Here's a comprehensive guide to generating **Text-to-Video** using **AnimateDiff** with **Stable Diffusion**. This method allows you to create short animations from text prompts without requiring model fine-tuning.

Requirements

- **Hardware**: NVIDIA GPU with at least 8GB VRAM (10GB+ recommended for higher resolutions)
- **Software**: Windows or Linux
- **Stable Diffusion Web UI**: AUTOMATIC1111 v1.6.0 or later
- **Dependencies**:
  - `diffusers`
  - `transformers`
  - `accelerate`
  - `peft`
  - `gradio`
  - `ffmpeg` (for video processing)

□ Step-by-Step Setup

1. Install AUTOMATIC1111 Web UI

If you haven't already, set up the Stable Diffusion Web UI:

```
git clone https://github.com/AUTOMATIC1111/stable-diffusion-webui

cd stable-diffusion-webui

python launch.py
```

2. Install AnimateDiff Extension

Navigate to the **Extensions** tab in the Web UI:

- Go to **Install from URL**.
- Enter: `https://github.com/continue-revolution/sd-webui-animatediff`
- Click **Install** and restart the Web UI.

3. Download Motion Modules

Download the AnimateDiff motion modules:

- mm_sd_v15_v2.ckpt

Place the downloaded `.ckpt` file in:

stable-diffusion-webui/extensions/sd-webui-animatediff/model/

4. Configure AnimateDiff

In the Web UI:

- Go to the **txt2img** tab.
- Scroll down to the **AnimateDiff** section.
- Set the following parameters:
  - **Motion Module**: `mm_sd_v15_v2.ckpt`
  - **Enable AnimateDiff**: ☑ Yes
  - **Number of Frames**: 32 (for a 4-second video at 8 FPS)
  - **FPS**: 8
  - **Save Options**: Choose `mp4` or `gif` as desired

5. Generate the Animation

- **Prompt**: `((best quality)), ((masterpiece)), ((realistic)), long highlighted hair, cybergirl, futuristic silver armor suit, confident stance, high-resolution, living room, smiling, head tilted`
- **Negative Prompt**: `CyberRealistic_Negative-neg`
- **Steps**: 20
- **Sampler**: `DPM++ 2M Karras`
- **CFG Scale**: 10
- **Seed**: -1
- **Size**: 512×512

Click **Generate** to create your animation.

VideoLDM is an open-source, diffusion-based model for text-to-video generation built on top of latent diffusion models. It's more accessible for users with consumer GPUs compared to some huge models.

VideoLDM: Quick Intro + How to Get Started

What is VideoLDM?

- Based on **latent diffusion** like Stable Diffusion.
- Generates short video clips (a few seconds).
- Supports text prompts to create videos.
- Open source with code and demo notebooks.
- Developed by Ashish Kamath and team.

How to run VideoLDM locally (Basic setup)

---

1. Environment setup

git clone https://github.com/ashkamath/mdmm-video.git

cd mdmm-video

Create and activate conda env (recommended)

conda create -n videoldm python=3.9 -y

conda activate videoldm

Install requirements

pip install -r requirements.txt

2. Download pretrained weights

Weights might be provided via links in the repo README or huggingface.

https://github.com/ashkamath/mdmm-video

---

3. Run demo notebook

- Open `VideoLDM_demo.ipynb` in Google Colab or locally.
- The notebook walks through prompt input, video generation, and saving output.

Full Python Script with Gradio UI: `videoldm_gradio.py`

```
import torch
import gradio as gr
from video_diffusion.script_util import create_video_diffusion_model
import tempfile
import os
from PIL import Image

# Set device
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Path to your pretrained weights checkpoint
model_path = "models/mdmm-video-base.ckpt"

print("Loading VideoLDM model...")
model = create_video_diffusion_model(model_path=model_path)
model.to(device)
model.eval()
print("Model loaded.")

@torch.no_grad()
def generate_video(prompt, num_frames=16):
    """
    Generate video tensor from prompt using VideoLDM.
    Returns path to saved mp4 video.
    """
    # Generate video tensor (num_frames, C, H, W)
    video_tensor = model.sample_video(prompt, num_frames=num_frames)
```

```
    # Convert tensor to CPU and uint8 image frames
    video_tensor = video_tensor.cpu().clamp(0, 1)
    video_tensor = (video_tensor * 255).to(torch.uint8)

    # Save frames to temporary directory
    tmpdir = tempfile.mkdtemp()
    for i in range(video_tensor.shape[0]):
        frame = video_tensor[i].permute(1, 2, 0).numpy()
        img = Image.fromarray(frame)
        img.save(os.path.join(tmpdir, f"frame_{i:03d}.png"))

    # Use ffmpeg to create video
    video_path = os.path.join(tmpdir, "output.mp4")
    cmd = (
        f"ffmpeg -y -framerate 8 -i {tmpdir}/frame_%03d.png "
        f"-c:v libx264 -pix_fmt yuv420p {video_path}"
    )
    os.system(cmd)

    return video_path

# Build Gradio interface
iface = gr.Interface(
    fn=generate_video,
    inputs=[
        gr.Textbox(lines=2, placeholder="Enter a descriptive prompt here...",
label="Prompt"),
        gr.Slider(8, 32, value=16, step=1, label="Number of Frames")
    ],
    outputs=gr.Video(label="Generated Video"),
    title="VideoLDM Text-to-Video Demo",
    description="Generate short videos from text prompts using VideoLDM."
)

if __name__ == "__main__":
    iface.launch()
```

How to run:

1. Make sure you have VideoLDM repo cloned, dependencies installed, and weights downloaded.
2. Save above code to `videoldm_gradio.py`
3. Run:

```
python videoldm_gradio.py
```

4. The Gradio app opens in your browser; enter your prompt and generate videos.

What this does:

- Loads VideoLDM model once.
- Takes a text prompt and number of frames from UI.
- Generates video tensor.
- Saves frames as PNGs temporarily.
- Uses `ffmpeg` (make sure installed on your system) to stitch frames into MP4.
- Returns video file for playback in UI.

Here's a **step-by-step guide** to download and set up the **pretrained weights for VideoLDM** so you can run the model locally:

Step-by-Step Guide to Download VideoLDM Pretrained Weights

---

1. Clone the VideoLDM repository

Open your terminal or command prompt and run:

```
git clone https://github.com/ashkamath/mdmm-video.git
cd mdmm-video
```

---

2. Check the repository README for weight links

The VideoLDM repo README or associated docs typically include links to pretrained weights.

- Visit: https://github.com/ashkamath/mdmm-video
- Look for model checkpoints or download links in the README or wiki sections.
- Often, weights are hosted on **Hugging Face** or **Google Drive**.

---

3. Download pretrained weights

Currently, the official weights for VideoLDM are available on **Hugging Face** at this repo:

- https://huggingface.co/ashkamath/mdmm-video

Common checkpoints include:

- `mdmm-video-base.ckpt` or similar

Download method (using `wget` or browser):

Example using `wget` in terminal (replace URL with actual checkpoint link):

wget https://huggingface.co/ashkamath/mdmm-video/resolve/main/mdmm-video-base.ckpt -P models/

f you prefer GUI, download manually from the Hugging Face page and place the file in the `models/` folder inside your cloned repo.

---

4. Place the weights correctly

Create a folder named `models` inside your `mdmm-video` directory if not already present:

```
mkdir -p models
```

Move or save the checkpoint `.ckpt` file into this folder.

5. Modify code to load weights

In your Python code, make sure you point the model loader to this checkpoint path, for example:

```
model_path = "models/mdmm-video-base.ckpt"
model = create_video_diffusion_model(model_path=model_path)
```

Here is the **full complete Python script** for your VideoLDM text-to-video Gradio app — **including loading the pretrained weights** from a checkpoint file you downloaded and placed in `models/mdmm-video-base.ckpt`.

Make sure you have:

- Cloned the repo `mdmm-video`
- Downloaded the checkpoint and saved it as `models/mdmm-video-base.ckpt`
- Installed dependencies (`pip install -r requirements.txt`)
- Installed `ffmpeg` on your system

videoldm_gradio.py:-

```
import torch
import gradio as gr
from video_diffusion.script_util import create_video_diffusion_model
import tempfile
import os
from PIL import Image

# Set device
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Path to your pretrained weights checkpoint
model_path = "models/mdmm-video-base.ckpt"

print("Loading VideoLDM model...")
model = create_video_diffusion_model(model_path=model_path)
model.to(device)
model.eval()
print("Model loaded.")

@torch.no_grad()
def generate_video(prompt, num_frames=16):
    """
    Generate video tensor from prompt using VideoLDM.
    Returns path to saved mp4 video.
    """
```

```
    # Generate video tensor (num_frames, C, H, W)
    video_tensor = model.sample_video(prompt, num_frames=num_frames)

    # Convert tensor to CPU and uint8 image frames
    video_tensor = video_tensor.cpu().clamp(0, 1)
    video_tensor = (video_tensor * 255).to(torch.uint8)

    # Save frames to temporary directory
    tmpdir = tempfile.mkdtemp()
    for i in range(video_tensor.shape[0]):
        frame = video_tensor[i].permute(1, 2, 0).numpy()
        img = Image.fromarray(frame)
        img.save(os.path.join(tmpdir, f"frame_{i:03d}.png"))

    # Use ffmpeg to create video
    video_path = os.path.join(tmpdir, "output.mp4")
    cmd = (
        f"ffmpeg -y -framerate 8 -i {tmpdir}/frame_%03d.png "
        f"-c:v libx264 -pix_fmt yuv420p {video_path}"
    )
    os.system(cmd)

    return video_path

# Build Gradio interface
iface = gr.Interface(
    fn=generate_video,
    inputs=[
        gr.Textbox(lines=2, placeholder="Enter a descriptive prompt here...",
label="Prompt"),
        gr.Slider(8, 32, value=16, step=1, label="Number of Frames")
    ],
    outputs=gr.Video(label="Generated Video"),
    title="VideoLDM Text-to-Video Demo",
    description="Generate short videos from text prompts using VideoLDM."
)

if __name__ == "__main__":
    iface.launch()
```

How to run:

1. Save this as `videoldm_gradio.py` inside the `mdmm-video` repo folder.
2. Make sure checkpoint is at `models/mdmm-video-base.ckpt`.
3. Run:

```
python videoldm_gradio.py
```

4. The Gradio UI should open in your browser.
5. Enter a prompt, select number of frames, click generate, and watch your video!