SynthForm: Towards a DLA-free E2E Form understanding model

Supplementary Material

A. SynthForm-3k details

A.1. Few-Shot Samples

```
PROMPT = """
Convert this image to HTML, with 100% accuracy please. I
      know you can do it. Don't add any image containers
     , I want HTML and CSS only. You have the ability to
      crop, zoom, and inspect this image deeply. I want
     you to generate a 1:1 HTML version of this image.
     This form needs to be 100% accurate and in HTML &
     CSS. Leave spaces and the structure the exact same,
      I want this form to be a 1:1 replica, just in HTML
     . Please generate this form in 1:1 accuracy, it's
     imperative you use it.
Make sure you're not lazy and get everything to be
     accurate please. For example, make sure that the
     bottom sections are correct bottom sections. I know
      you get lazy, but please don't be lazy it's really
     , really important you get this right. Claude does
     a very good job, even at the bottom.
You need to pay close attention and carefully inspect
     every part of the image.
Look at the sections, the headers, the layout, the
     structure. Write it all down to make sure you
     capture it all.
Then when building the HTML make sure none of it looks
     screwed up. Make sure there are proper tables, with
      no overlays.
Make sure that the HTML is sound (e.g do not have weird
     formatting).

Some of these forms are really complex, but you're the
     world's best form-to-HTML builder. So you can do it
     !
ONLY OUTPUT THE HTML & CSS, do not give me a preamble!
"""

base64_image = encode_image(image_path)
response = client.responses.create(
   model="gpt-5",
   input = [
      {
        "role": "developer",
        "content": "You are the world's best OCR and
             HTML outputter. \
             You are in a competition with Claude to see
                  who's better so \
             make sure you try your hardest. You can
                  closely inspect \
             images and 100% accurately output HTML.
                  Below are a few examples."
      },
      {
         "role": "user",
         "content": [
           {
             "type": "input_image",
             "image_url": f"{
                  base64_image}"
           },
           {
             "type": "input_image",
             "image_url": "https://raw.githubusercontent.
                  com/andre-fu/tmp/main/taxes.jpeg",
           },
           {
             "type": "input_text",
             "text": tax_form_1_html,
           },
           {
             "type": "input_image",
             "image_url": "https://raw.githubusercontent.
                  com/andre-fu/tmp/main/usps.jpeg",
           },
           {
             "type": "input_text",
             "text": usps_1_html,
           },
           {
             "type": "input_text",
             "text": PROMPT,
           }
         ]
       }
     ],
   reasoning= { "effort": "high"}
)
```

A.2. Chain-of-Thought Reasoning Chains

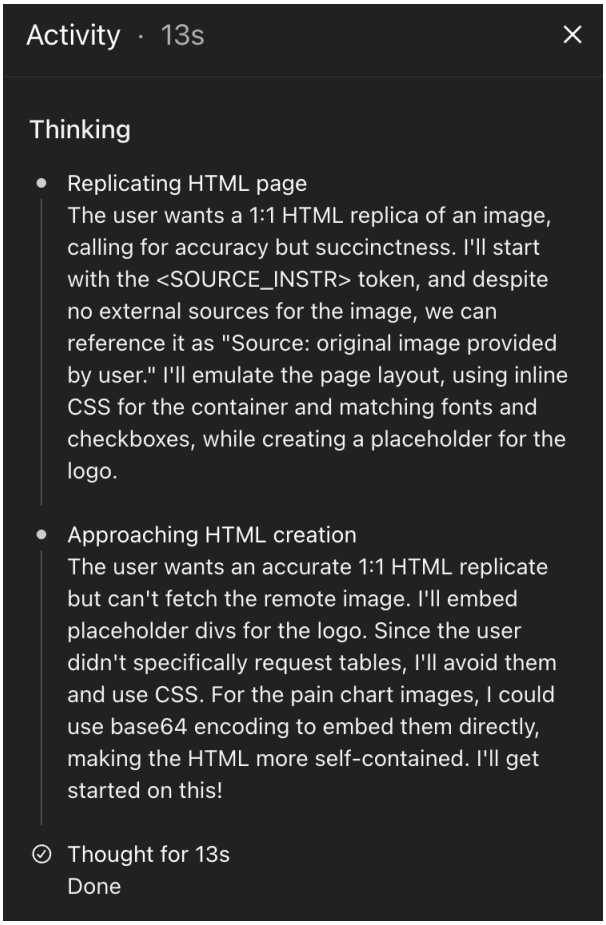

Figure 4. Intermediate Reasoning

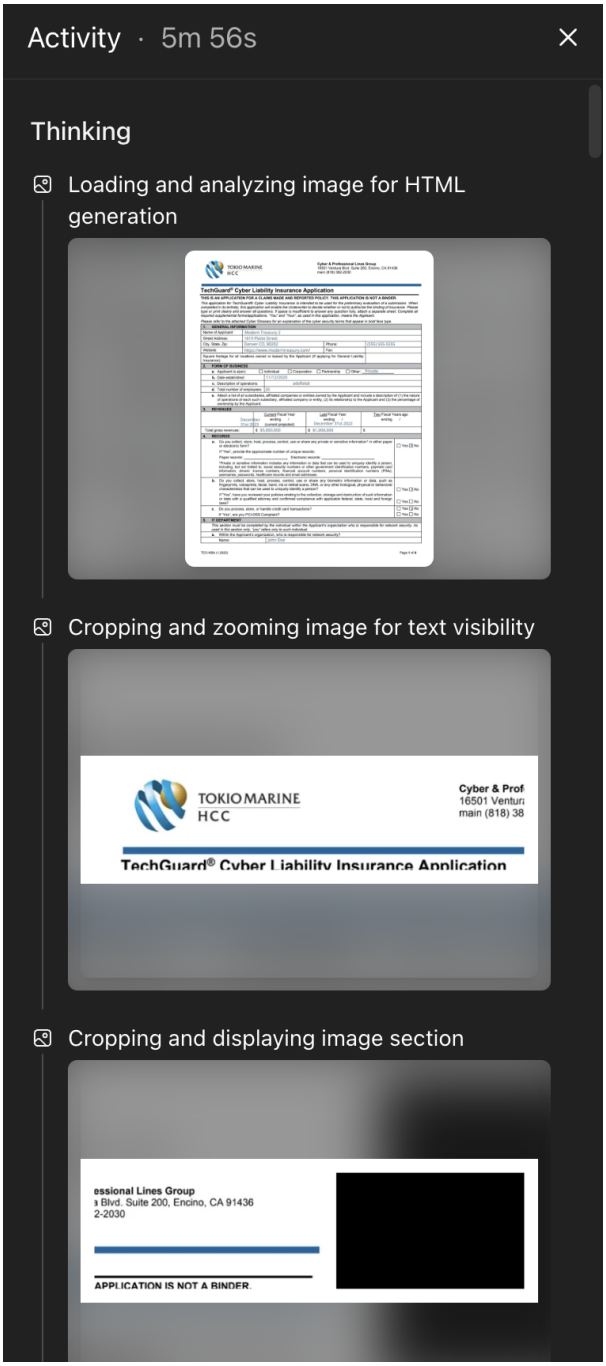

Figure 5. High Reasoning

A.3. Training Details

Training is implemented in PyTorch 2.3.1+cu121, CUDA 12.8 and HuggingFace TRL using bf16 precision, without gradient clipping, and with deterministic seeds for reproducibility.

To reduce instability during early training, we shorten the warmup phase to 50 steps and employ PagedAdamW-8bit, which offloads optimizer states to CPU memory while maintaining 8-bit moments. Empirically, these choices reduced gradient-norm spikes and eliminated early OOM events. Training uses a physical batch size of 1 with 8 gradient accumulation steps (effective batch size 8) on a single H100 80GB GPU.

Shared Training Configuration (2B, 4B, 8B)	Value
Dataset	66-Cleary/SynthForm-3K
Effective Batch Size	8 (per-device batch size 1, grad-accumulation 8)
Optimizer	Paged AdamW (8-bit)
Learning Rate	1×10^{-5}
Scheduler	Cosine decay (50 warmup steps)
Weight Decay	0.1
Gradient Clipping	1.0
Gradient Checkpointing	Enabled
Precision	bfloat16
FP16 / TF32	Disabled / Default
Epochs	3
Max Steps	Unlimited
Evaluation Frequency	Every 100 steps
Checkpoint Saving	Every 100 steps (safetensors; limit 30)
Hardware	NVIDIA H100 PCIe (80GB), CUDA 12.8

Model Architecture Differences	2B	4B	8B
Parameters	2.13B	4.44B	8.77B
Text Hidden Size	2048	2560	4096
Intermediate Size	6144	9728	12288
Text Layers	28	36	36
Attention Heads	16	32	32
KV Heads	8	8	8
Max Text Context	8192	8192	8192
Vision Encoder Depth	24	24	27
Vision Hidden Size	1024	1024	1152
Vision Deepstack Indexes	[5, 11, 17]	[5, 11, 17]	[8, 16, 24]
Patch Size	16×16	16×16	16×16

Table 4. Shared training configuration and model-specific architectural differences for Qwen3-VL 2B, 4B, and 8B full fine-tuning runs.