# Guida al Progetto AI Influencer su Windows11 (WSL2 + GPU)

## 1. Prerequisiti (WSL2 + Docker + driver NVIDIA + GPU nel container)

* **HW minimo:** Windows11 aggiornato, GPU NVIDIA (es. RTX 4070 con almeno 12 GB VRAM), almeno 16 GB di RAM e 20 GB di spazio libero.
* **SW minimo:** Attiva WSL2 (es. Ubuntu 22.04) e installa Docker Desktop con integrazione WSL attivata. Installa driver NVIDIA aggiornati (Build 495+ per WSL [NVIDIA WSL user guide](https://docs.nvidia.com/cuda/wsl-user-guide/index.html)).
* **Comandi di setup:** Apri PowerShell come amministratore e:
* wsl --install -d Ubuntu # installa Ubuntu WSL2  
  wsl --set-version Ubuntu 2 # assicura WSL2
* In Ubuntu WSL2:
* # Installa Docker (se non usi Docker Desktop)  
  curl -fsSL https://get.docker.com -o get-docker.sh  
  sudo sh get-docker.sh  
  sudo usermod -aG docker $USER # aggiunge utente al gruppo docker  
  # Installa NVIDIA Container Toolkit per GPU  
  distribution=$(. /etc/os-release;echo $ID$VERSION\_ID)  
  curl -s -L https://nvidia.github.io/libnvidia-container/gpgkey | sudo apt-key add -  
  curl -s -L https://nvidia.github.io/libnvidia-container/experimental/${distribution}/libnvidia-container.list | sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list  
  sudo apt-get update  
  sudo apt-get install -y nvidia-container-toolkit  
  sudo systemctl restart docker
* **Verifica GPU in Docker:** Esegui ad esempio il container CUDA NVIDIA:
* docker run --rm --gpus all nvidia/cuda:11.8.0-base-ubuntu22.04 nvidia-smi
* Output atteso: elenco della tua GPU (RTX 4070) e driver [[1]](https://gist.github.com/atinfinity/f9568aa9564371f573138712070f5bad#:~:text=docker%20run%20,smi). Errori comuni: “no gpu devices found” se il container toolkit non è installato, “oci runtime error” se Docker non è in modalità WSL2/linux.
* **Debug:** Se nvidia-smi non trova la GPU, verifica i driver Windows installati; prova nvidia-smi direttamente in WSL (deve funzionare con i driver WSL). Verifica che Docker Desktop stia usando i container Linux (imposta in Settings→General).

## 2. Installazione del progetto “ai-influencer-hybrid-pro” da ZIP

* **Prerequisiti SW:** Python 3.8+ (installato in WSL), pip. Oppure utilizzo di Docker se il progetto include Dockerfile.
* **Comandi:** Copia il file ZIP in WSL (ad esempio nella home), poi:
* cd ~  
  unzip ai-influencer-hybrid-pro.zip -d ai\_influencer  
  cd ai\_influencer  
  ls # controlla struttura (es. requirements.txt, scripts/, etc.)
* Se c’è requirements.txt ed è un’app Python standalone:
* python3 -m venv venv  
  source venv/bin/activate  
  pip install -r requirements.txt
* Se è fornito un Dockerfile, puoi creare l’immagine e avviare il container:
* docker build -t ai\_influencer:latest .  
  docker run --rm --gpus all -v ~/ai\_influencer/data:/data -p 8000:8000 ai\_influencer:latest
* **Struttura cartelle:** Dopo unzip potresti avere: ai\_influencer/ con sottocartelle (src/, models/, assets/), file Dockerfile, requirements.txt, README.md. Mostra screenshot della cartella (es. tree o ls -R).
* **Output atteso:** Se l’app fornisce un’interfaccia web, dovrebbe essere raggiungibile su http://localhost:8000 (o altra porta specificata). Se è una CLI, dovrebbe esserci un comando principale (python main.py). Controlla messaggi di avvio nell’output del terminale.
* **Errori comuni:** Mancata dipendenza Python (pip fallisce, verifica Python3), errori di permessi (esegui con sudo se serve), porte occupate (modifica -p in docker run).
* **Debug:** Controlla la documentazione interna del progetto. Usa docker logs <container> per output container. Assicurati di eseguire tutto in Linux (WSL) e non mescolare percorsi Windows/Linux (usa percorsi /mnt/... se necessario).

## 3. Inserimento immagini iniziali e generazione massiva con Leonardo.ai (API)

* **Prerequisiti:** Account Leonardo.ai (piano gratuito disponibile, con crediti giornalieri) e chiave API. Leonardo.ai è un servizio SaaS gratuito fino a certi limiti; la licenza free consente uso commerciale non esclusivo delle immagini[[2]](https://leonardo.ai/pricing/#:~:text=Free%20tier%20users%3A%C2%A0Leonardo,only%20as%20enabled%20by%20a).
* **Comandi:** Imposta la chiave API come variabile ambiente e usa curl o uno script Python. Esempio curl:
* export LEO\_KEY=TUO\_API\_KEY  
  curl --request POST \  
   --url https://cloud.leonardo.ai/api/rest/v1/generations \  
   --header 'Authorization: Bearer '$LEO\_KEY \  
   --header 'Content-Type: application/json' \  
   --data '{"modelId":"<MODEL\_ID>","prompt":"ritratto fotorealistico di influencer cyberpunk","width":1024,"height":1024}'
* Questo invia una richiesta di generazione immagine[[3]](https://docs.leonardo.ai/docs/generate-your-first-images#:~:text=curl%20,32b51c1f67d3). Leonardo restituisce un JSON con link all’immagine. Per generazioni massive, esegui in loop su prompt o usa script Python che effettua tante chiamate. Puoi anche usare il parametro “controlnets” per guida da immagine iniziale[[4]](https://docs.leonardo.ai/docs/generate-images-using-image-to-image-guidance#:~:text=Sample%20Request).
* **Immagini iniziali:** Per “style reference”, carica le tue immagini tramite l’API (endpoints upload) e usa l’initImageId nel campo controlnets[[4]](https://docs.leonardo.ai/docs/generate-images-using-image-to-image-guidance#:~:text=Sample%20Request). In alternativa, puoi allegare un’immagine come URL o base64 nel payload.
* **Output atteso:** JSON contenente URL o base64 dell’immagine generata. Scarica le immagini (es. con wget) e salvale in una cartella leonardo\_outputs/. Ad es.: leonardo\_out\_0001.png, ecc.
* **Controllo qualità preliminare:** Rinomina file ordinatamente (e.g. con uno script bash), e filtra subito quelle inutili (p.es. molto scure o rotte).
* **Errori comuni:** “401 Unauthorized” se API key errata; “429 Too Many Requests” se superi il limite giornaliero; formattazione JSON invalida (verifica virgolette e virgole).
* **Suggerimenti di debug:** Testa prima con un prompt semplice e verifica il JSON. Registra i prompt usati. Verifica su dashboard Leonardo se i job sono registrati. Se l’API fallisce, prova a generare manualmente per capire limiti.

## 4. Verifica e controllo qualità (face-similarity, blur)

* **Obiettivo:** Filtrare immagini fuori tema (senza volto chiaro o sfocate).
* **Comandi:** Puoi scrivere uno script Python o una pipeline in Bash. Ad esempio, usando OpenCV per blur e face\_recognition per volti:
* import cv2, face\_recognition  
  img = cv2.imread('leonardo\_out\_0001.png')  
  gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
  lap = cv2.Laplacian(gray, cv2.CV\_64F).var()  
  print("Varianza Laplaciano:", lap)
* Se la varianza lap < soglia (e.g. 100), considerare sfocata[[5]](https://www.geeksforgeeks.org/computer-vision/how-to-check-for-blurry-images-in-your-dataset-using-the-laplacian-method/#:~:text=,var). Per volti:
* face\_locations = face\_recognition.face\_locations(img)  
  if not face\_locations:  
   print("Nessun volto riconosciuto")  
  else:  
   enc = face\_recognition.face\_encodings(img, known\_face\_locations=face\_locations)  
   # Confronta con enc.codifica di riferimento se necessario
* In alternativa, face\_recognition.compare\_faces([ref\_enc], unknown\_enc) per similarità[[6]](https://pypi.org/project/face-recognition/#:~:text=import%20face_recognition%20known_image%20%3D%20face_recognition.load_image_file%28,unknown.jpg).
* **Output atteso:** Dopo filtraggio, ottieni solo immagini con volti nitidi. Ad esempio, salva gli scarti in discarded/.
* **Errori comuni:** Il modello può non trovare un volto (imposta soglia di confidenza). Per blur, una soglia troppo alta classifica tutto come sfocato.
* **Suggerimenti di debug:** Adatta la soglia di varianza Laplaciano osservando alcuni esempi con print(lap). Usa face\_recognition.face\_locations per controllare la presenza di volti prima di codifiche. Se trova più volti, decidere come agire (ad esempio seleziona il più grande).

## 5. Augment + caption + metadata

* **Augmentation:** Aumenta i dati di training con trasformazioni. Installa librerie Python come albumentations o usa torchvision.transforms. Comandi esempio:
* pip install albumentations opencv-python
* Esempio Python (rota, flip, ruota colori, ecc.). Otterrai varianti di ogni immagine salvandole in augmented/.
* **Captioning:** Genera una didascalia per ogni immagine. Ad esempio usa il modello BLIP (Huggingface). Scarica il codice BLIP (es. [image-captioning](https://github.com/cobanov/image-captioning)). Esempio comando:
* git clone https://github.com/cobanov/image-captioning  
  cd image-captioning  
  pip install -r requirements.txt  
  python inference.py -i ~/ai\_influencer/images --batch 8 --gpu 0
* Il comando di esempio genera caption con GPU[[7]](https://github.com/cobanov/image-captioning#:~:text=Example). Otterrai un file con testo descrittivo per ogni immagine.
* **Metadata:** Crea un CSV/JSON con colonne filename, caption, tags, LoRA\_name, ecc. Puoi includere dati di similarità volto, caratteristiche (età/genero se rilevabili con *DeepFace*), punteggio blur, ecc. Questo servirà come dataset per il training.
* **Output atteso:** Cartelle images/, augmented/ con immagini; file captions.txt o data.json con le descrizioni; eventualmente metadata.csv con etichette.
* **Errori comuni:** BLIP può esaurire la VRAM per batch troppo grandi; prova batch 1 se OOM. Assicurati che le immagini siano nel formato atteso (RGB, giuste dimensioni).
* **Suggerimenti di debug:** Testa il modello caption su poche immagini. Controlla che i font (in BLIP) siano supportati. Per debug visivo, confronta alcune immagini originali e corrispondenti caption.

## 6. Training di un LoRA SDXL con kohya-ss (via Docker)

* **Prerequisiti HW/SW:** GPU >12 GB VRAM. SDXL è pesante; potresti dover ridurre batch\_size (es. 1–2) o usare precisione fp16. Assicurati di avere spazio su disco per checkpoints (diversi GB).
* **Comandi (Docker):** Scarica il repository [kohya\_ss](https://github.com/bmaltais/kohya_ss). Puoi usarne il Dockerfile o uno predefinito. Esempio generic:
* # Dopo aver creato Dockerfile (vedi doc) o usato immagine esistente  
  docker build -t kohya\_ss\_image .  
  docker run --rm --gpus all \  
   -v ~/kohya\_data:/data \  
   -v ~/kohya\_models:/models \  
   kohya\_ss\_image \  
   python train\_network.py \  
   --pretrained\_model\_name\_or\_path /models/SDXL.ckpt \  
   --train\_data\_dir /data/images \  
   --resolution 1024 \  
   --output\_dir /models/output\_lora \  
   --save\_model\_as safetensors \  
   --network\_alpha 128
* In questo esempio, i dati sono in /data/images, il modello base SDXL in /models/SDXL.ckpt, e l’output (il file LoRA) in /models/output\_lora/\*.safetensors. I parametri variano secondo esigenza (vedi guida LoRA [[8]](https://github.com/bmaltais/kohya_ss#:~:text=To%20train%20a%20LoRA%2C%20you,one%20GUI)).
* **Output atteso:** File \*.safetensors (LoRA) in kohya\_models/output\_lora. Durante l’addestramento, il log stamperà loss, immagini campione, etc. Al termine, verifica che il file LoRA sia presente e abbia dimensioni consistenti (p.es. 10–100 MB).
* **Errori comuni:** OOM CUDA: diminuisci il batch\_size o risoluzione (--resolution), abilita gradient checkpoint (--xformers). File corrotto: controlla spazio disco. Richiesta accesso alla GPU: docker run --gpus all. Permessi: usa sudo docker se necessario o aggiungi utente al gruppo docker.
* **Debug:** Se l’addestramento si blocca o fallisce, verifica che i percorsi siano corretti e che l’immagine Docker includa tutte le dipendenze (torch, diffusers, ecc.). Monitora nvidia-smi per uso VRAM. Controlla i log per errori Python (mancanze di moduli). Per basso uso GPU, prova ad avviare con flag --accelerator.

## 7. Test rapido del LoRA in ComfyUI con prompt suggeriti

* **Installazione ComfyUI:** Scarica l’ultima versione da [ComfyUI GitHub](https://github.com/comfyanonymous/ComfyUI) e avviala in WSL (o Windows) con Python 3.10. Ad esempio:
* git clone https://github.com/comfyanonymous/ComfyUI  
  cd ComfyUI  
  ./install.sh # in WSL  
  python main.py
* Si aprirà l’interfaccia web locale (di solito http://127.0.0.1:8188).
* **Carica modelli:** Copia il checkpoint SDXL (ad es. sdxl.ckpt) in ComfyUI/models/checkpoints. Copia il tuo LoRA .safetensors in ComfyUI/models/loras. ComfyUI rileva automaticamente i file in queste cartelle.
* **Workflow ComfyUI:** Apri ComfyUI e costruisci o carica un workflow. Usa un nodo **Load Checkpoint** per il modello base e un nodo **Load LoRA** per il tuo LoRA[[9]](https://docs.comfy.org/tutorials/basic/lora#:~:text=1.%20Ensure%20,to%20generate%20the%20image). Connetti poi un nodo di prompt (CLIPTextEncode) e un nodo di generazione (KSampler o altro). Imposta strength\_model su un valore medio (es. 0.5–0.7)[[10]](https://docs.comfy.org/tutorials/basic/lora#:~:text=Image%3A%20Load%20LoRA%20Node%20Models,be%20loaded%20using%20this%20node). Premi *Queue* per generare.  
  *Screenshot di un tipico workflow ComfyUI text-to-image.* Segui i passaggi dell’esempio: assicurati che il nodo Load Checkpoint carichi SDXL e Load LoRA il tuo file[[9]](https://docs.comfy.org/tutorials/basic/lora#:~:text=1.%20Ensure%20,to%20generate%20the%20image).
* **Prompt suggeriti:** Prova descrizioni come “ritratto di influencer futurista, illuminazione cinematica, alta definizione” o variazioni (aggiungi stile artistico, colore, clima ecc.). Cambia strength\_model per vedere l’effetto del LoRA.
* **Output atteso:** Un’immagine che combina il contenuto del prompt con lo stile appreso dal LoRA. Se il LoRA è buono, l’influencer generato rispecchierà lo stile target. Controlla che l’immagine abbia senso e qualità (volto riconoscibile).
* **Errori comuni:** Se ComfyUI non mostra il LoRA, verifica di averlo messo nella cartella models/loras[[10]](https://docs.comfy.org/tutorials/basic/lora#:~:text=Image%3A%20Load%20LoRA%20Node%20Models,be%20loaded%20using%20this%20node) e di aver aggiornato ComfyUI. Prompt troppo lunghi possono troncare output; ottimizza il testo. Se il workflow si blocca, ricontrolla i collegamenti dei nodi e la compatibilità versione.
* **Debug:** Usa il pannello log di ComfyUI (console) per errori Python. Se l’immagine esce nera o bianca, riduci i parametri (passaggi, steps). Se il LoRA non influenza l’immagine, aumenta strength\_model o controlla di aver applicato il nodo LoRA sul modello giusto.
* **Screenshot:** Il workflow sopra mostra come caricare checkpoint e LoRA. Puoi utilizzare modelli aggiuntivi (Stable Diffusion XL 1.0) e nodi di aggiustamento prompt per affinare i risultati.

## 8. Produzione contenuti (immagini e clip video brevi)

* **Generazione di immagini:** Usa ComfyUI come sopra per generare una serie di immagini varie. Per efficienza, usa un workflow batch che salva output in cartelle (nodo Save Image). Alternativa: altri UI come AUTOMATIC1111 o Stable Diffusion Web UI. Assicurati di salvare le immagini finali (rinominate es. img0001.png, img0002.png...).
* **Generazione video:** Con i modelli video open-source come **Wan2.1** (Alibaba, Apache-2.0) integrati in ComfyUI, puoi creare brevi clip. Il modello Wan2.1 1.3B richiede solo ~8 GB VRAM[[11]](https://docs.comfy.org/tutorials/video/wan/wan-video#:~:text=Wan2,It%20offers%20two%20versions). Installa i modelli da HuggingFace (UTM5 text encoder, VAE, CLIP, etc.) nelle cartelle ComfyUI/models/. Poi carica il workflow di Wan2.1 Text-to-Video (t2v) o Image-to-Video presente nella documentazione. Imposta il prompt di inizio e fine o un’immagine di input. Avvia l’esecuzione (vedi [86†L201-L210]). Otterrai un file video (ogg/mp4) o una sequenza di immagini frames.  
  *Esempio di workflow ComfyUI per image-to-image.* Per video Wan2.1, usi il nodo EmptyHunyuanLatentVideo con prompt iniziale/finale, poi Render Video. Scarica il video generato (solitamente in ComfyUI/outputs/).
* **Altre alternative:** Se non usi ComfyUI, considera librerie come imageio o script Python per interpolare immagini. Puoi anche usare modelli come Deforum SD o SmoothVideo (anche se non nativi in Docker). L’output è tipicamente un breve clip (es. 5–15s) con risoluzione 720p/1080p.
* **Output atteso:** Cartella images/ con PNG finali, file video clip.mp4 con artefatti minimi. Contenuti coerenti col prompt.
* **Errori comuni:** Se il video è troppo breve/monocromatico, rivedi i prompt di inizio/fine. VRAM insufficiente causa errore; riduci batch di frames. Il video potrebbe essere lento (numero frame basso); aumenta frames o fps.
* **Debug:** Controlla che i modelli video siano nella giusta cartella (models/diffusion\_models). Se ComfyUI non aggiorna, esegui l’ultima versione. Se usi Wan2.1, prova prima solo testo-to-text. Verifica eventuali messaggi di errore nei nodi (ad es. mancanza EmptyHunyuanLatentVideo).

## 9. Voiceover locale con TTS open-source (XTTS, Bark, Silero)

* **Prerequisiti:** Python 3.9+, librerie audio (scipy, etc.). GPU consigliata per velocità, ma CPU basterà. Modelli TTS come Silero e Bark sono gratuiti e locali.
* **Silero TTS:** Libreria Python semplice. Installazione e esempio:
* pip install silero-tts  
  python silero\_tts.py --language en --input-file script.txt --output-file out.wav
* Ad esempio, questo comando sintetizza il testo in script.txt in inglese e salva out.wav[[12]](https://github.com/daswer123/silero-tts-enhanced#:~:text=2,text%20file). Puoi anche passare direttamente --text "Hello world!". Silero offre vari speaker (consultare documentazione silero\_tts --list-speakers).
* **Bark (Suno.ai):** Modello TTS avanzato generativo. Installazione e Python:
* git clone https://github.com/suno-ai/bark.git  
  cd bark  
  pip install .   
  python -c "from bark import SAMPLE\_RATE, generate\_audio, preload\_models; preload\_models(); audio = generate\_audio('Ciao, sono il tuo influencer'); from scipy.io.wavfile import write; write('bark\_out.wav', SAMPLE\_RATE, audio)"
* Questo esempio carica i modelli, genera audio dal testo e salva bark\_out.wav[[13]](https://github.com/suno-ai/bark#:~:text=,models%20preload_models). È anche possibile usare il CLI: python -m bark --text "Testo da leggere" --output\_filename audio.wav[[14]](https://github.com/suno-ai/bark#:~:text=Command%20line). Bark supporta preset vocali multipli (es. "v2/en\_speaker\_6").
* **Output atteso:** File audio .wav con parlato chiaro. Controlla la qualità (intonazione, ritmo). Per Bark, verifica che i modelli si scarichino correttamente (richiede connessione internet alla prima esecuzione).
* **Errori comuni:** Se Silero fallisce, controlla che esista il modello per la lingua (--list-models). Bark può richiedere librerie aggiuntive (CUDA). L’installazione via pip va fatta da repo ufficiale (non pip install bark che è diversa).
* **Suggerimenti di debug:** Prova prima testi brevi. Verifica con --device cpu se GPU non disponibile (per Silero). Se la voce è monotona o glitch, passa a un altro speaker (nell’esempio Bark). Controlla il sample rate al salvataggio (48000 Hz vs 16000 Hz).

## 10. Editing automatico (FFmpeg, Blender scripting) per shorts

* **Montaggio video (FFmpeg):** Usa FFmpeg per assemblare immagini, aggiungere audio, sovrapposizioni e tagli. Ad esempio, per trasformare immagini in video:
* ffmpeg -framerate 1 -pattern\_type glob -i 'frames/\*.png' -c:v libx264 -r 30 -pix\_fmt yuv420p out.mp4
* Questa linea crea un video a 30 fps dove ogni immagine \*.png dura 1 sec[[15]](https://stackoverflow.com/questions/24961127/how-to-create-a-video-from-images-with-ffmpeg#:~:text=ffmpeg%20,pix_fmt%20yuv420p%20out.mp4). Per aggiungere la traccia audio generata:
* ffmpeg -framerate 1 -pattern\_type glob -i 'frames/\*.png' -i voice.wav -c:a copy -shortest -c:v libx264 -r 30 -pix\_fmt yuv420p final.mp4
* Questo fonde immagini e audio, tagliando alla fine del più corto[[16]](https://stackoverflow.com/questions/24961127/how-to-create-a-video-from-images-with-ffmpeg#:~:text=match%20at%20L208%20ffmpeg%20,pix_fmt%20yuv420p%20out.mp4).
* **Overlay e watermark (FFmpeg):** Puoi aggiungere logo/intro con l’overlay filter. Esempio:
* ffmpeg -i final.mp4 -i logo.png -filter\_complex "overlay=10:10" branded.mp4
* Qui logo.png è posizionato a (10,10) pixel dal bordo superiore sinistro[[17]](https://gist.github.com/bennylope/d5d6029fb63648582fed2367ae23cfd6#:~:text=Once%20you%20have%20ffmpeg%20installed%2C,an%20overlay%20filter%20like%20so). Puoi anche centrare o animare usando main\_w, overlay\_w come nell’esempio[[18]](https://gist.github.com/bennylope/d5d6029fb63648582fed2367ae23cfd6#:~:text=ffmpeg%20,test1.mp4).
* **Blender scripting:** Per intro/outro elaborate, usa Python in Blender: carica clip, aggiungi testo/effetti nelle strip del Video Sequence Editor. Un semplice script esempio:
* import bpy  
  bpy.ops.sequencer.image\_strip\_add(directory='/percorsi/img/', files=[{'name':'frame001.png'}], frame\_start=1)  
  # aggiungi audio  
  bpy.ops.sequencer.sound\_strip\_add(filepath='/percorsi/voice.wav', frame\_start=1)  
  # esporta video  
  bpy.context.scene.render.filepath = '/percorsi/output.mp4'  
  bpy.context.scene.render.ffmpeg.format = 'MPEG4'  
  bpy.ops.render.render(animation=True)
* Questo può essere eseguito in Blender (blender --background --python script.py).
* **Thumbnail e overlay di testo:** Usa uno strumento grafico (es. GIMP) o PIL in Python per creare miniatura (1280×720, testo/icone) a partire da un frame del video. Esempio FFmpeg per catturare frame:
* ffmpeg -i final.mp4 -ss 00:00:01.000 -vframes 1 thumbnail.jpg
* **Output atteso:** File video finale in mp4 di breve durata (15–60s) ottimizzato per social (16:9, 30fps). Versione con logo semitrasparente e voiceover sincronizzato. Thumbnail in .jpg.
* **Errori comuni:** Codec mancanti: installa FFmpeg completo. Se l’audio non si sente, controlla -c:a. Se i video risultano troppo grandi, riduci bit rate con -b:v 2M. Formati non supportati: preferisci mp4/H.264.
* **Debug:** Controlla i log FFmpeg per warning. Per audio, prova a separare i comandi (prima crea video muto, poi aggiungi audio). In caso di script Blender, verifica versioni compatibili e che si lancino in headless mode.

## 11. Branding: thumbnail, intro/outro, overlay

* **Thumbnail:** Crea un’immagine 1280×720 (o 1920×1080) con titolo e logo del canale. Usa software open-source (GIMP, Inkscape) o script Python (PIL). Ad esempio:
* from PIL import Image, ImageDraw, ImageFont  
  im = Image.new('RGB',(1280,720),(255,255,255))  
  draw = ImageDraw.Draw(im)  
  font = ImageFont.truetype('/usr/share/fonts/truetype/dejavu/DejaVuSans-Bold.ttf', 80)  
  draw.text((50,300),"Titolo Video",font=font,fill=(0,0,0))  
  im.save('thumbnail.jpg')
* **Intro/Outro:** Usa Blender o FFmpeg per un breve video introduttivo (3–5s) con logo animato. Puoi creare un file intro.mp4 statico o con dissolvenze, poi concatenarlo con il contenuto principale usando FFmpeg:
* ffmpeg -i "concat:intro.mp4|main.mp4|outro.mp4" -c copy final\_video.mp4
* Oppure con filtro concat.
* **Overlay (grafica fissa):** Se vuoi un logo o barra sul video, usa l’overlay FFmpeg come sopra[[17]](https://gist.github.com/bennylope/d5d6029fb63648582fed2367ae23cfd6#:~:text=Once%20you%20have%20ffmpeg%20installed%2C,an%20overlay%20filter%20like%20so). Assicurati il logo sia PNG trasparente. Puoi anche posizionarlo al centro o altrove (uso di main\_w/overlay\_w).
* **Output atteso:** Video con intro/logo coerenti con il brand. Thumbnail attraente. Controlla che gli overlay non coprano elementi importanti.
* **Errori comuni:** Testo tagliato o piccolissimo: verifica risoluzione e DPI. Script Blender bloccati: controlla versioni Python/Blender. Per overlay, se lo sfondo è animato, il logo potrebbe risultare mosso; mantieni statico o consisente.
* **Debug:** Apri il video finale e verifica su dispositivi mobili. Se qualcosa non appare, rivedi la timeline o i percorsi dei file. Verifica che il logo abbia canale alpha (transparenza).

## 12. Automazione pubblicazione su YouTube (n8n o Python)

* **n8n (open-source):** Installa n8n e crea un workflow di automazione. Il nodo **YouTube** di n8n consente di caricare video programmati. Ad esempio, puoi triggerare l’esecuzione quando un nuovo file appare in una cartella. La [documentazione n8n](https://n8n.io/integrations/youtube/) conferma che è possibile automatizzare upload su YouTube[[19]](https://n8n.io/integrations/youtube/#:~:text=apps%20and%20services,Just). Configura l’API YouTube con OAuth nel nodo (devi registrare app Google). Poi imposta parametri: titolo, descrizione, privacy, percorso video.
* **Python (YouTube API):** Usa la libreria Google API Python. Esempio basico da Google Developers:
* pip install google-api-python-client oauth2client  
  # Prepara client\_secrets.json con le credenziali OAuth2  
  python upload\_video.py --file="/path/final\_video.mp4" \  
   --title="Titolo Video" --description="Descrizione" \  
   --keywords="tag1,tag2" --category="22" --privacyStatus="public"
* Questo script esempio (scaricato dai samples Google) invia il video e i metadati[[20]](https://developers.google.com/youtube/v3/guides/uploading_a_video#:~:text=python%20upload_video.py%20,privacyStatus%3D%22private). Devi avere abilitato l’API YouTube Data sul tuo progetto Google Cloud e il file client\_secrets.json.
* **Output atteso:** Il video appare sul tuo canale YouTube con titolo/descrizione desiderati. n8n segnala successo, Python stampa l’ID video. Impostazione --privacyStatus definisce pubblico/privato.
* **Errori comuni:** OAuth fallito: controlla client\_secrets.json e autorizzazioni OAuth (Google Cloud Console). Limiti API superati: YouTube ha quote giornaliere. In n8n, se FFmpeg deve essere eseguito (per tagli), ricorda che n8n Cloud non lo supporta – serve n8n self-hosted.
* **Debug:** Leggi i log di n8n per dettagli. Con Python, guarda l’errore HTTP restituito. Spesso è un problema di scope OAuth (“https://www.googleapis.com/auth/youtube.upload”). Verifica le credenziali dell’account YouTube.

**Fonti:** Documentazione NVIDIA e Docker[[1]](https://gist.github.com/atinfinity/f9568aa9564371f573138712070f5bad#:~:text=docker%20run%20,smi); API Leonardo[[3]](https://docs.leonardo.ai/docs/generate-your-first-images#:~:text=curl%20,32b51c1f67d3)[[2]](https://leonardo.ai/pricing/#:~:text=Free%20tier%20users%3A%C2%A0Leonardo,only%20as%20enabled%20by%20a); esempi BlurDetection[[5]](https://www.geeksforgeeks.org/computer-vision/how-to-check-for-blurry-images-in-your-dataset-using-the-laplacian-method/#:~:text=,var) e Face Recognition[[6]](https://pypi.org/project/face-recognition/#:~:text=import%20face_recognition%20known_image%20%3D%20face_recognition.load_image_file%28,unknown.jpg); BLIP captioning[[7]](https://github.com/cobanov/image-captioning#:~:text=Example); kohya-ss LoRA[[8]](https://github.com/bmaltais/kohya_ss#:~:text=To%20train%20a%20LoRA%2C%20you,one%20GUI); ComfyUI LoRA[[9]](https://docs.comfy.org/tutorials/basic/lora#:~:text=1.%20Ensure%20,to%20generate%20the%20image)[[10]](https://docs.comfy.org/tutorials/basic/lora#:~:text=Image%3A%20Load%20LoRA%20Node%20Models,be%20loaded%20using%20this%20node); modelli video Wan2.1[[11]](https://docs.comfy.org/tutorials/video/wan/wan-video#:~:text=Wan2,It%20offers%20two%20versions); TTS Silero[[12]](https://github.com/daswer123/silero-tts-enhanced#:~:text=2,text%20file) e Bark[[13]](https://github.com/suno-ai/bark#:~:text=,models%20preload_models); FFmpeg slideshow[[15]](https://stackoverflow.com/questions/24961127/how-to-create-a-video-from-images-with-ffmpeg#:~:text=ffmpeg%20,pix_fmt%20yuv420p%20out.mp4)[[16]](https://stackoverflow.com/questions/24961127/how-to-create-a-video-from-images-with-ffmpeg#:~:text=match%20at%20L208%20ffmpeg%20,pix_fmt%20yuv420p%20out.mp4) e watermark[[17]](https://gist.github.com/bennylope/d5d6029fb63648582fed2367ae23cfd6#:~:text=Once%20you%20have%20ffmpeg%20installed%2C,an%20overlay%20filter%20like%20so); n8n e YouTube API[[19]](https://n8n.io/integrations/youtube/#:~:text=apps%20and%20services,Just)[[20]](https://developers.google.com/youtube/v3/guides/uploading_a_video#:~:text=python%20upload_video.py%20,privacyStatus%3D%22private). Tutti gli strumenti citati sono open-source o gratuiti.

[[1]](https://gist.github.com/atinfinity/f9568aa9564371f573138712070f5bad#:~:text=docker%20run%20,smi) Install NVIDIA Container Toolkit on WSL2 · GitHub

<https://gist.github.com/atinfinity/f9568aa9564371f573138712070f5bad>

[[2]](https://leonardo.ai/pricing/#:~:text=Free%20tier%20users%3A%C2%A0Leonardo,only%20as%20enabled%20by%20a) Leonardo.Ai Pricing: Individual, Team & API Plans | Leonardo.Ai

<https://leonardo.ai/pricing/>

[[3]](https://docs.leonardo.ai/docs/generate-your-first-images#:~:text=curl%20,32b51c1f67d3) Generate your First Images

<https://docs.leonardo.ai/docs/generate-your-first-images>

[[4]](https://docs.leonardo.ai/docs/generate-images-using-image-to-image-guidance#:~:text=Sample%20Request) Generate Images Using Image Guidance

<https://docs.leonardo.ai/docs/generate-images-using-image-to-image-guidance>

[[5]](https://www.geeksforgeeks.org/computer-vision/how-to-check-for-blurry-images-in-your-dataset-using-the-laplacian-method/#:~:text=,var) How to Check for Blurry Images in Your Dataset Using the Laplacian Method - GeeksforGeeks

<https://www.geeksforgeeks.org/computer-vision/how-to-check-for-blurry-images-in-your-dataset-using-the-laplacian-method/>

[[6]](https://pypi.org/project/face-recognition/#:~:text=import%20face_recognition%20known_image%20%3D%20face_recognition.load_image_file%28,unknown.jpg) face-recognition · PyPI

<https://pypi.org/project/face-recognition/>

[[7]](https://github.com/cobanov/image-captioning#:~:text=Example) GitHub - cobanov/image-captioning: Image captioning using python and BLIP

<https://github.com/cobanov/image-captioning>

[[8]](https://github.com/bmaltais/kohya_ss#:~:text=To%20train%20a%20LoRA%2C%20you,one%20GUI) GitHub - bmaltais/kohya\_ss

<https://github.com/bmaltais/kohya_ss>

[[9]](https://docs.comfy.org/tutorials/basic/lora#:~:text=1.%20Ensure%20,to%20generate%20the%20image) [[10]](https://docs.comfy.org/tutorials/basic/lora#:~:text=Image%3A%20Load%20LoRA%20Node%20Models,be%20loaded%20using%20this%20node) ComfyUI LoRA Example - ComfyUI

<https://docs.comfy.org/tutorials/basic/lora>

[[11]](https://docs.comfy.org/tutorials/video/wan/wan-video#:~:text=Wan2,It%20offers%20two%20versions) ComfyUI Wan2.1 Video Examples - ComfyUI

<https://docs.comfy.org/tutorials/video/wan/wan-video>

[[12]](https://github.com/daswer123/silero-tts-enhanced#:~:text=2,text%20file) GitHub - daswer123/silero-tts-enhanced: Silero TTS Enhanced is a Python library that enhances the original Silero TTS project, providing a convenient way to synthesize speech from text using Silero TTS models. It offers a user-friendly interface for both standalone script usage and integration into Python projects, along with additional features

<https://github.com/daswer123/silero-tts-enhanced>

[[13]](https://github.com/suno-ai/bark#:~:text=,models%20preload_models) [[14]](https://github.com/suno-ai/bark#:~:text=Command%20line) GitHub - suno-ai/bark: Text-Prompted Generative Audio Model

<https://github.com/suno-ai/bark>

[[15]](https://stackoverflow.com/questions/24961127/how-to-create-a-video-from-images-with-ffmpeg#:~:text=ffmpeg%20,pix_fmt%20yuv420p%20out.mp4) [[16]](https://stackoverflow.com/questions/24961127/how-to-create-a-video-from-images-with-ffmpeg#:~:text=match%20at%20L208%20ffmpeg%20,pix_fmt%20yuv420p%20out.mp4) How to create a video from images with FFmpeg? - Stack Overflow

<https://stackoverflow.com/questions/24961127/how-to-create-a-video-from-images-with-ffmpeg>

[[17]](https://gist.github.com/bennylope/d5d6029fb63648582fed2367ae23cfd6#:~:text=Once%20you%20have%20ffmpeg%20installed%2C,an%20overlay%20filter%20like%20so) [[18]](https://gist.github.com/bennylope/d5d6029fb63648582fed2367ae23cfd6#:~:text=ffmpeg%20,test1.mp4) FFmpeg add a watermark to video · GitHub

<https://gist.github.com/bennylope/d5d6029fb63648582fed2367ae23cfd6>

[[19]](https://n8n.io/integrations/youtube/#:~:text=apps%20and%20services,Just) YouTube integrations | Workflow automation with n8n

<https://n8n.io/integrations/youtube/>

[[20]](https://developers.google.com/youtube/v3/guides/uploading_a_video#:~:text=python%20upload_video.py%20,privacyStatus%3D%22private) Upload a Video  |  YouTube Data API  |  Google for Developers

<https://developers.google.com/youtube/v3/guides/uploading_a_video>