LTX-2

What is LTX-2?

LTX-2 is an audio-visual diffusion model released by Lightricks that can generate both audio and video simultaneously.

Recommended Settings

- Resolution
  - 640×640 (1:1)
  - 768×512 (3:2)
  - 704×512 (4:3)
  - *Upscaled 2x in post-processing, so actual output will be 1280×1280, etc.*
  - *Must be a multiple of 32*
- FPS
  - 24 / 25 / 30
- Frames
  - Max: 257 frames (approx. 10 sec at 25fps)
  - Recommended: 121–161 (balance of quality and memory)
  - *Must be 8n+1*

Model Download

- checkpoints (VAE included)
  - ltx-2-19b-dev-fp8.safetensors
- latent_upscale_models
  - ltx-2-spatial-upscaler-x2-1.0.safetensors
- loras
  - ltx-2-19b-distilled-lora-384.safetensors
- text_encoders
- gemma_3_12B_it_fp8_scaled.safetensors

```
ComfyUI/
└── models/
    ├── checkpoints/
    │   └── ltx-2-19b-dev-fp8.safetensors
    ├── latent_upscale_models/
    │   └── ltx-2-spatial-upscaler-x2-1.0.safetensors
    ├── loras/
    │   └── ltx-2-19b-distilled-lora-384.safetensors
    └── text_encoders/
        └── gemma_3_12B_it_fp8_scaled.safetensors
```

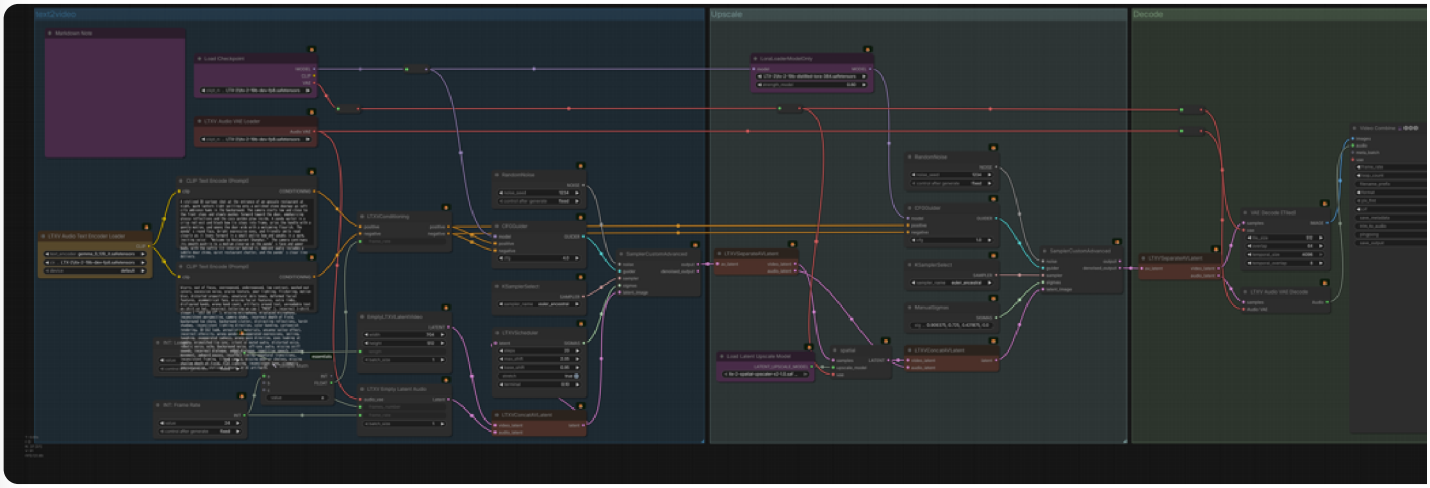
Basic Process Flow

It might feel complicated because there are more nodes compared to Wan, but this is all it does:

- 1. text2video + audio
  - First, generate the base video (and audio).
- 2. Hires.fix (2nd stage)
  - Upscale the generated video by 2x and refine it with video2video.
  - You can skip this and decode directly, but Hires.fix is recommended for quality.
- 3. Decode
  - Decode video and audio separately for output.

---

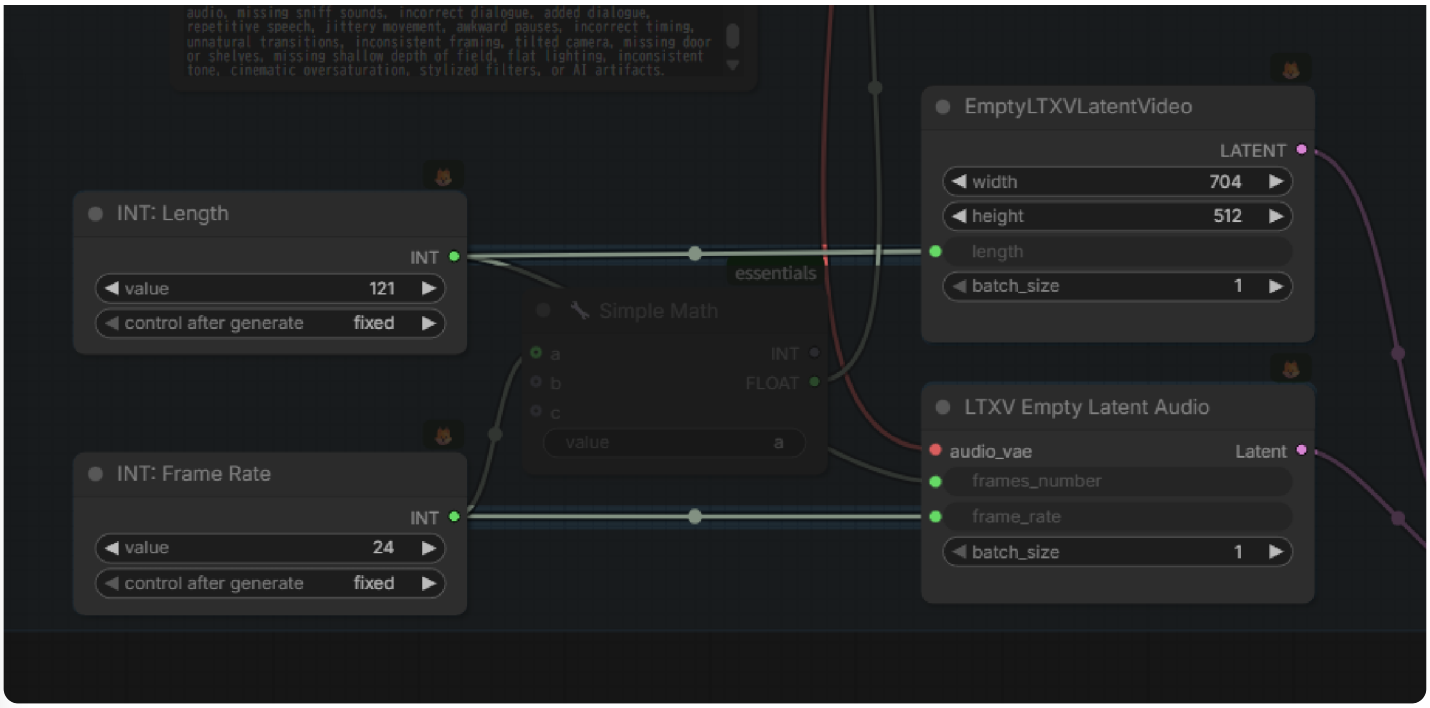
text2video



LTX-2_text2video_V2.json

Follow the basic flow explained above to build the workflow.

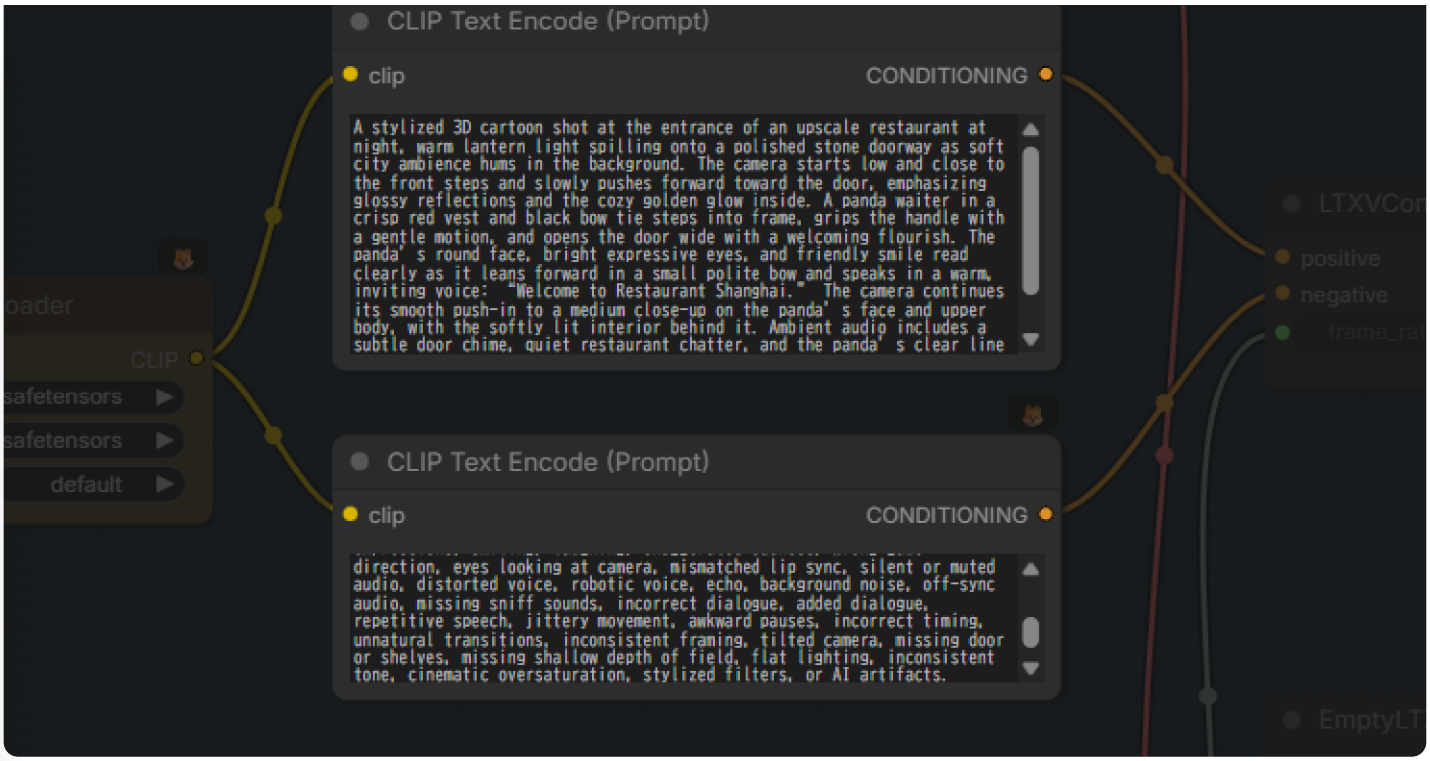
- 1, 2, 3 are the 1st stage.
- 4, 5 are Hires.fix.
- 6 is Decode.



1. Set Video Resolution, Length, FPS

Decide the parameters for the video and audio you want to generate here.

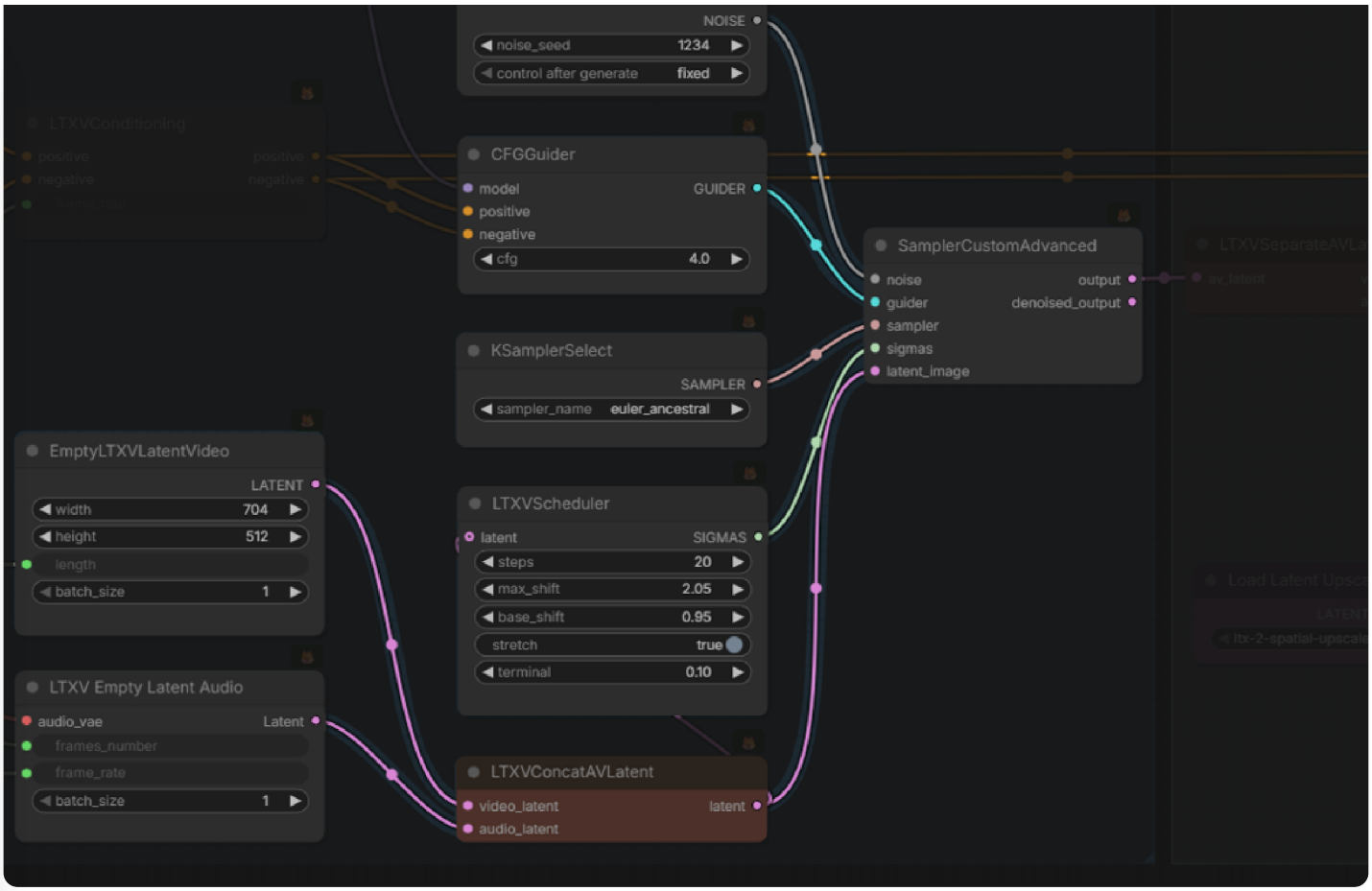
- Enter resolution, frame count, and FPS in `EmptyLTXVLatentVideo` / `LTXV Empty Latent Audio` .
- Follow the Recommended Settings.
- 🚨 Resolution will be doubled in post-processing.
  - In other words, set the resolution here to **half** the value of the video you want to create.



2. Prompt

A characteristic of the LTX series is that you need to be somewhat particular about the prompt, otherwise you won't get a very good video.

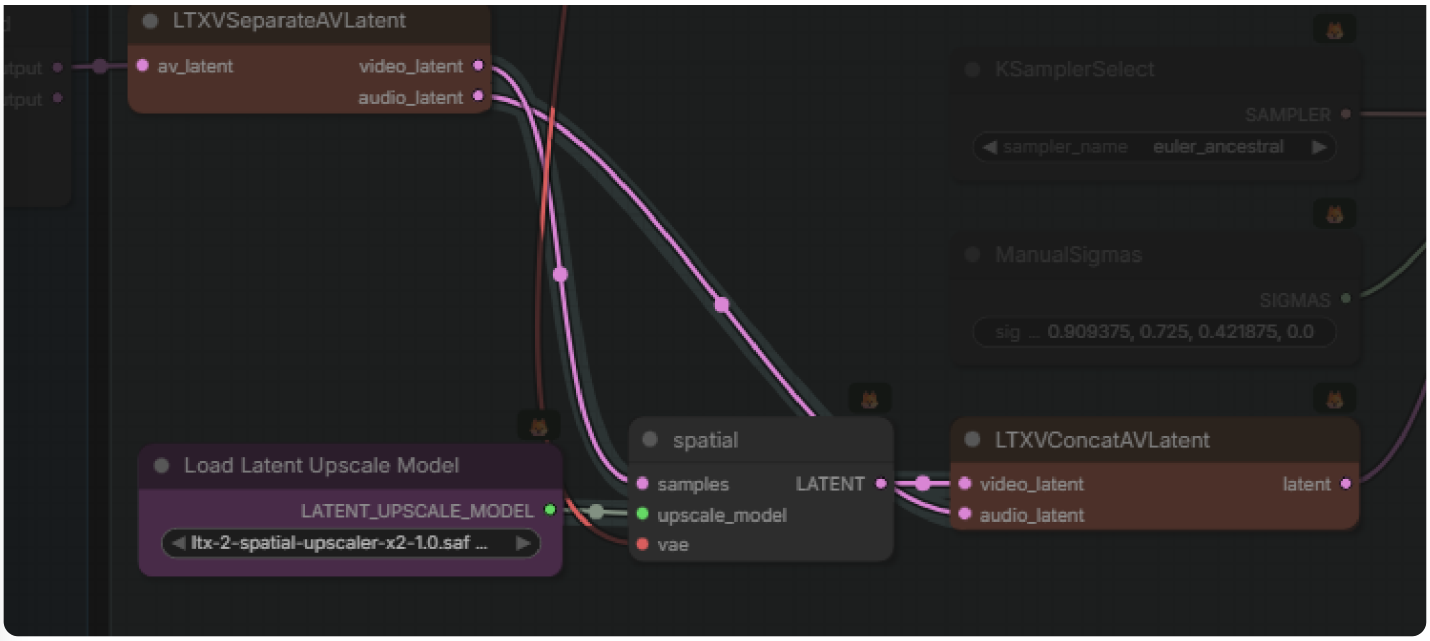
- That said, there isn't a strict format like when borrowing the power of LLMs.
- Try describing the video you want to generate as if you were writing a novel.
- cf. Prompting Guide for LTX-2



3. Sampling (1st Stage)

It doesn't look like the familiar `KSampler` so it might seem a bit complicated, but the basics are just "decide steps and CFG and sample".

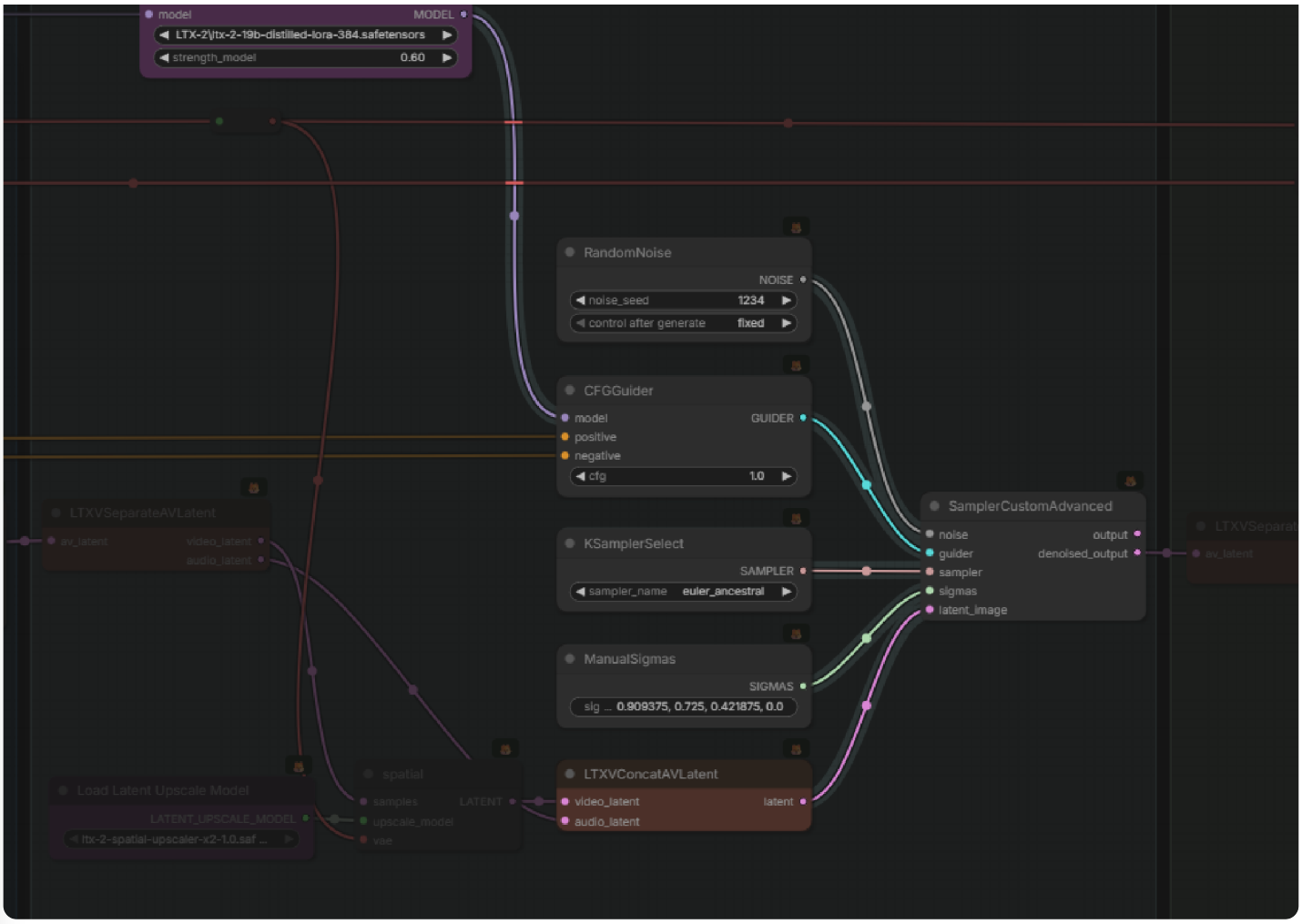
- In this workflow, the 1st stage is run with 20 steps / CFG 4.0.
- It uses a dedicated scheduler called `LTXVScheduler` .
  - It behaves similarly to `linear_quadratic` , but you don't need to worry about it too much.
- Since LTX-2 handles video and audio simultaneously, combine video latent and audio latent into one with `LTXVConcatAVLatent` .



4. Latent Upscale (x2)

Upscale the resolution of the video latent by 2x.

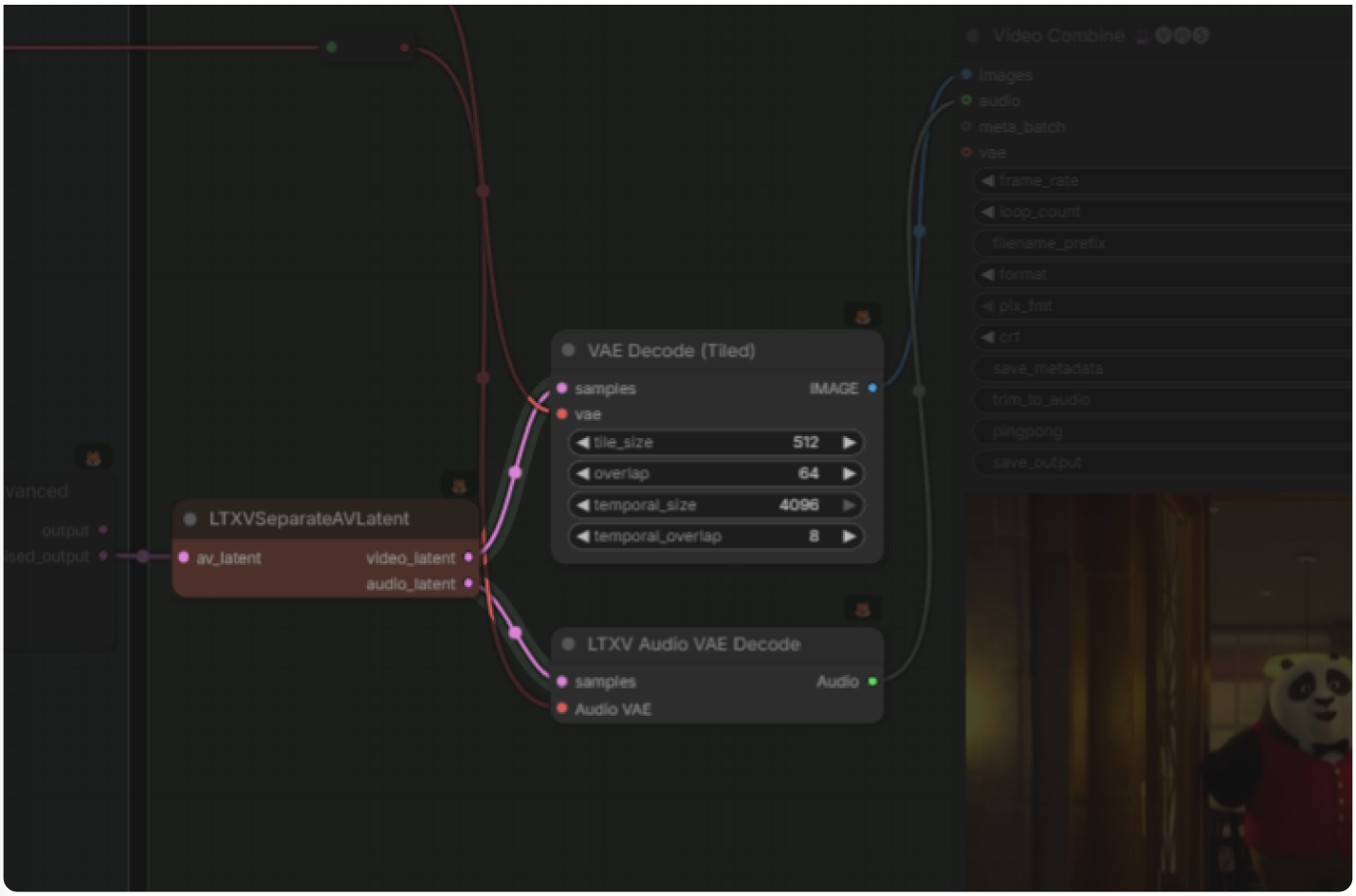
- Use a dedicated model (`ltx-2-spatial-upscaler-x2`).



5. Sampling (2nd Stage / video2video)

Refine the upscaled latent with short steps.

- Here we use `distilled-lora` which allows generation in 4~8 steps.
  - Think of it as something like Lightning / Turbo in other models.
  - This workflow runs in **3 steps**.
  - Accordingly, CFG is changed to `1.0` .
- Because it uses `Manual Sigma` , it's a bit hard to understand, but if thinking in terms of `Simple` , it behaves somewhat close to `denoise = 0.47` .



6. Decode

Finally, decode and export video and audio respectively.

- Separate the latent for video / audio and decode with appropriate VAE.
- (Tiled VAE is used because VRAM is tight.)

---

text2video 8 steps

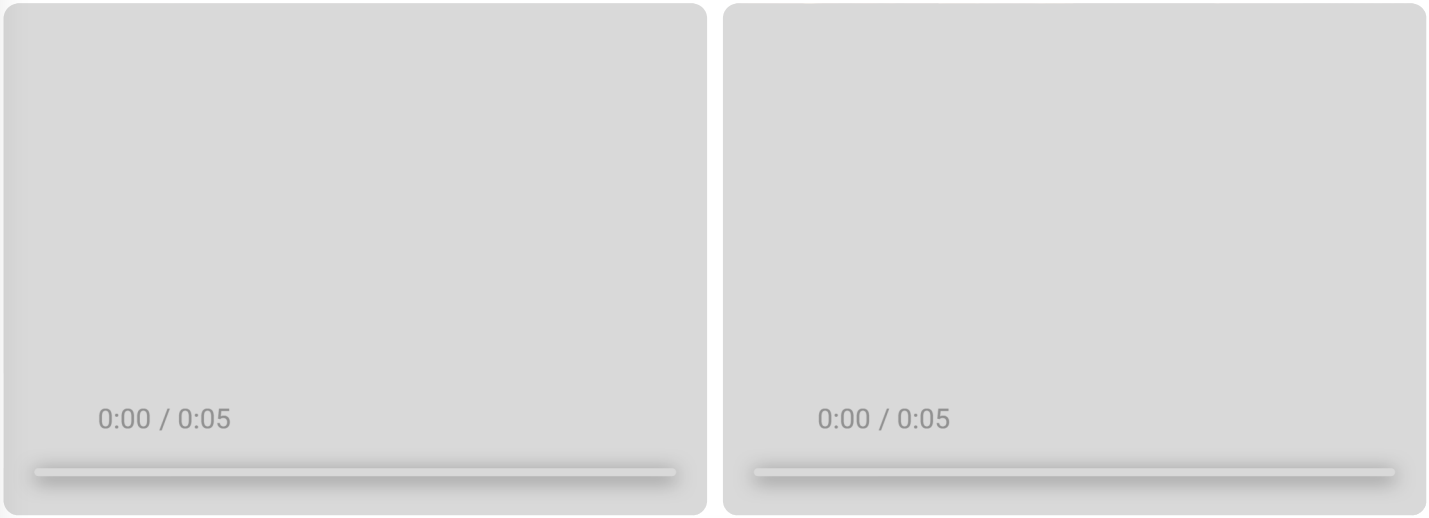
Above, we used `distilled-lora` only for Hires.fix, but let's apply it to the 1st stage as well and generate quickly in 8 steps.

LTX-2_text2video_distilled_V2.json

To apply `distilled-lora` , change some sampling settings.

- CFG : `1.0`
- scheduler : `Simple`
- steps : `8`

20 steps / 8 steps distilled-lora Comparison



20 steps

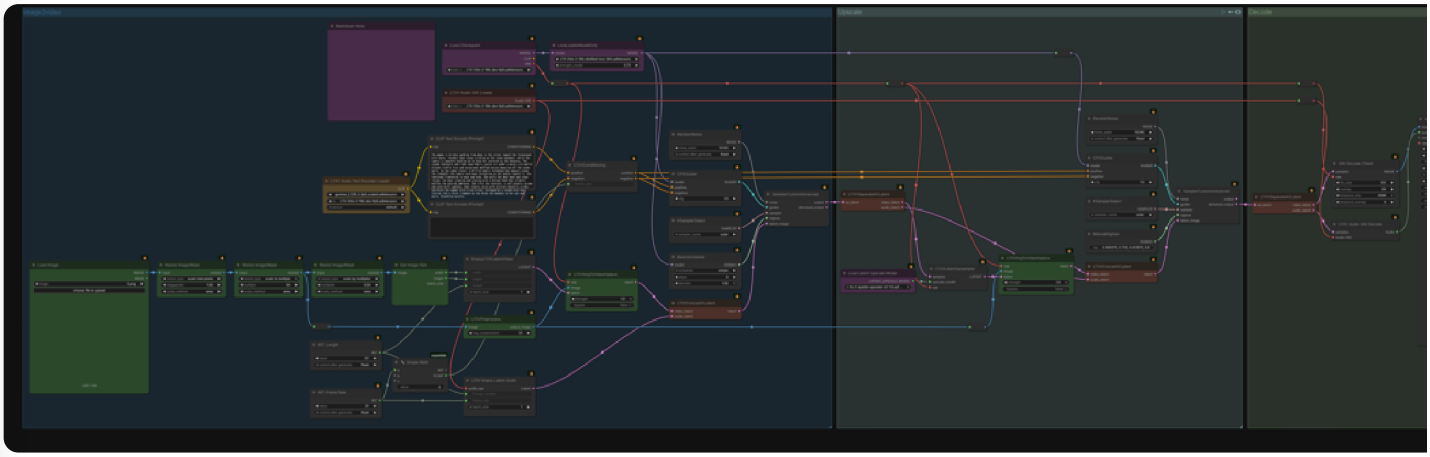
8 steps (distilled-lora)

> *As far as I tried, applying distilled-lora produces more stable generations.*
> *Therefore, for speed and stability, all subsequent workflows apply **distilled-lora** from the 1st stage.*

---

image2video

single-frame I2V



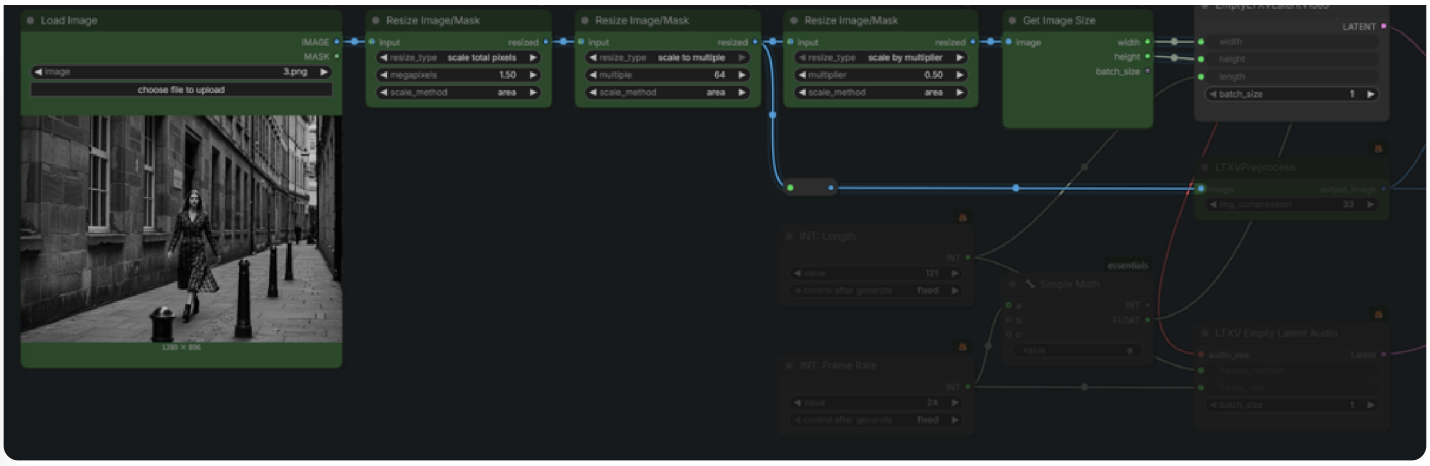
LTX-2_image2video_distilled_V2.json

The basic idea is "fix the 1st frame with input image and generate the rest".

For example, if creating a 121-frame video, the flow is roughly like this:

```
(1) Create a frame for 121 frames (8n+1)
    [ ⬜ ⬜ ⬜ ⬜ ⬜ ... ⬜ ]

(2) Overwrite only the 1st frame with input image
    [ 🖼️ ⬜ ⬜ ⬜ ⬜ ... ⬜ ]

(3) Generate the remaining 120 frames
    [ 🖼️ ✨ ✨ ✨ ✨ ... ✨ ]
```

Imagine the frames (✨) filling up consecutively starting from 🖼️ .



1. Resize Input Image (Create 2 versions)

- First, create a full-resolution version matching the final output resolution.
  - Resize to arbitrary size (here 1MP).
  - Width and height must be multiples of 64.
    - Since the 1st stage runs at 1/2 resolution, make it a multiple of 64 so it remains a multiple of 32 when halved.
- Next, for the 1st stage (half resolution), create a version with width/height halved from the above image.
  - Input this half-resolution width/height into `EmptyLTXVLatentVideo`.

2. Image Preprocessing

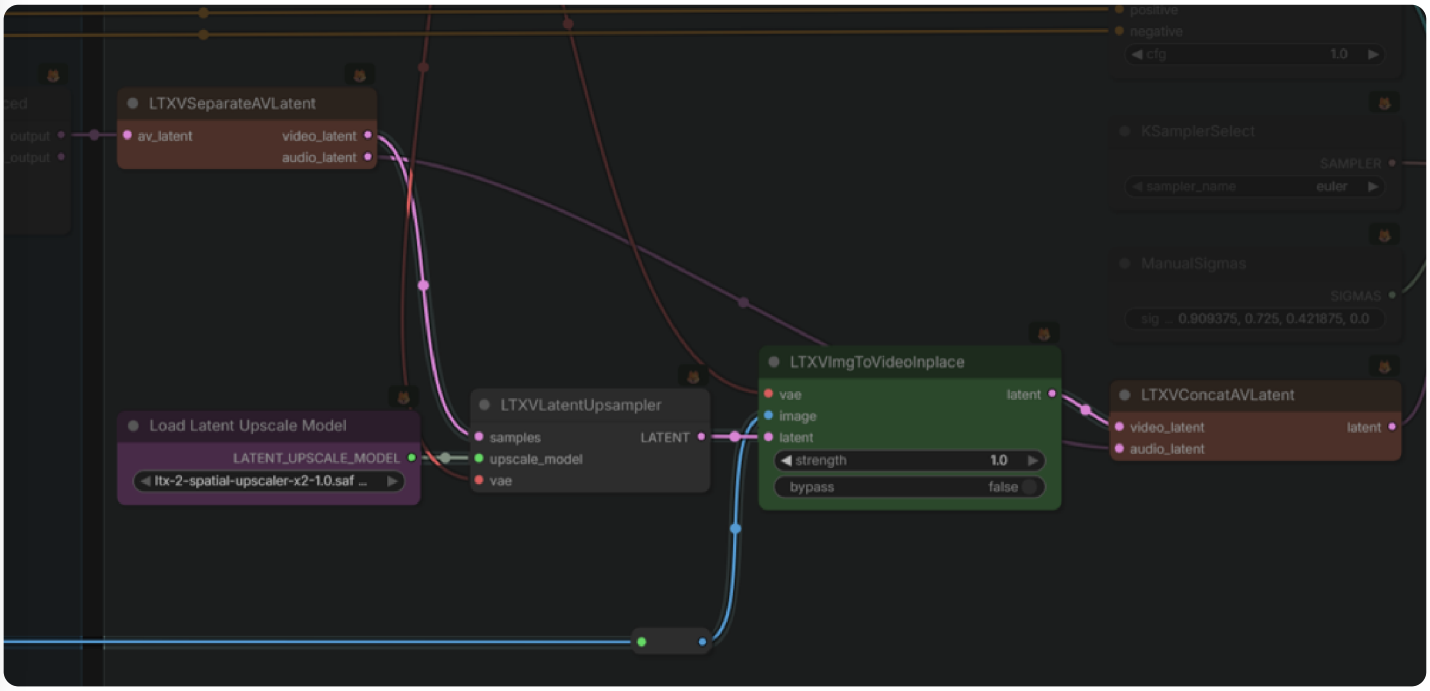
A characteristic from LTX-Video is that since video is slightly compressed and degraded compared to still images, using an image that is too clean may result in a video that doesn't

– To avoid this, intentionally degrade it to look like video compression with `LTXVPreprocess`.

3. LTXVImgToVideoInplace (Insert into 1st Stage)

This is the core of image2video.

– Insert the image as the 1st frame into the video latent of the 1st stage (half resolution).

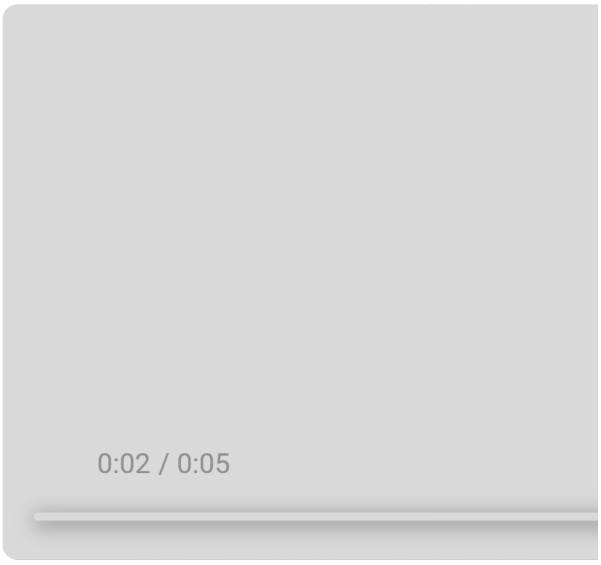


Insert the image into the 2nd stage as well.

- Make sure to connect this node **after** the spatial node.
- Set strength to `1.0` .
  - If you reduce this, the inserted image itself will behave like it's being image2image'd.
  - That's fine if you want it to blend in as a whole, but if you want to match the input image and 1st frame perfectly, set it to `1.0` .

Output Example

Input

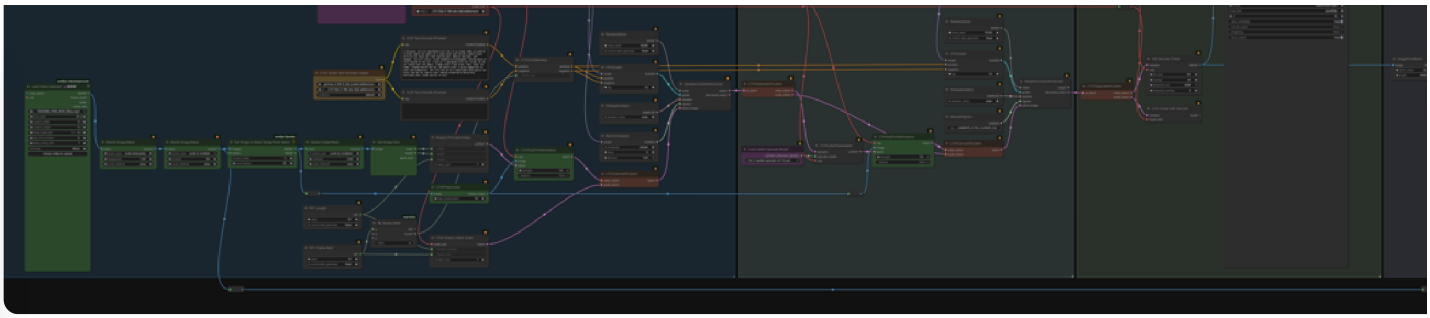


Output

---

multi-frame I2V

The previous image2video workflow can take not only a **single image** but also an **image batch (= video)** as input.
By applying this, you can create a workflow that uses the end of an arbitrary video as a "connector" and extends it further.



LTX-2_Extension_distilled.json

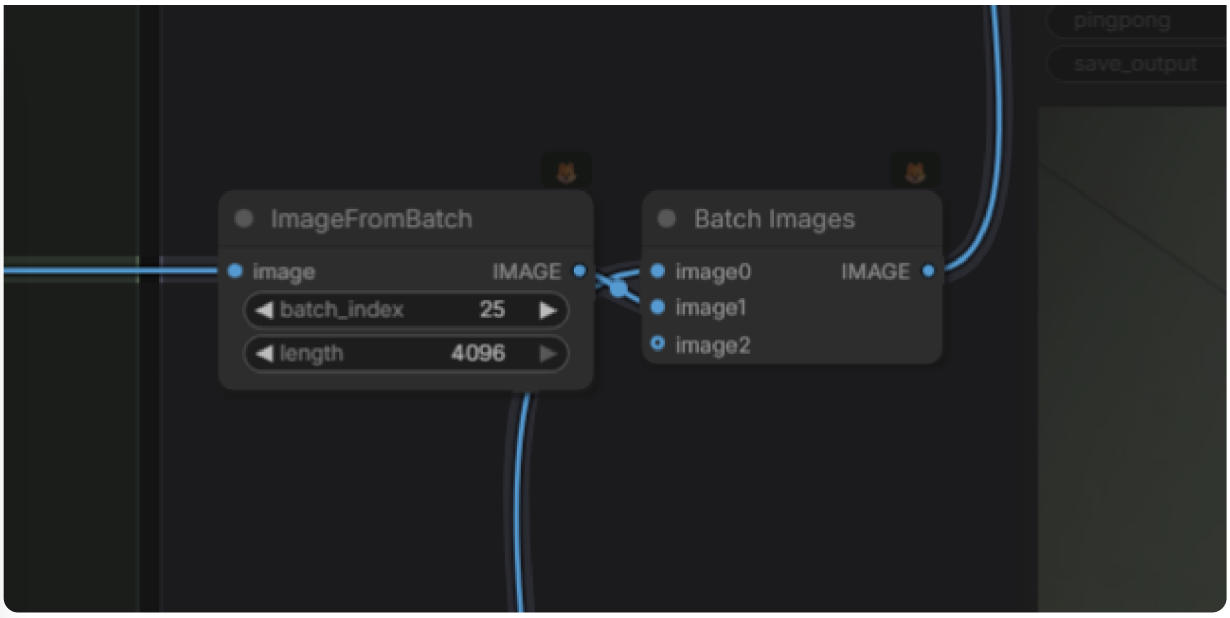
It takes the last few frames of the input video and generates the continuation.

```
(1) Input video (= image batch)
    [ 🖼️ 🖼️ 🖼️ 🖼️  ... 🖼️ 🖼️ 🖼️ ]

(2) Take N frames from the end (N = 8n+1)
    [ 🖼️ 🖼️ 🖼️ 🖼️... 🖼️ 🖼️ 🖼️ ]
                    └─── N ───┘

(3) Create a 121 frames slot and overwrite the beginning with N frames
    [ 🖼️ 🖼️ 🖼️ ⬜ ⬜ ⬜ ... ⬜ ]
      └── N ──┘

(4) Generate the remaining (121 - N frames) to make the continuation
    [ 🖼️ 🖼️ 🖼️ ✨ ✨ ✨ ... ✨ ]

(5) Delete the first N frames (as they duplicate the end of original video)
    [ ✨ ✨ ✨ ... ✨ ]

(6) Concatenate original video + continuation
    [ 🖼️ 🖼️ 🖼️ ... 🖼️] + [ ✨ ✨ ✨ ... ✨ ]
```

1. Get End Image Batch

Get the image batch that serves as the connector from the end of the input video.

- Enter an arbitrary number in `num_frames` of `Get Image or Mask Range From Batch` (must be 8n+1).
- Increasing N makes it easier to inherit the movement and atmosphere of the original video.
- However, since the generated section becomes 121 - N frames, increasing N makes the "continuation" shorter.



2. Concatenate Generated Video and Original Video

The generation result includes the "connector (N frames from end of original video)" at the beginning, but since this part duplicates the original video, delete it before concatenation.

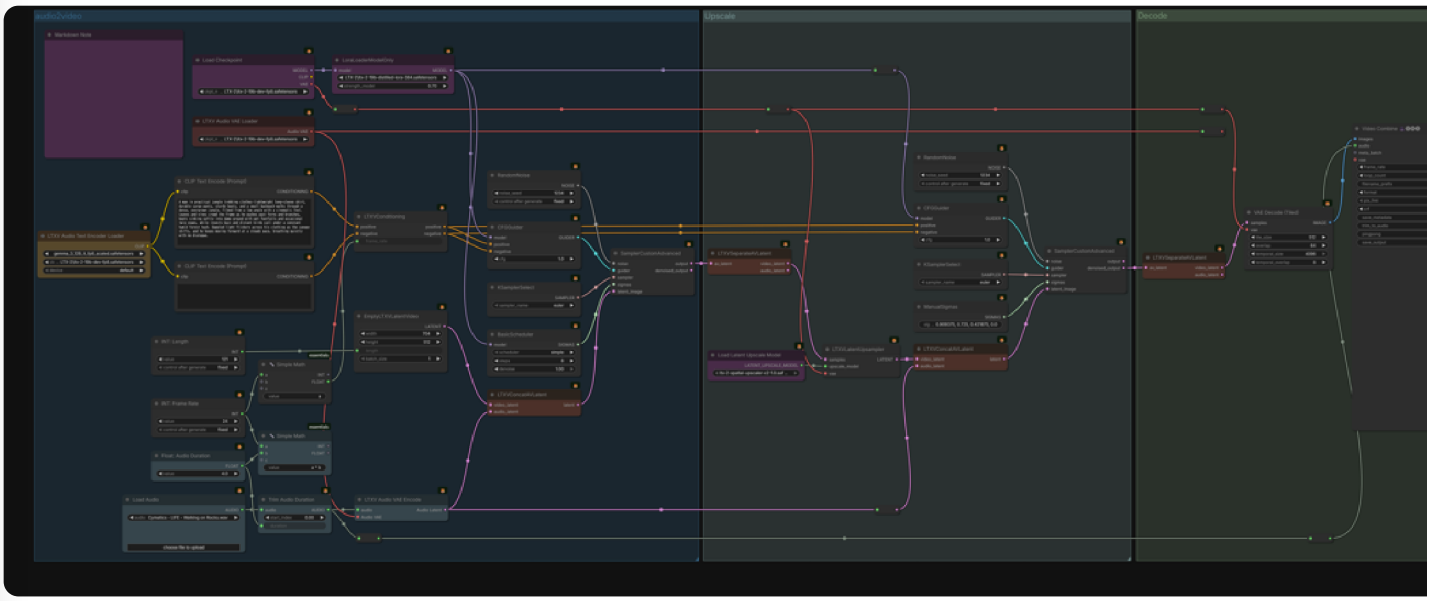
- Delete the first N frames of the generated video (25 frames in this example)
- Concatenate to the end of the original video

Output Example

Input Output

audio2video

Since LTX-2 is a model that handles "video + audio" simultaneously, you can configure it to take audio as input and create a video driven by the sound.



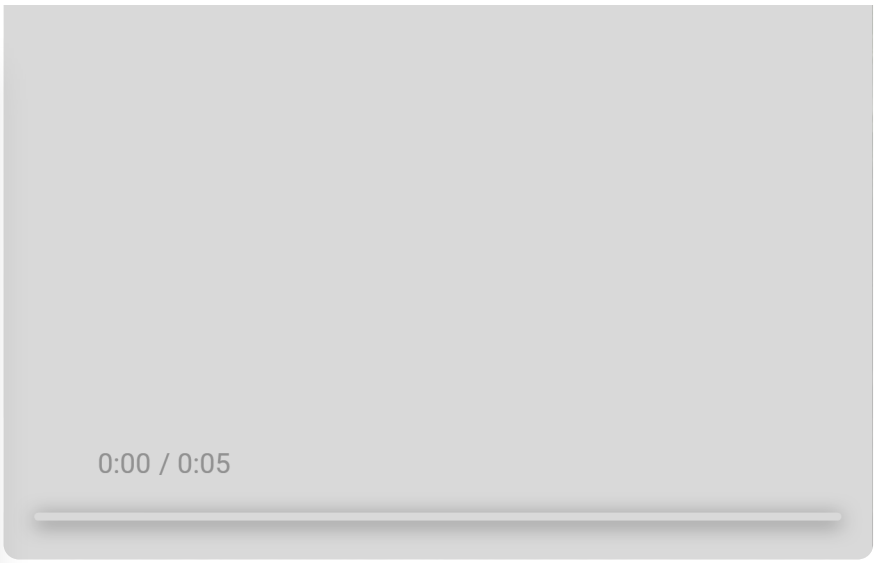
LTX-2_audio2video_distilled_V2.json

- Trim audio to appropriate length with `Trim Audio Duration` .
- Encode audio and connect to `LTXVConcatAVLatent` .
- Connect to the second stage `LTXVConcatAVLatent` as well.
- Use the input audio as is for the output video (do not use generated audio).

> 🚨 *If the audio length is **shorter** than the generated video length, the audio condition will not work. A video unrelated to the sound will be generated. Even if it's silent, you need to make it longer than the video being generated.*

I see workflows using `Set Latent Noise Mask` here, but the result is the same whether it's there or not.

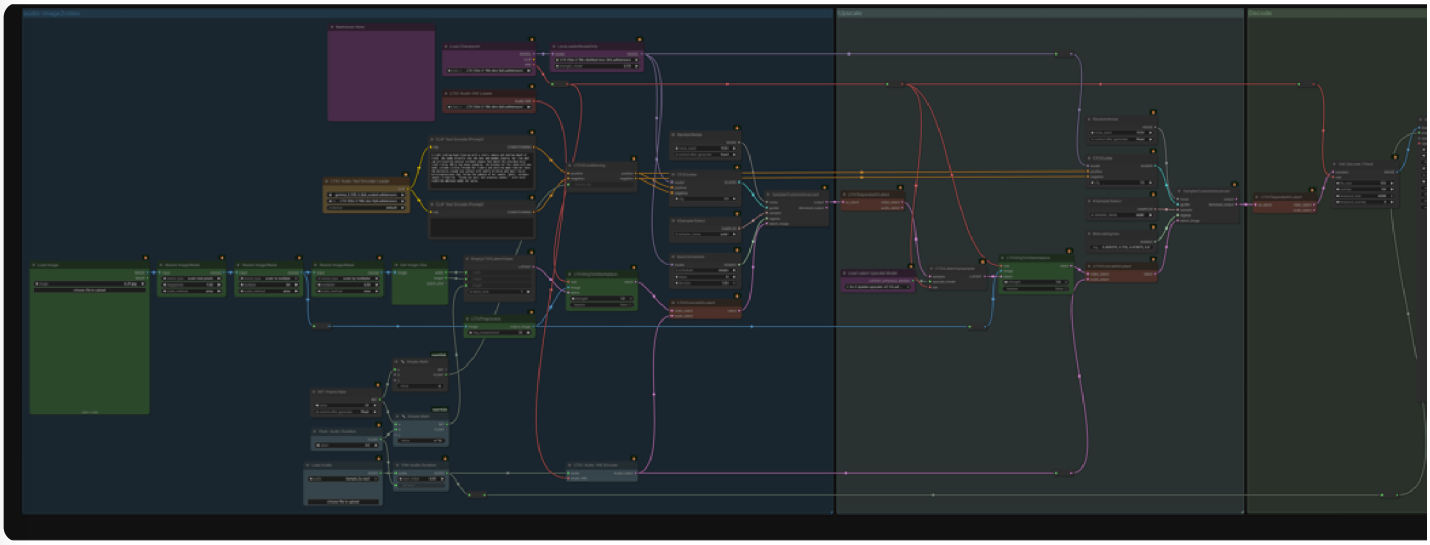
Output Example



---

audio-image2video

You can combine the above two. If you combine a face image with spoken audio, you can do something like a talking head. Let's try it.

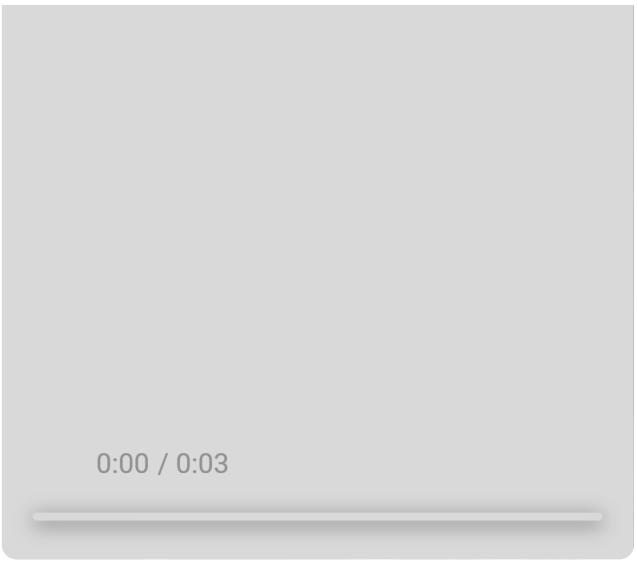


LTX-2_audio-image2video_distilled_V2.json

- Just combine the audio2video / image2video workflows.

Output Example

Input



Output

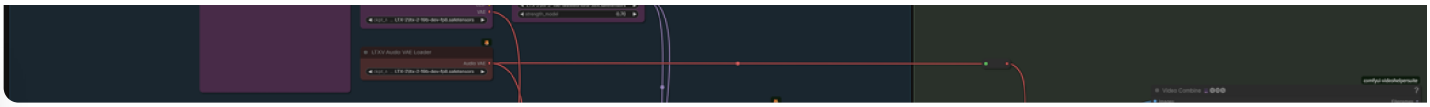
> *Actually, because the video didn't follow the dialogue very well, I put the dialogue in the prompt. There might be a better workflow.*

---

video2audio

Contrary to audio2video, you can also input a video and generate sound (sound effects or environmental sounds) that matches it.

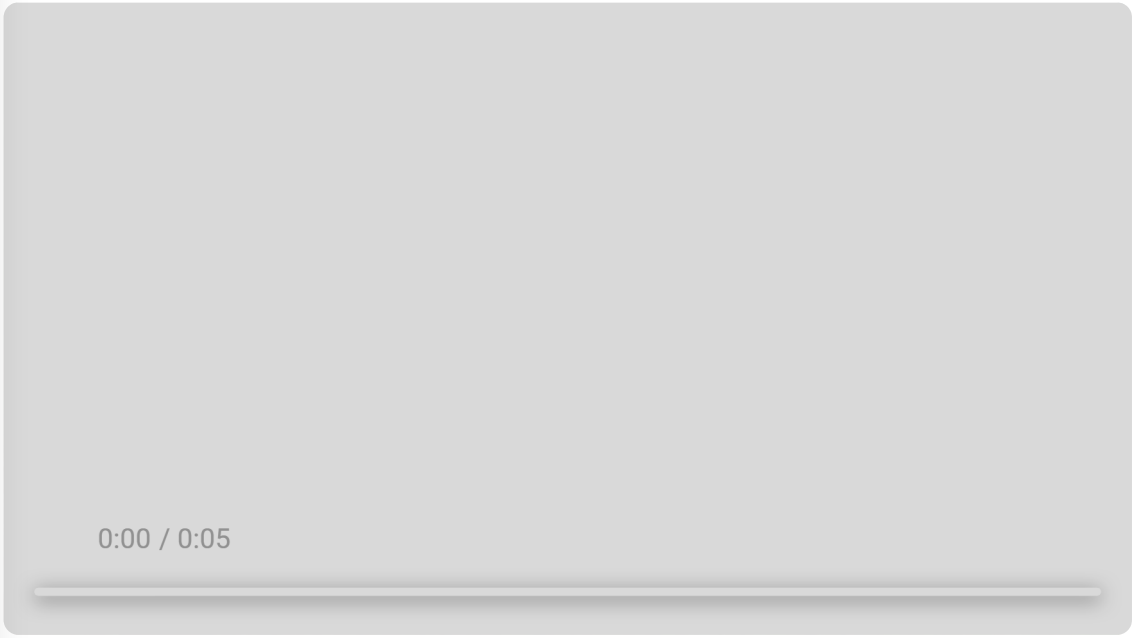
> *This task is unstable. Probably needs improvement.*



LTX-2_video2audio_distilled.json

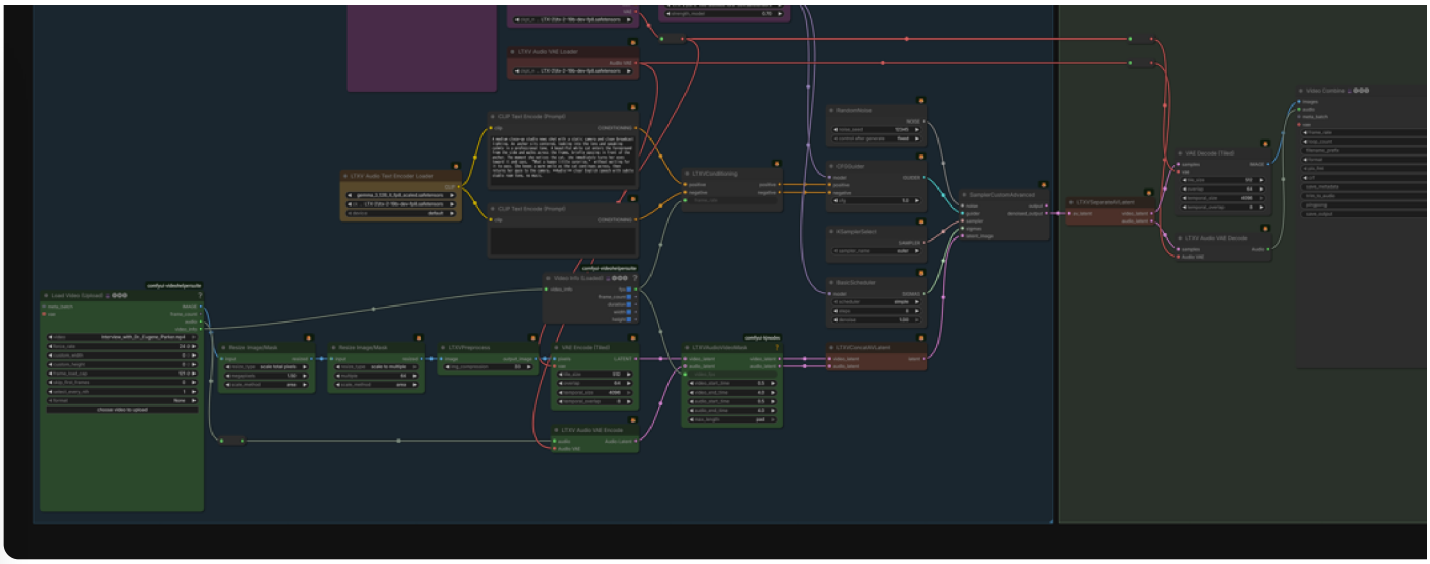
Output Example

Caution: Sound may be loud.



Temporal inpainting

This is temporal inpainting (= repairing only a part of the video). Think of it like VACE Extension.

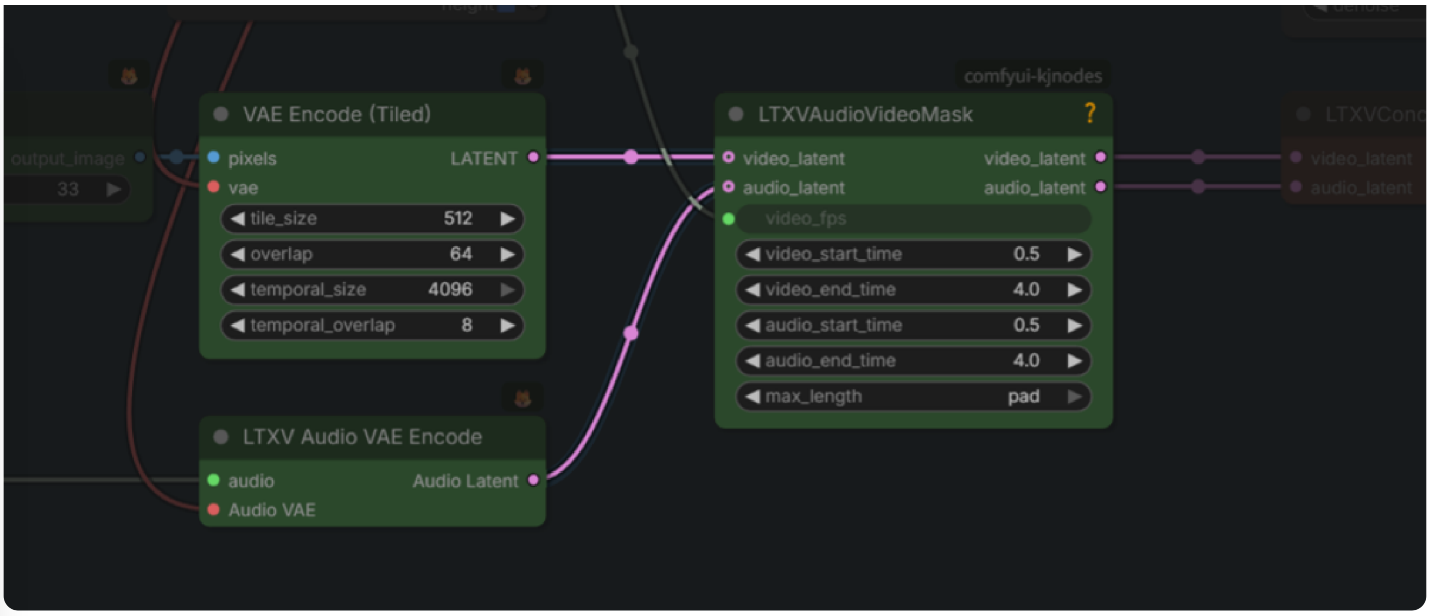


LTX-2_temporal-inpainting_distilled.json

Basically it is video2video. Mask only the "time range you want to remake" of the video and regenerate only that section.

```
(1) Input video (= existing video latent)
    [ 🖼️ 🖼️ 🖼️ 🖼️ 🖼️ 🖼️ 🖼️ 🖼️ 🖼️ 🖼️ ]

(2) Specify section to remake (start_time ~ end_time)
    e.g.: 2.0s ~ 4.0s
    [ 🖼️ 🖼️ | 🖼️ 🖼️ 🖼️ | 🖼️ 🖼️ 🖼️ ]
              ^           ^
         start_time    end_time

(3) Mask only the specified section
    [   0    0 |  1   1   1 |  0   0   0  ]
               └─── Mask ────┘

(4) Regenerate only the masked section
    [ 🖼️ 🖼️ | ✨ ✨ ✨ | 🖼️ 🖼️ 🖼️ ]
              └─ inpaint ─┘
```

> *Structurally, it is difficult to assemble a two-stage workflow (low resolution -> Hires.fix), so we generate at 1.5MP from the beginning.*



1. LTXVAudioVideoMask

Specify the time range you want to inpaint.

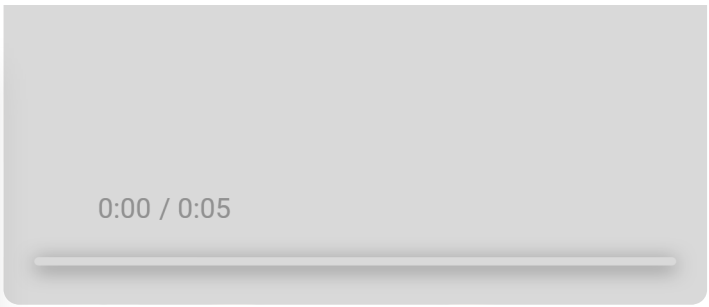
- `video_fps` : Basically set to same fps as input video
- `video_start_time` : Inpainting start (seconds)
- `video_end_time` : Inpainting end (seconds)
- `audio_start_time` / `audio_end_time` : Basically same as video, but by shifting them you can do "edit video only while keeping sound" or "edit sound only while keeping video"

Can also Extend

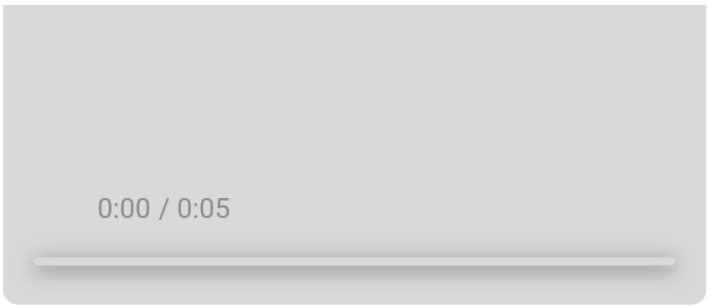
If you specify `end_time` beyond the length of the input video, the overlapping part is newly generated, resulting in extended video. e.g.: If input is "2 seconds"

- Remake 2.0s → 5.0s (= newly generate after 2 seconds to extend)
- `start_time = 2.0` / `end_time = 5.0`

Output Example



Input



Output

IC-LoRA

IC-LoRA creates video from control signals such as pose, depth map, edges, etc.

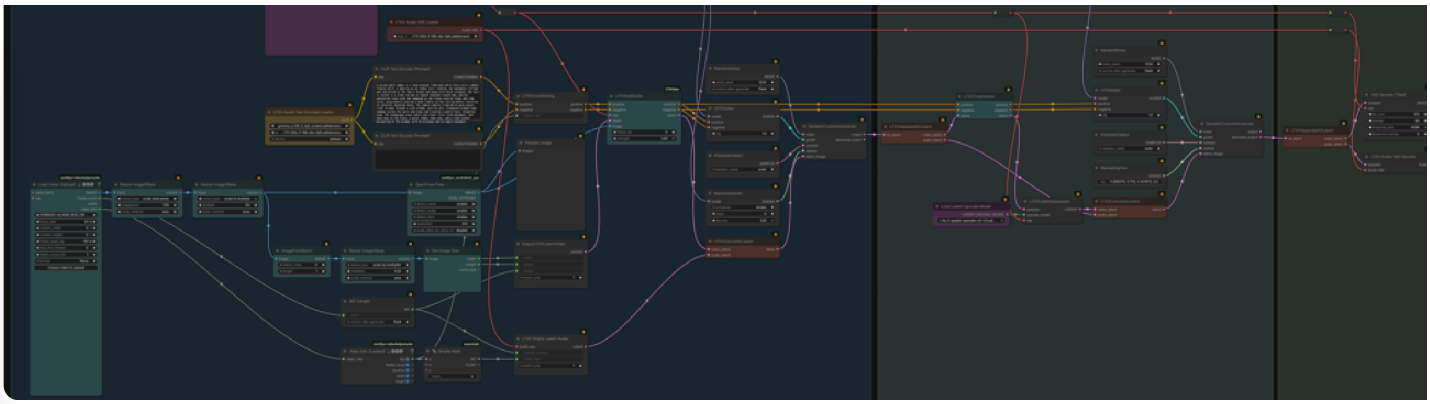
Model Download

- loras
  - ltx-2-19b-ic-lora-canny-control.safetensors
  - ltx-2-19b-ic-lora-depth-control.safetensors
  - ltx-2-19b-ic-lora-detailer.safetensors
  - ltx-2-19b-ic-lora-pose-control.safetensors

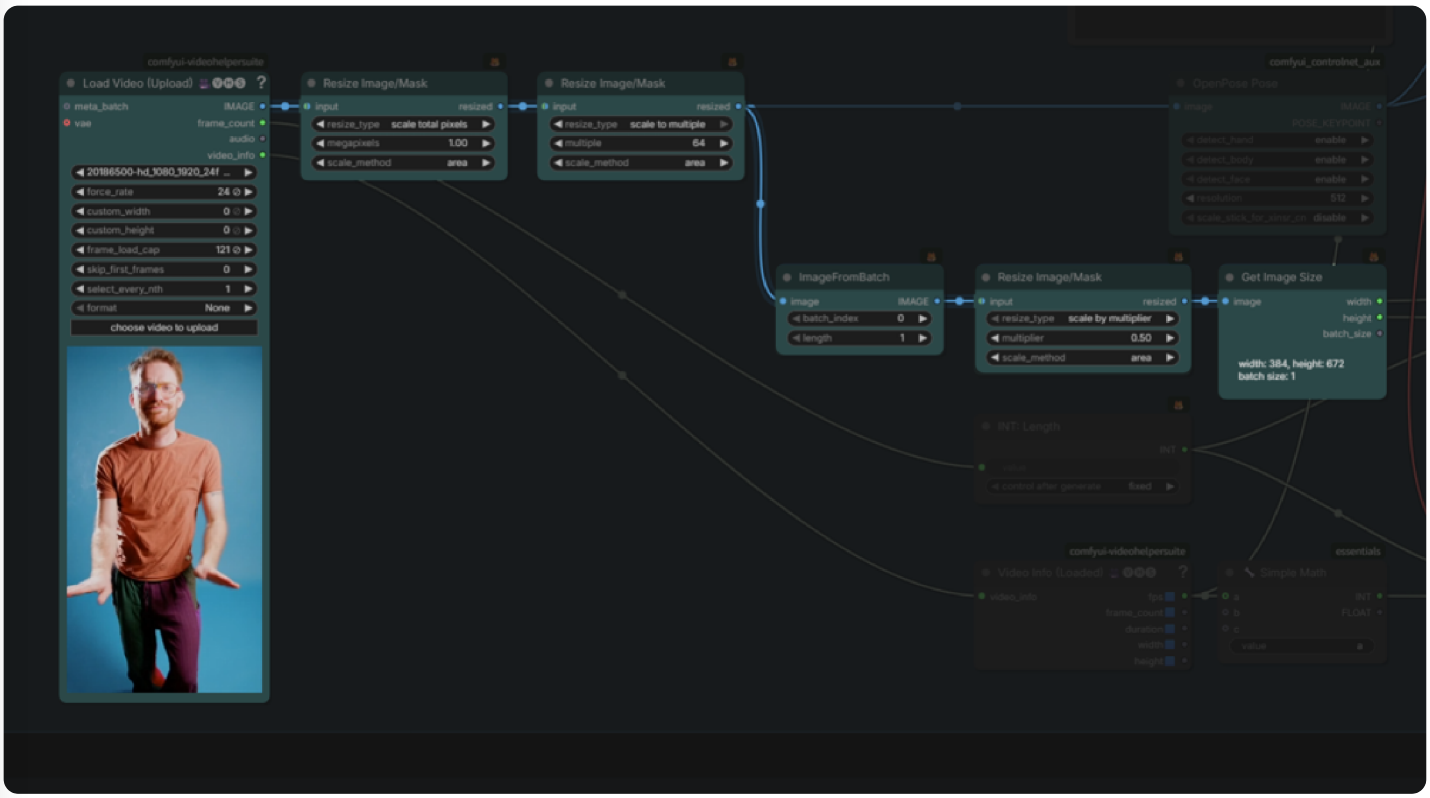
```
ComfyUI/
└── models/
    └── loras/
        ├── ltx-2-19b-ic-lora-canny-control.safetensors
        ├── ltx-2-19b-ic-lora-depth-control.safetensors
        ├── ltx-2-19b-ic-lora-detailer.safetensors
        └── ltx-2-19b-ic-lora-pose-control.safetensors
```

IC-LoRA (Pose)

Add control video based on text2video.



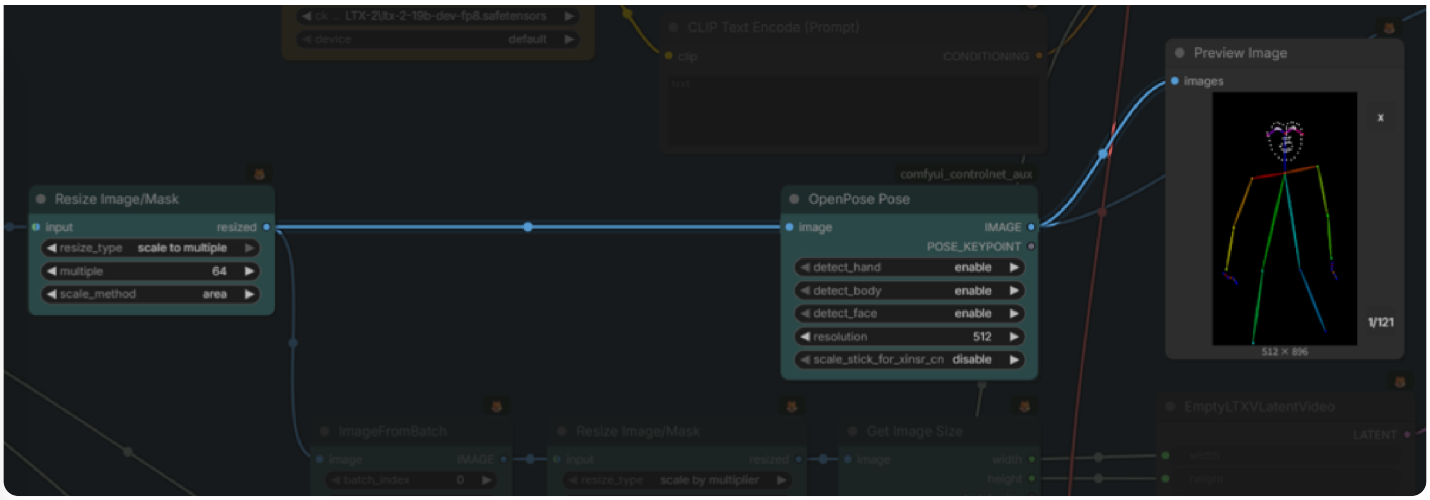
LTX-2_IC-LoRA(Pose)_distilled_V2.json



1. Resize Control Video

Align to the same ratio and resolution as the video to be generated.

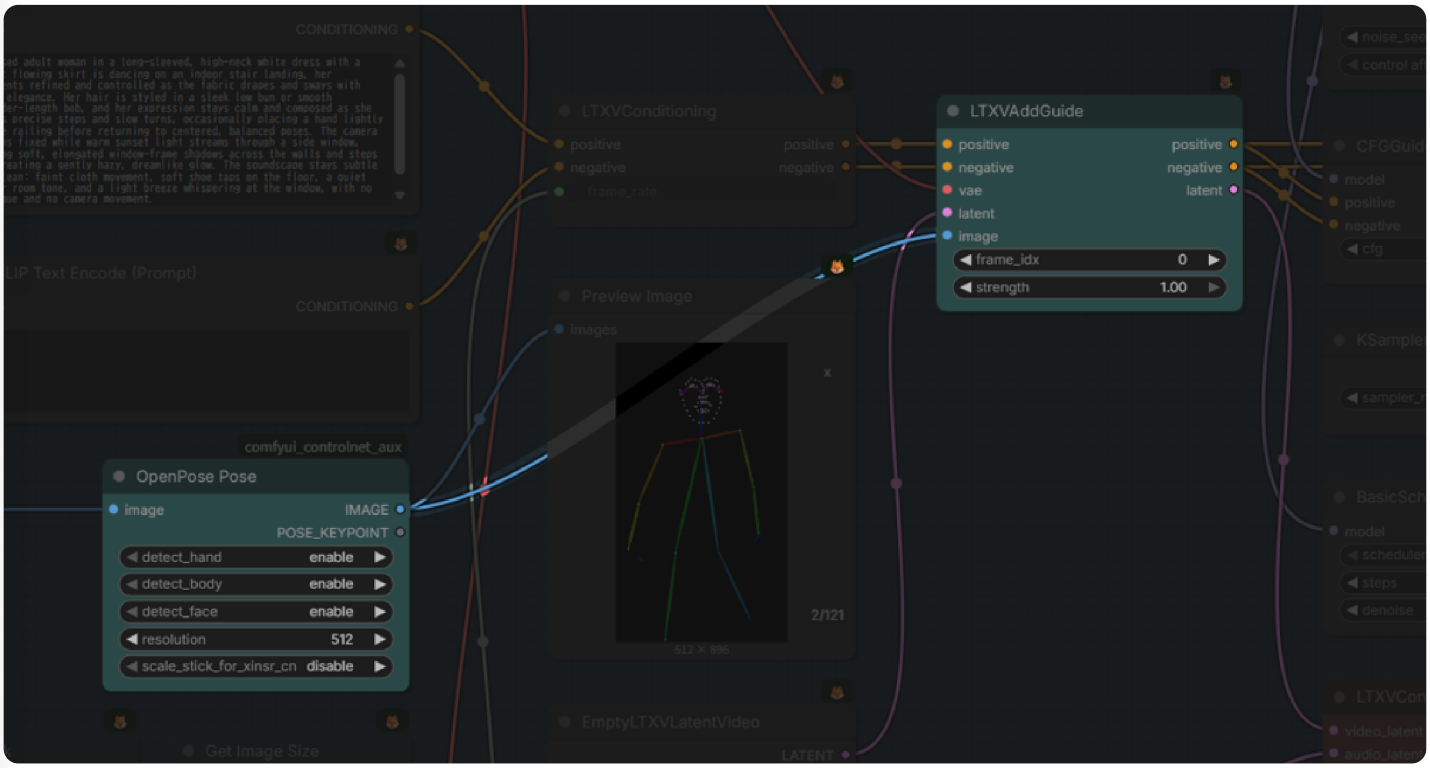
- Resize to arbitrary size (here 1.5MP).
- Width and height must be multiples of 64.
- Input the width/height of the image halved vertically and horizontally into `EmptyLTXVLatentVideo` .



2. Generate Pose Image

Create stick figure images from video.

- Extract pose with OpenPose or DWPose.



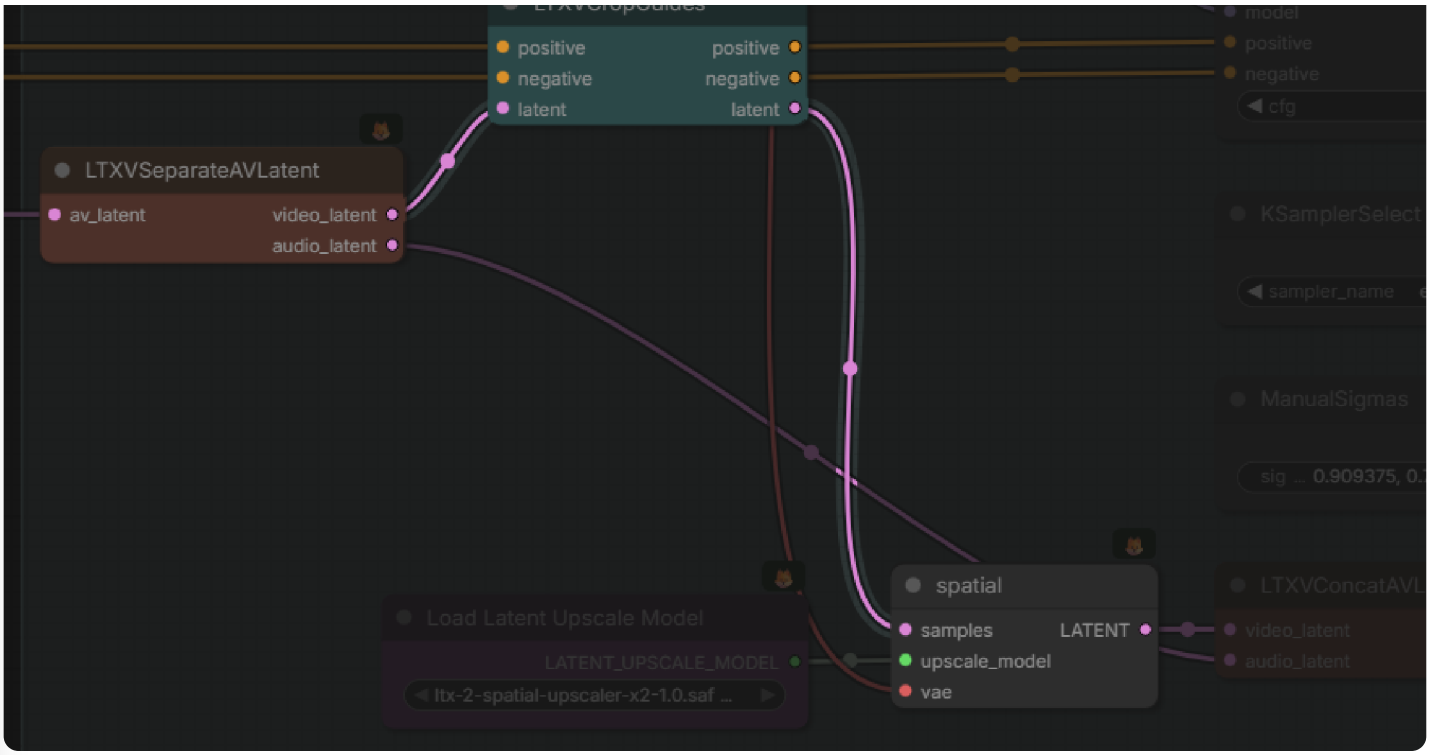
3. LTXVAddGuide

Put the control signal (pose video) into conditioning.

4. Apply IC-LoRA

Apply IC-LoRA (Pose this time) and sample.

- IC-LoRA is designed assuming `strength = 1.0` .
- In this workflow, IC-LoRA is applied only to the 1st stage sampling.
  - Making the 2nd stage focus on refining results in a cleaner video.



5. LTXVCropGuides

If you decode once after the 1st stage is finished, it's easy to understand, but the generated video is mixed with the pose video created earlier.

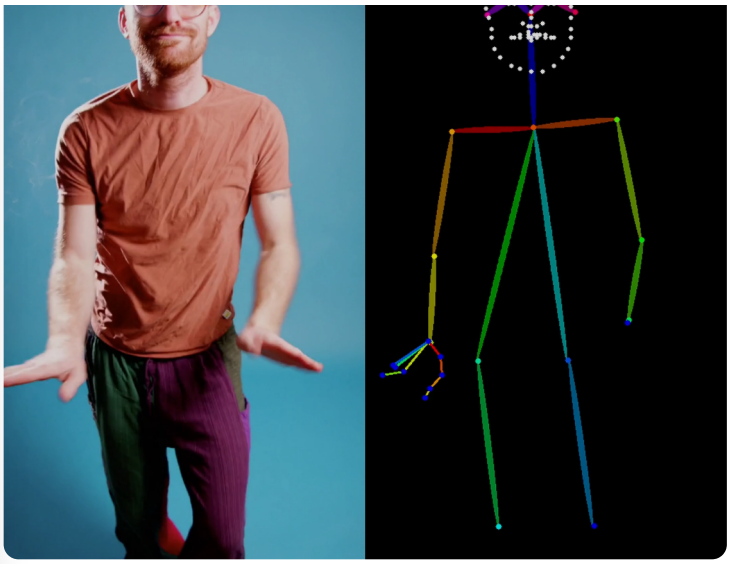
- Focus on the latter half: **Before LTXVCropGuides.mp4**

This is exactly how IC-LoRA works, but since it is unnecessary for the output, remove it before entering the 2nd stage.

- `LTXVCropGuides` is a node for removing control images from latent / conditioning.

> *You can use it in the same way by changing Pose Image / IC-LoRA to Canny / Depth. Note that using basically one type is recommended. (Applying Pose and Depth at the same time is not recommended.)*

Output Example

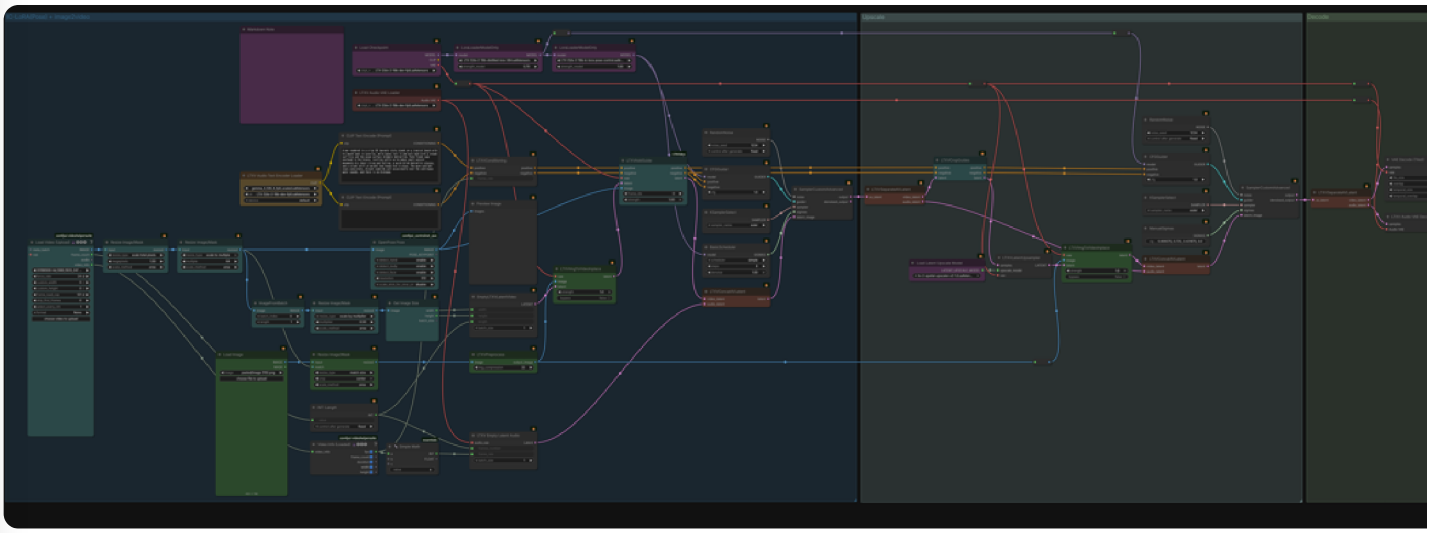


Input

Output

IC-LoRA (Pose) + image2video

You cannot stack multiple IC-LoRAs, but you can combine with image2video or audio2video.



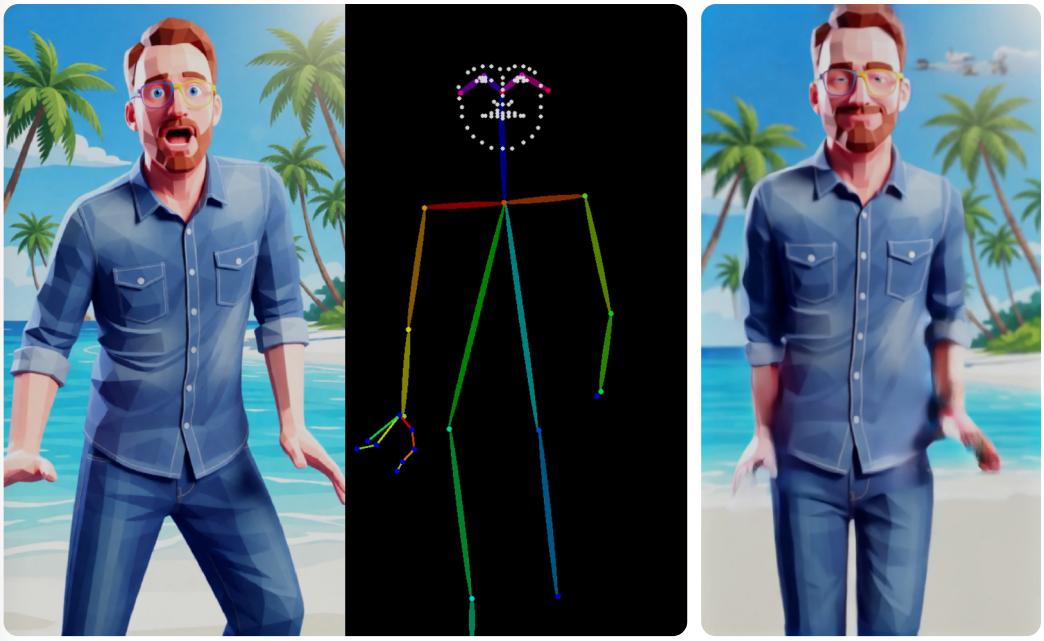
LTX-2_IC-LoRA(Pose)_image2video_distilled_V2.json

What it's doing is just combining IC-LoRA (Pose) above with image2video.

- Note that `LTXVAddGuide` is connected **after** `LTXVImgToVideoInplace` .

- This is strictly image2video, not reference2video like VACE.
  - Since the input image is "an image fixed as the 1st frame", if it deviates significantly from the 1st frame of the pose video, you won't get the expected video.
  - Create an "image close to the 1st frame of pose" with ControlNet or Qwen-Image-Edit etc. in advance.

Output Example



Input

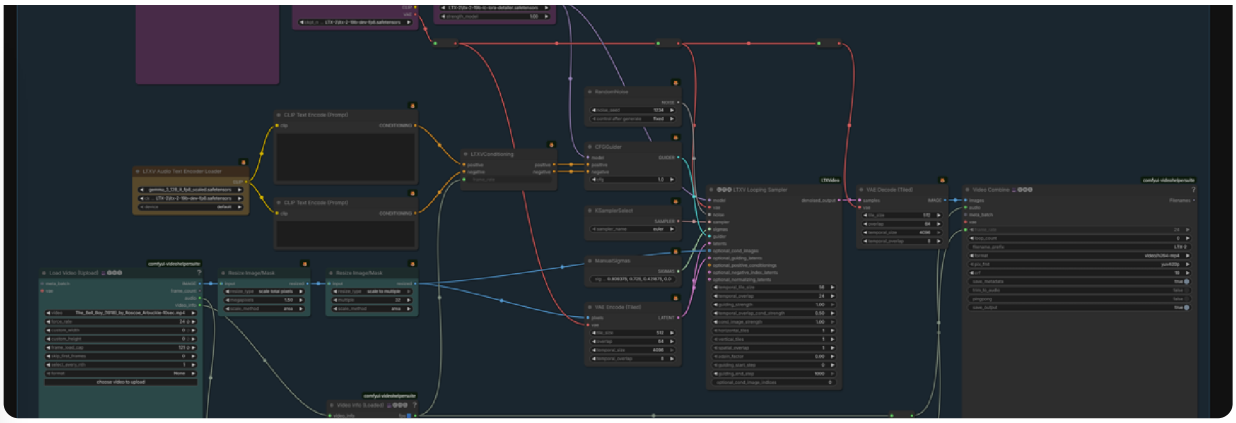
Output

IC-LoRA (Detailer)

IC-LoRA (Detailer) restores details and textures of low-resolution videos.

Install Custom Nodes

- ComfyUI-LTXVideo
- You can run it with just core nodes, but custom nodes are required to handle large resolutions / long duration videos.



LTX-2_IC-LoRA(Detailer)_V2.json

Basically it is video2video with IC-LoRA(Detailer) applied.

- 🟦 First, resize the input video to the desired final size.
- Use 🅛🅣🅧 `LTXV Looping Sampler` instead of `SamplerCustomAdvanced` .
  - This works like Ultimate SD upscale, processing time/space in tiles, allowing you to save VRAM.
  - In this workflow, only the time direction is tiled.
- It does not use distilled LoRA, but generates in 3 steps.

Output Example

Input

Output

- Prompting Guide
- LTX-2 Official Doc
- Lightricks/ComfyUI-LTXVideo/example_workflows
- Comfy.Org blog
