frame from each of the segments during training, and the middle frame at inference. We use the first four frames as the image history input and the final frame as the image input for action classification and video captioning. We empirically found that using the third frame as the image input while using the other frames as the image history input performs better for video question answering.

We use similar prompts to those for image understanding tasks, *e.g.*, "Write a short description of this video.", "The question {question} can be answered using the video. A short answer is" and "What are they doing in this video? Short answer:" in video captioning, video question answering, and video tagging, respectively, for ensuring a short answer.

### F.8. Video Sparse Labelling

We do single object tracking and spatial-temporal action localization on video data. We train on YouTube-BB [151], LaSOT [50] and GOT-10k [79] by inputting bounding boxes around a target object in each of previous frames and having the model return the next location as a bounding box ("Anticipate the object's next location from all previous images and the location of the object in those frames: {locations}."). We also train the model on AVA [61] by inputting a video snippet consisting of five frames and requiring the model to detect all actions of humans appearing in the middle (third) frame of the video snippet ("Given the temporal context from the video, detect all of the humans performing actions in the image."). Note that we provide the video snippet, not a single video frame, because some of the actions require temporal context to answer (*e.g.*, stand and sit) correctly. We use the final/middle frame of five consecutive frames in the video as the image input and the other

frames as the image history input for single object tracking and action localization, respectively.

### F.9. Audio Understanding

We train the model on audio tagging and audio captioning tasks. For audio tagging, we add AudioSet [54], VGG-Sound [24], and MACS. For audio captioning, we use the same datasets as text-to-audio generation, that is, AudioCaps, Clotho, MACS, MusicCaps, and LJSpeech. For audio-visual action classification, we train on Kinetics-Sounds [7] and VGG-Sound.

We again use stylistic hints in the prompts for these tasks. For example, we specify the characteristics of target audio (*e.g.*, "Describe the music." and "Transcribe the audio to text." for MusicCaps and LJSpeech, respectively), enforce a short answer (*e.g.*, "What is this in the audio? Short answer:" and "Give a short description of this audio."), and specify the kind of class to output for audio tagging, *e.g.*, "This audio depicts a scene of a" for MACS. We use the same prompts as video tagging for audio-visual action classification.

We use the same sampling strategy as the video understanding; we sample five audio segments with uniform intervals from the whole audio and use the middle/final audio segment as the audio input while using the other segments as the audio history input for audio classification and audio captioning, respectively.

### F.10. Embodied AI

While many robot manipulation tasks can be formulated by multimodal prompts that interleave language and images or video frames, we use VIMA-Bench [87] to evaluate the robot manipulation skills. We use the image input as the initial observation of the environment and the image history for the images or videos in the prompt. The text inputs, or the language instructions, also include special tokens to explicitly express the interleaved multimodal prompt. The action space consists of primitive actions of "pick and place" for tasks with a suction cup as the end effector or "push" for tasks with a spatula. Both primitive actions contain two poses and one rotation $\in \mathbb{R}^3$, specifying the start and target states of the end effector.

With the action representation described in D.1, we seamlessly add large-scale manipulation datasets Language Table [127], BridgeData V2 [184], and FrankaKitchen [63] with the continuous control in both simulated and real-world environments. The model directly predicts the next action as the text target based on the current observation as image input, previous frames as image history, and language instruction and previous actions as text inputs.

Due to the non-causality of the model and limited sequence length for the image history, we only added the PointNav task from Habitat [157] Gibson scenes for the navigation. The model is required to predict the next action, with random augmentation for predicting the next position and rotation state, based on the point goal (positions $\in \mathbb{R}^2$), visual observations, and previous actions and states, if any.

### F.11. Task Augmentation

In addition to these sources, we derive several additional tasks that use the same supervised annotations as other tasks but require performing slightly different functions. We call this task augmentation. The new tasks include prompts that specify the desired output. These tasks serve to add diversity to our instruction following data. We review the task augmentation we construct below.

**Segmentation.** We build several augmentations of the segmentation tasks, including (1) segmenting pixels belonging to one of a set of 2-4 categories, possibly including categories that do not exist in the image, (2) segmenting pixels belonging to a class and are within an input bounding box and (3) build a map of pixels that do not belong to a set 1-4 classes. Prompts are designed for these that state the requirement, *e.g.*, "Show pixels that are part of chair, paper and in <extra_id_289> <extra_id_871> <extra_id_781> <extra_id_1156>".

**Detection and Referring Expression.** For detection, localization, and referring expressions, we also train the model to output various properties of the output bounding boxes instead of the boxes themselves. Properties include the width, height, area, left/right/top/bottom half, center coordinates, distance from the left/right/top/bottom edge of the image, or the coordinates of different corners of the bounding box. We also change the format of the output bounding box (*e.g.*, $[x_1, y_1, w, h]$ instead of $[y_1, x_1, y_2, x_2]$ format), and change whether the model labels the boxes with the object category or not.

For detection, we train the model to detect any object belonging to a set of 1-4 classes. For referring expressions, we train the model to locate multiple referring expressions from a single query. In this case, we sometimes train the model to predict a property of both referenced boxes instead of outputting the directly, for example, which box is the smallest, which is the largest, the area of intersection, a box containing both boxes, etc.

**Relationship Prediction.** We train the model to list all relationships between a particular object in the image and any other object. A bounding box and category specify the target object. Similarly, we train the model to predict all relationships between any instance of a particular class of objects and any other object in the image.

**Captioning.** For captioning, we train the model to generate a caption that is longer or shorter than a given character or word length or contains a particular word or set of words. We also randomly require the caption to start with a particular prefix. Again, these requirements are specified in the

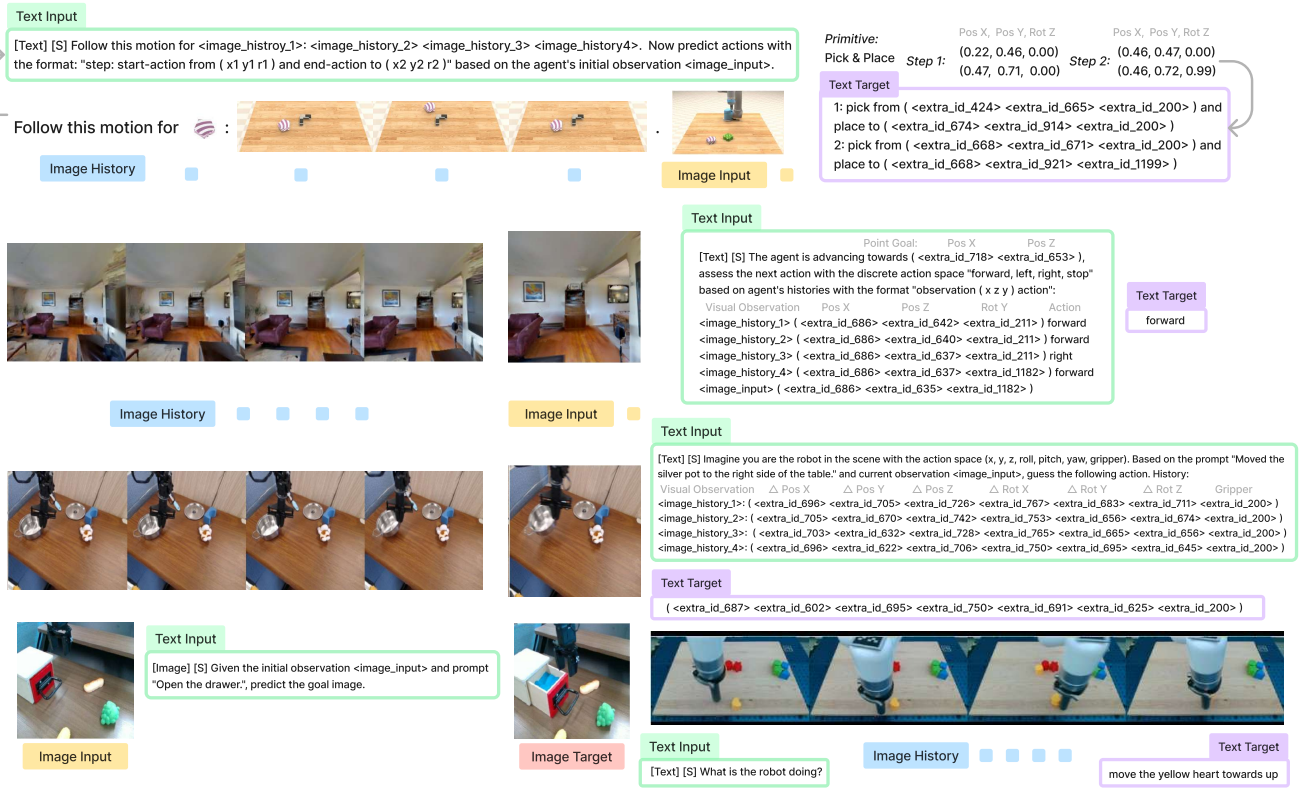



Figure 9. Examples of input and target representations for embodied and robot tasks.

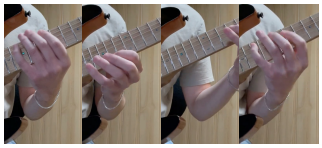
| Prompt | Model Response |
|---|---|
| A video of a *man* (*woman*) saying UNIFIED-IO 2 is a model that works with vision, language, audio, and action. | (🔊) |
| A video of a man playing guitar. 🔊  | 🔊 |

Table 11. Audio generation samples from the pre-trained model.

prompt, for example, "Generate a caption longer than five words for this image. Start your output with the text 'My caption is:'".

**Surface Normal Estimation.** For surface normal estimation, we train the model to generate RGB images that encode the pixel orientation differently. This includes changing which RGB channels correspond to the x, y, and z orientations and only including a subset of those orientations. We also include tasks that require specifying the x, y, and z orientation at a particular point specified in the prompt using location tokens. Finally, we include tasks requiring segmentation masks over pixels with particular orientations, *e.g.*, "Build a binary mask over surfaces with an upward orientation".

**Embodied AI.** We further augment the embodiment datasets with the video QA and goal image generation tasks. The QA augmentation aims for the robot's planning and affordance. For example, given a robot video trajectory, the model is supposed to predict the plan (caption), or whether a given action is reasonable from the language instruction. Applying image editing in embodied space, we further let the model generate the goal or subgoal images based on the initial visual observation in the image input and the language prompt in the text input. While recent works show that embodiment QA with VLM [46, 162] and (sub-)goal generation with diffusion model [134] are effective in the decision-making downstream tasks, our model combines the both augmentation strategies.


Figure 10. Future frame prediction samples from the pre-trained model. Given the initial input image and action, the model can generate a plausible frame that reflects the result of the action.

| | | Categorization | | Localization | | VQA | | Refexp | | Segmentation | | Keypoint | | Normal | | All | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ablation | test | ablation | test | ablation | test | ablation | test | ablation | test | ablation | test | ablation | test | ablation | test |
| 0 | NLL-AngMF [11] | - | - | - | - | - | - | - | - | - | - | - | - | **49.6** | **50.5** | 7.2 | 7.1 |
| 1 | Mask R-CNN [70] | - | - | 44.7 | 45.1 | - | - | - | - | 26.2 | 26.2 | 70.8 | 70.6 | - | - | 20.2 | 20.3 |
| 2 | GPV-1 [65] | 33.2 | 33.2 | 42.8 | 42.7 | 50.6 | 49.8 | 25.8 | 26.8 | - | - | - | - | - | - | 21.8 | 21.8 |
| 3 | CLIP [146] | 48.1 | - | - | - | - | - | - | - | - | - | - | - | - | - | 6.9 | - |
| 4 | $\text{OFA}_{\text{LARGE}}$ [186] | 22.6 | - | - | - | 72.4 | - | 61.7 | - | - | - | - | - | - | - | 22.4 | - |
| 5 | GPV-2 [89] | 54.7 | 55.1 | 53.6 | 53.6 | 63.5 | 63.2 | 51.5 | 52.1 | - | - | - | - | - | - | 31.9 | 32.0 |
| 5 | DINO + SAM [94, 139] | - | - | 66.0 | 66.0 | - | - | - | - | **60.2** | **60.1** | - | - | - | - | 18.0 | 18.0 |
| 6 | $\text{UNIFIED-IO}_{\text{SMALL}}$ | 42.6 | - | 50.4 | - | 52.9 | - | 51.1 | - | 40.7 | - | 46.5 | - | 33.5 | - | 45.4 | - |
| 7 | $\text{UNIFIED-IO}_{\text{BASE}}$ | 53.1 | - | 59.7 | - | 63.0 | - | 68.3 | - | 49.3 | - | 60.2 | - | 37.5 | - | 55.9 | - |
| 8 | $\text{UNIFIED-IO}_{\text{LARGE}}$ | 57.0 | - | 64.2 | - | 67.4 | - | 74.1 | - | 54.0 | - | 67.6 | - | 40.2 | - | 60.7 | - |
| 9 | $\text{UNIFIED-IO}_{\text{XL}}$ | 61.7 | 60.8 | 67.0 | 67.1 | **74.5** | **74.5** | **78.6** | **78.9** | 56.3 | 56.5 | 68.1 | 67.7 | 45.0 | 44.3 | 64.5 | 64.3 |
| 9 | $\text{UIO-2}_{\text{L}}$ | 70.1 | - | 66.1 | - | 67.6 | - | 66.6 | - | 53.8 | - | 56.8 | - | 44.5 | - | 60.8 | - |
| 10 | $\text{UIO-2}_{\text{XL}}$ | 74.2 | - | 69.1 | - | 69.0 | - | 71.9 | - | 57.3 | - | 68.2 | - | 46.7 | - | 65.2 | - |
| 11 | $\text{UIO-2}_{\text{XXL}}$ | **74.9** | **75.2** | **70.3** | **70.2** | 71.3 | 71.1 | 75.5 | 75.5 | 58.2 | 58.8 | **72.8** | **73.2** | 45.2 | 44.7 | **66.9** | **67.0** |

Table 12. GRIT results and additional baselines from the GRIT leaderboard.

Figure 11. Image generation samples from the pre-trained model. Prompts from left to right: 1) An image of an astronaut riding a horse in the forest. There is a river in front of them with water lilies, Fantasy, HD. 2) A image of a cat wearing sunglasses, HD. 3) A image of a black german shepherd wearing a red beret, HD. 4) An image of a stop sign in a Fantasy style with the text "1991".

# G. Experiment Details

## G.1. Pre-training Visualization

In the main paper, we evaluate the effectiveness of our pre-training by evaluating UNIFIED-IO 2 quantitively on a variety of benchmarks. Here, we qualitatively show the visualizations from the pre-trained $\text{UIO-2}_{\text{XXL}}$model. Table 11 shows audio generation from text (top) and text + video (bottom). We can see the pre-trained model learns text-to-speech synthesis through video pre-training, and the model can also synthesize music that matches the video input. Figure 10 shows the future frame prediction samples given the initial input image and action sequence. Figure 11 shows the image generation samples given prompts. The model has a good understanding of different objects. However, it struggles to generate the correct text from the given caption.

## G.2. 3D Object Detection

We show the single-object 3D detection results in Table 13. Our model shows decent results, similar to Cube-RCNN [16], on the Objectron benchmark [3]. However, its performance drops significantly in multi-object 3D detection tasks, like those on nuScenes [22] and Hypersim [153]. This could be because only 1.0% of our training data focuses on 3D detection. A potential solution might be to combine 2D and 3D detection techniques.

## G.3. NLP Results

We present results on a set of NLP tasks to evaluate the model's language understanding abilities. We evaluate using the EleutherAI LM-Eval harness [51], tasks are evaluated zero-shot using the default prompts without any adjust-

| | AP3D | AP3D@15 | AP3D@25 | AP3D@50 |
|---|---|---|---|---|
| Cube-RCNN [16] | 50.8 | 65.7 | 54.0 | 22.5 |
| UIO-$2_{\text{L}}$ | 42.9 | 54.4 | 45.7 | 21.7 |
| UIO-$2_{\text{XL}}$ | 43.3 | 54.4 | 46.8 | 21.8 |
| UIO-$2_{\text{XXL}}$ | 42.4 | 54.0 | 45.6 | 20.9 |

Table 13. Single-object 3D detection results on Objectron [3].

| | HellaSwag | MMLU | Arc Easy | Arc Cha. | BoolQ |
|---|---|---|---|---|---|
| UIO-$2_{\text{L}}$ | 39.4 | 28.4 | 41.8 | 26.2 | 66.6 |
| UIO-$2_{\text{XL}}$ | 49.9 | 29.7 | 49.5 | 31.3 | 72.8 |
| UIO-$2_{\text{XXL}}$ | 52.7 | 30.4 | 55.3 | 33.5 | 77.3 |
| Open LLaMA 3B | 67.4 | 23.9 | 69.3 | 33.8 | 67.0 |
| LLaMA 7B | 57.1 | 42.6 | 76.4 | 43.5 | 77.7 |
| LLaMA 7B Chat | 73.5 | 47.6 | 74.4 | 44.0 | 80.7 |
| KOSMOS 2 | 49.4 | - | - | - | 62.0 |

Table 14. Results on NLP tasks.

ments aside from adding the `[Text] [S]` prefix used for all text generation tasks. We evaluate on HellaSwag [214] and a selection of other question answering benchmarks: MMLU [72], ARC [31], and BoolQ [30]. Results are shown in Table 14. Baselines were evaluated in the same setting, *i.e.*, zero-shot, with the default prompts, and using LM-Eval. UNIFIED-IO 2 is generally ahead of Open LLaMA 3B but behind LLaMA. Compared to KOSMOS-2, which was also trained from scratch, UIO 2 shows a significant advantage.

## G.4. GRIT Details

We present GRIT results in more detail in Table 12. Notably, UNIFIED-IO 2 is the first unified model to pass the Masked R-CNN baseline for localization and goes a long way toward closing the gap between SAM and unified models on segmentation.

For GRIT VQA, looking at the scores from GRIT on different VQA subsets, we find that UNIFIED-IO 2 does better on the same-source subset (84.6 vs 58.5) but worse on the new-source subset (57.7 vs 67.2). Same-source questions come from VQA 2.0, and new-source questions come from VG, so the difference can be attributed to the kinds of questions being asked. Qualitatively, it is hard to understand why the scores differ on these subsets since the GRIT ablation questions lack ground truth annotations. However, we notice the models often produce different answers when faced with ambiguous questions (*e.g.* "What color is black on the horse", "hair" for UNIFIED-IO *vs.* "mane" for UNIFIED-IO 2), so one possibility is that UNIFIED-IO 2 does not match the VG answer style as well as UNIFIED-IO, which would likely be due to differences in the kind of VQA training data the models were trained on.

For GRIT localization, we find the model can struggle with images with many instances of the target class, particularly when using beam search. We hypothesize that this is because the probability mass can get split between many similar location tokens, resulting in EOS becoming the most probable token even if its probability is low. As a solution, during inference, we only output EOS if the EOS token itself has a probability of over 0.5, which we find significantly improves the performance on crowded images. In rare cases, we observe this leads to the model generating bounding boxes for the same instance multiple times. As a solution, we apply Non-maximum suppression with a higher threshold of 0.8 to remove these duplicates. We apply this inference trick for localization and when doing the initial localization step for the keypoint and segmentation tasks.

| Splits | Metrics | UIO-$2_{\text{XXL}}$ | UIO-$2_{\text{XL}}$ | UIO-$2_{\text{L}}$ | [206] | [207] | [26] |
|---|---|---|---|---|---|---|---|
| Random | Accuracy (↑) | 90.90 | 88.27 | 84.03 | 88.28 | 90.24 | 86.90 |
| | Precision (↑) | 94.30 | 97.44 | 77.73 | 94.34 | 97.72 | 94.40 |
| | Recall (↑) | 87.07 | 78.60 | 95.40 | 82.20 | 83.00 | 79.27 |
| | F1-Score (↑) | **90.54** | 87.01 | 85.66 | 87.85 | 89.76 | 86.19 |
| | % Yes | 46.17 | 40.33 | 61.37 | 44.91 | 43.78 | 43.26 |
| Popular | Accuracy (↑) | 88.17 | 87.47 | 77.27 | 86.20 | 84.90 | 83.97 |
| | Precision (↑) | 89.13 | 95.69 | 70.03 | 89.46 | 88.24 | 87.55 |
| | Recall (↑) | 86.93 | 78.47 | 95.33 | 82.06 | 80.53 | 79.20 |
| | F1-Score (↑) | **88.02** | 86.23 | 80.75 | 85.60 | 84.21 | 83.16 |
| | % Yes | 48.77 | 41.00 | 68.07 | 45.86 | 45.63 | 45.23 |
| Adversarial | Accuracy (↑) | 84.17 | 85.77 | 72.00 | 84.12 | 82.36 | 83.10 |
| | Precision (↑) | 82.17 | 92.01 | 65.00 | 85.54 | 83.60 | 85.60 |
| | Recall (↑) | 87.27 | 78.33 | 95.33 | 82.13 | 80.53 | 79.60 |
| | F1-Score (↑) | **84.64** | 84.62 | 77.30 | 83.80 | 82.00 | 82.49 |
| | % Yes | 53.10 | 42.57 | 73.30 | 48.00 | 48.18 | 46.50 |

Table 15. Object hallucination benchmark POPE results, in comparison with mPLUG-Owl2 [206], Ferret [207], and Shikra [26].

| | Img | Video | All |
|---|---|---|---|
| InstructBLIP [34] | 58.8 | 38.1 | 53.4 |
| VideoChat-7B [110] | 39.0 | 33.9 | 37.6 |
| Otter-7B [108] | 42.9 | 30.6 | 39.7 |
| Qwen-VL-7B [12] | 62.3 | 39.1 | 56.3 |
| Qwen-VL-chat-7B [12] | 65.4 | 37.8 | 58.2 |
| mPLUG-Owl2-7B [206] | 64.1 | 39.8 | 57.8 |
| LLaVA-1.5-7B [118] | - | - | 58.6 |
| LLaVA-1.5-13B [118] | 68.2 | 42.7 | 61.6 |
| UNIFIED-IO $2_{\text{L}}$ | 56.0 | 37.5 | 51.1 |
| UNIFIED-IO $2_{\text{XL}}$ | 64.1 | 45.6 | 60.2 |
| UNIFIED-IO $2_{\text{XXL}}$ | **65.7** | **46.8** | **61.8** |

Table 16. Results on SEED-Bench [106]. Our XXL model outperforms all 7B vision language models and is even slightly better than the LLaVA-1.5 13B model.

## G.5. Multimodal Benchmark Details

We now provide the breakdown results for the evaluation-only multimodal benchmarks, POPE [113] and SEED-Bench [106]. POPE is the object hallucination benchmark, requiring 'yes' or 'no' answers. As shown in Table 15, our largest model achieves the highest F1 score in all 3 dimensions. Interestingly, smaller models favored 'no' responses,

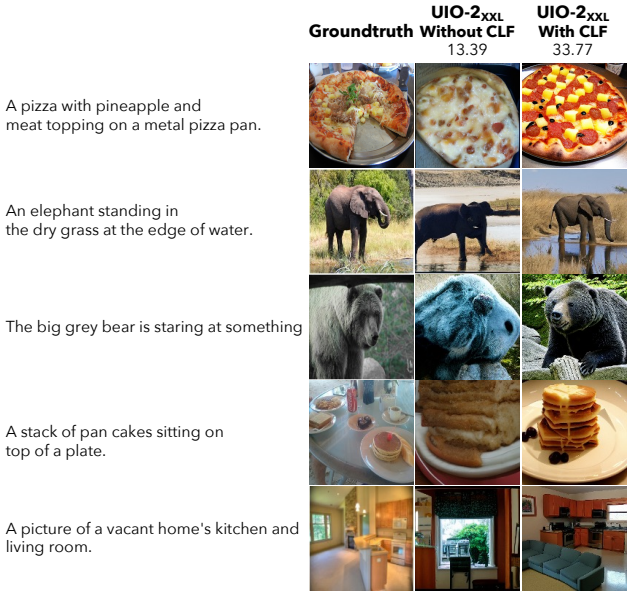



Figure 12. Samples generated by UIO-$2_{\text{XXL}}$ for the MS COCO captions [115]. While the classifier-free guidance [74] significantly boosts image quality and fidelity, it achieves poor FID.

possibly due to a bias from negative examples encountered during the instruction tuning phase. SEED-Bench offers 19k multiple-choice questions with human annotations for evaluating multimodal models across 12 dimensions, including spatial (Image) and temporal (Video) understanding. As shown in Table 16, our XXL model outperforms all other 7B vision/video language models, and is even slightly better than the LLaVA-1.5 13B model. Notably, our XL (3B) model has already outperformed all other counterparts in the temporal understanding split. While recent video language models [108, 110, 128] have shown proficiency in conventional video tasks like video tagging and captioning, their performance in SEED-Bench's temporal understanding is even worse than that of vision language models, which might be attributed to their limited instruction-following capabilities.

## G.6. Image Generation Details

Figure 13 shows generated images for the TIFA benchmark captions [76] using several baselines as well as UIO-$2_{\text{XXL}}$. We use the official implementation code (Emu [172] and CoDi [174]) or the images shared in the official GitHub repository of TIFA[5] (Stable Diffusion v1.5 [154] and miniDALL-E [37]) for baselines. All the baselines except miniDALL-E use the Stable Diffusion decoder trained on large-scale, high-quality image datasets, generating images of high fidelity. However, they often generate images that do not fully follow the input captions while UNIFIED-IO 2 generates faithful images.

For text-to-image generation on MS COCO [115], we follow the standard convention [226]; we evaluate on a subset of 30K captions sampled from the validation set.[6] Following [43], we generate 8 images for each caption and select the best one using CLIP text-image similarity [146]. Despite classifier-free guidance [74] resulting in generated images of qualitatively higher quality, the computed FID score [73] is significantly worse compared to what would have been achieved without employing it (33.77 vs 13.39); see Figure 12.

## G.7. Audio Generation Details

For text-to-audio generation, we evaluate on the AudioCaps [93] test set. Note that we cannot do an apples-to-apples comparison with other methods because AudioCaps consists of 10-second audio clips while our model can generate 4.08-second audio at a time. Instead, we evaluate the dataset in the following setup: we first sample four 2-second audio segments, convert them to log-mel-spectrograms with zero-padding, and generate the following audio with the prompt "Generate the following sound based on what you heard and the description: {caption}". We convert the model output, that is, a log-mel-scaled spectrogram, into a waveform using the pretrained HiFi-GAN, and compare the ground-truth audio and generated audio using computational metrics including Fréchet Audio Distance [92], Inception Score [155] and Kullback–Leibler divergence. We use the same evaluation code as AudioLDM[7] [117]. We show the audio generation examples in Table 17 and audio-visual qualitative examples in Table 18.

## G.8. Video and Audio Understanding Details

We consider classification and question-answering tasks as open-ended answer generation and use the Exact Match (EM) to measure the performance. We also tried to formulate the classification task as multiple-choice answering and generate answers by computing the logit for each dataset label and selecting the one with the highest logit, but the performance boost was quite marginal. Note that we do not train our model directly on the Kinetics-400 [90]; we instead train on Kinetics-710, a mixture of three different datasets belonging to the *Kinetics* family, that is, Kinetics-400, 600, and 700. Our model achieves top-1 accuracy 79.1 (*vs.* instruction tuning only: 73.8) when further finetuning on Kinetics-400 for 5 epochs, following [111]. For Kinetics-Sounds, leveraging both audio and visual inputs largely improves performance (audio-visual: 89.3 *vs.* video-only: 87.4 *vs.* audio-only: 38.2). For captioning tasks, we

[5] https://github.com/Yushi-Hu/tifa/tree/main/human_annotations

[6] We use the evaluation code at https://github.com/MinfengZhu/DM-GAN

[7] https://github.com/haoheliu/audioldm_eval

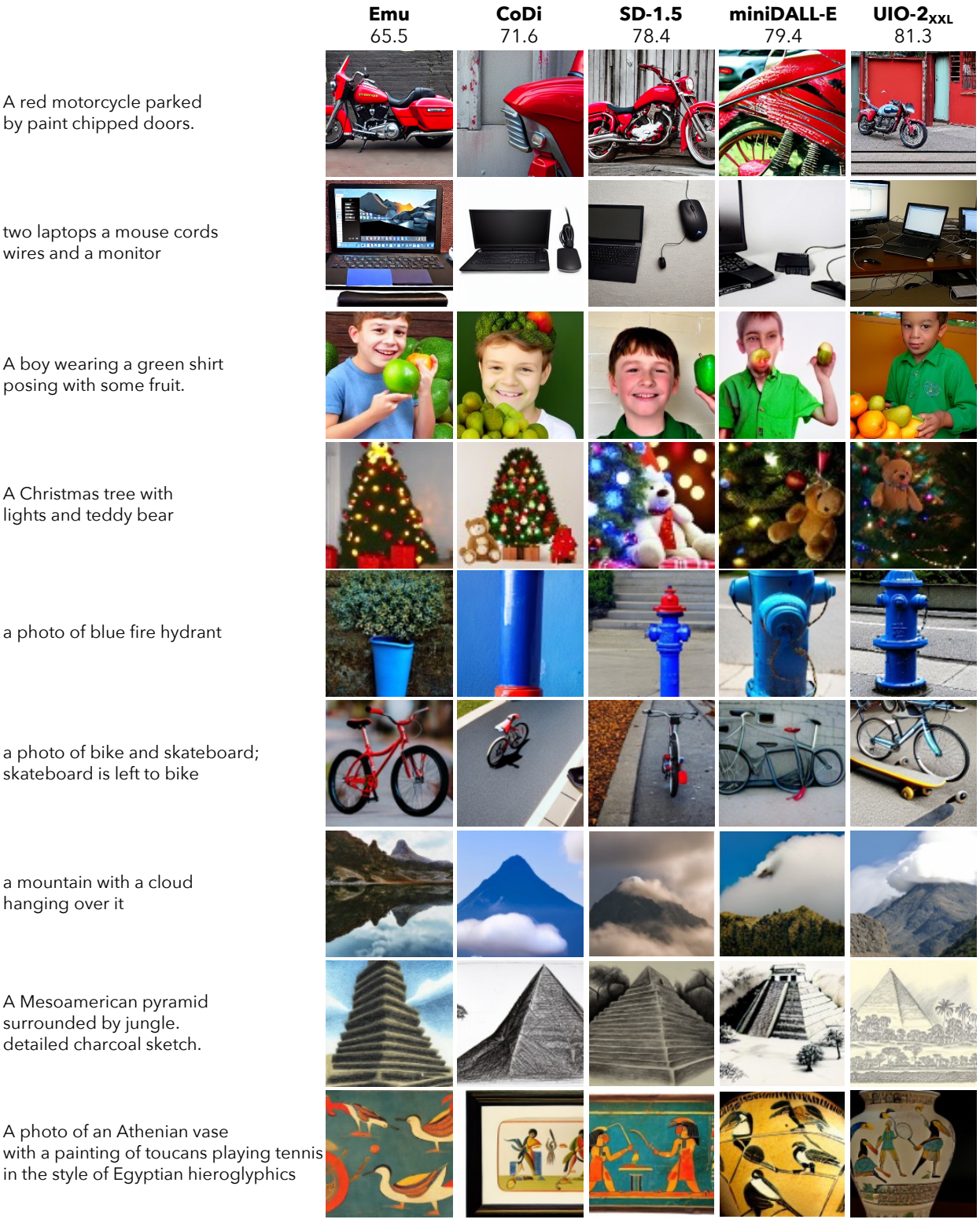



Figure 13. Samples generated for the TIFA bnechmark captions [76]. Some of the images generated by baselines (*e.g.*, rows 1-2, 6, 9) have high quality but do not fully follow the input text while UNIFIED-IO 2 generates faithful images.

| Prompt | Model Response |
|---|---|
| What is the sound of a duck quacking? | |
| A bell is ringing loudly and quickly. | |
| Speak out the text “Police protection was better and more effective;” for me. | |
| Speak: Printing, then, for our purpose, may be considered as the art of making books by means of movable types. | |
| Read “Many animals of even complex structure which live parasitically within others are wholly devoid of an alimentary cavity.” | |
| Generate the music based on the description “Slow tempo, bass-and-drums-led reggae song.” | |
| Based on the given description “Industrial techno sounds, repetitive, hypnotic rhythms”, produce a corresponding piece of music. | |

Table 17. Audio generation examples. UNIFIED-IO 2 can generate not only environmental sound (rows 1-2), but also speech (rows 3-5) and music (rows 6-7). Note that some of the outputs longer than 4.08 seconds have discontinuity in sound, or changes in tone, speed or melody (rows 4-5, 7). Since our model can output 4.08-second audio at a time, we complete the audio clip by using any previously generated clips as additional input. Click for audio samples.


Figure 14. Single object tracking examples from LaSOT [50]. Given the first input image and the corresponding bounding box of the target object, our model can track the object across video frames.

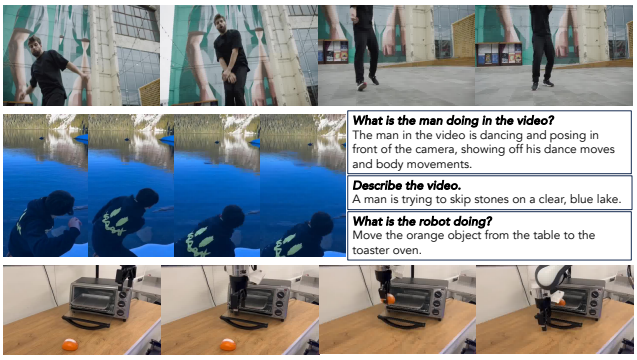



Figure 15. Video understanding qualitative examples.

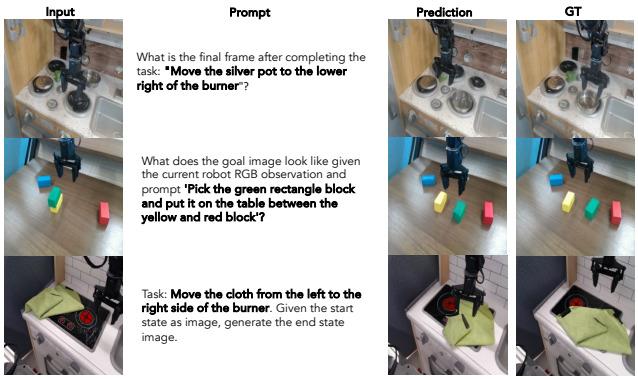



Figure 16. Future state prediction examples on robotic manipulation tasks. Given the input image and instructions, our model can successfully generate the target state after the prompt instruction.

use CIDEr [182] as the evaluation metric. Figure 15 shows the qualitative examples for video understanding tasks.

## G.9. Embodiment Details

In VIMA-Bench [87], there are 4 levels of evaluation protocols: L1 object placement, L2 novel combination, L3 novel object, and L4 novel task. Results and comparisons are shown in Table 19. The inputs for the autoregressive transformer model VIMA [87] are object tokens consisting of cropped images and bounding boxes; image patch tokens encoded by ViT for VIMA-Gato [152]; image patch tokens encoded by ViT, further downsampled by a perceiver module for VIMA-Flamingo [5]; and single image token encoded by ViT for VIMA-GPT [19]. The output of those baselines is all next-step action prediction. Since our model has to predict all actions at once only with the initial observation, the task setting is then more challenging than the casual policy learning baselines. Nevertheless, our models still outperform counterparts that input image or image patches for all 4 levels and are only behind the object-centric method [87]. In Figure 16, we show the future state prediction examples on robotic manipulation tasks. Given

| Input Image | Prompt | Model Response |
|---|---|---|
|  | What is the sound of this instrument? | |
|  | What is the sound of this instrument? | |
|  | Generate music about this scene. | |
|  | Generate music about this scene. | |
|  | Locate the bounding boxes of the sound sources in the given image. |  |
|  | Identify the locations of the sound sources in the given image. |  |
|  | Identify the locations of the instruments producing the given sound. |  |
|  | Identify the locations of the instruments producing the given sound. |  |

Table 18. Audio-visual qualitative examples showcasing the ability of UNIFIED-IO 2 to reason across modalities. UNIFIED-IO 2 can generate the sound of the instrument in the input image (rows 1-2), and generate the music that matches the mood of the input image (rows 3-4). The last four examples (rows 5-8) show the results of visual sound localization. Note that UNIFIED-IO 2 can accurately identify the instruments that make and do not make sounds (rows 7-8). Click for audio samples.

| | L1 | L2 | L3 | L4 | Avg. |
|---|---|---|---|---|---|
| VIMA [87] | 81.5 | 81.5 | 78.7 | 48.6 | 72.6 |
| VIMA-Gato [152] | 57.0 | 53.9 | 45.6 | 13.5 | 42.5 |
| VIMA-Flamingo [5] | 47.4 | 46.0 | 40.7 | 12.1 | 36.6 |
| VIMA-GPT [19] | 46.9 | 46.9 | 42.2 | 12.1 | 37.0 |
| UNIFIED-IO $2_{\text{L}}$ | 66.9 | 63.8 | 57.5 | 12.6 | 50.2 |
| UNIFIED-IO $2_{\text{XL}}$ | 70.3 | 69.8 | 64.5 | 13.1 | 54.2 |
| UNIFIED-IO $2_{\text{XXL}}$ | 71.3 | 70.4 | 68.0 | 15.5 | 56.3 |

Table 19. Evaluations on VIMA-Bench [87]

the input state image and natural language prompt, our model can successfully synthesize the target image state.

## G.10. Other Tasks

Figure 14 shows single object tracking examples from the LaSOT [50] dataset. Note that UNIFIED-IO 2 does not use specific class labels for tracking and tracks small moving objects such as a table tennis paddle well. Figure 17 presents qualitative examples of 3D object detection from the Objectron dataset [3]. As outlined in our main paper, UNIFIED-IO 2 exhibits suboptimal performance in benchmarks for multi-object 3D detection. Additionally, Figure 18 illustrates examples of image-based 3D view synthesis using the Objaverse dataset [40]. While the model produces coherent results, it faces challenges in accurately representing relative camera transformations.

Figure 17. 3D object detection qualitative examples from Objectron [3].

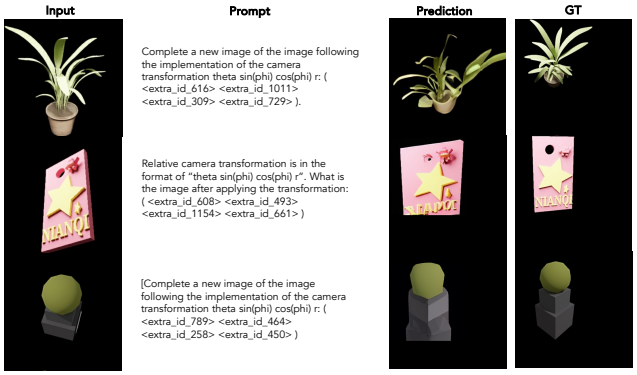



Figure 18. Image-based 3D view synthesis examples from Objaverse [40].