

DOI: <https://doi.org/10.15276/ict.03.2026.93>

UDC 004.032.26:004.932

Development of diffusion models for video sequence generation

Andrey V. Lytvynenko¹⁾

ORCID: <https://orcid.org/0000-0002-4611-7832>; akoshka1@gmail.com

Mariia O. Voronenko¹⁾

ORCID: <https://orcid.org/0000-0002-5392-5125>; mary.voronenko@gmail.com

¹⁾ Odesa Polytechnic National University, 1, Shevchenko Avenue. Odesa, 65044, Ukraine

ABSTRACT

This paper presents a software implementation for generating short video sequences on a central processing unit. A unified interface is developed for training a compact pixel-space diffusion model and applying pretrained Stable Video Diffusion and CogVideoX models. The educational model uses three-dimensional convolutions, diffusion-timestep conditioning, and noise-prediction error minimization. Stochastic denoising diffusion probabilistic model and deterministic denoising diffusion implicit model sampling are implemented. The mathematical formulation, network architecture, execution parameters, export procedures, and diagnostic evaluation are described. The architecture of the software pipeline for generating video sequences on a CPU is presented. The system includes three alternative branches: training and application of a compact diffusion model, image-to-video generation, and text-to-video generation. All branches share export, evaluation, and computational resource monitoring procedures. The dependence of the parameter count on network width is derived analytically, and the number of denoiser calls is compared across sampling schemes. A protocol is proposed for assessing the trade-off between computational cost and video properties. Results are reported for a smoke test of the compact model comprising two training iterations and the generation of four 8×8-pixel frames. The test confirms that the pipeline executes successfully but does not demonstrate the learning of coherent dynamics or an advantage over other models. This work occupies an engineering and methodological niche: it combines training a small pixel-space Denoising Diffusion Probabilistic Model with pretrained Stable Video Diffusion and CogVideoX models while saving outputs and resource reports. Its contribution is a transparent implementation and a reproducible research protocol; superiority over large models requires separate experimental verification.

Keywords: diffusion model; video generation, video sequences; three-dimensional convolution; latent diffusion; CogVideoX; temporal consistency; software pipeline

Для цитування: Lytvynenko A. V., Voronenko M. O. “Development of diffusion models for video sequence generation”. *Informatics. Culture. Engineering*. 2026; Vol 3 № 2 (4): 1191–1203. DOI: <https://doi.org/10.15276/ict.03.2026.93>

1. Introduction. Video generation requires reproducing both the visual structure of individual frames and changes in the scene over time. Independent frame synthesis can cause flickering, changes in object appearance, and motion discontinuities. Diffusion generates a sample by progressively removing noise; processing a video tensor allows the model to account for relationships between neighboring frames [1], [5].

The main practical difficulty is the computational cost of repeatedly applying a neural network and the increase in memory requirements with video duration and resolution. Compact CPU implementation is useful for educational and local research tasks because it enables investigation of the diffusion mechanism and recording of measurable execution characteristics.

The objective is to develop and formalize a software pipeline for training and applying diffusion models for video sequence generation on a CPU. We consider three tasks: unconditional synthesis using synthetic data, image-to-video generation, and text-to-video generation. The contribution is engineering and methodological: a compact video denoiser, unified output reports, and a protocol for separately evaluating quality and resource usage. No new class of diffusion models or superiority over existing systems is claimed.

The main contributions of the paper are as follows:

- development of a compact video diffusion model for Central Processing Unit (CPU) execution. A three-dimensional convolutional denoiser, conditioned on the diffusion timestep, jointly processes a video clip's spatial and temporal dimensions. Noise-prediction training and generation using Denoising Diffusion Probabilistic Model (DDPM) and Denoising Diffusion Implicit Model (DDIM) are supported;

- creation of a unified software pipeline for three video generation modes. The pipeline combines unconditional synthesis using the compact model, image-to-video generation using Stable Video Diffusion XT, and text-to-video generation using CogVideoX-2b. We implement common procedures for video export and saving execution parameters;
- development of diagnostic and resource evaluation procedures. We calculate frame-level and temporal indicators and record execution time and memory consumption. Separate evaluation of original frames and decoded MPEG-4 Part 14 (a multimedia container for storing video and audio) (MP4) video accounts for the effects of video compression on measurements;
- experimental verification of functionality and identification of implementation limitations. A smoke test of a model with 16,867 parameters confirmed training, DDIM generation, export, and report creation. Analysis of the resulting clip (Contrastive Language-Image Pre-training) showed that two training iterations do not establish temporal consistency and motivated full training and independent validation.

2. Related work. In DDPM, generation is formulated as a learned reversal of a Gaussian noise addition process [1]. In this implementation, noise is predicted for the entire video tensor. Improved DDPM investigates changes to the noise schedule, variance learning, and other ways to improve diffusion synthesis [3]. Of these techniques, the educational branch supports linear or cosine schedules; it does not implement variance learning.

DDIM allows a shorter sampling sequence without changing the basic denoiser training procedure [2]. Diffusion Probabilistic Model Solver (DPM-Solver) offers another approach: it treats sampling as solving a diffusion ordinary differential equation using a specialized high-order method [18]. These studies show that generation cost depends on both the network and the numerical method. The software implements DDPM and deterministic DDIM; DPM-Solver is a possible future extension. Reductions in network calls must be assessed together with quality and actual CPU execution time.

Video Diffusion Models extend diffusion to video through spatiotemporal feature processing [5]. This processing captures inter-frame relationships within a single generation process. The compact Three-dimensional Convolution (Conv3d) network in this work also processes the video tensor jointly, but its local convolutions and shallow depth substantially limit the modelling of complex scenes and long-range dependencies.

Imagen Video uses a cascade of a base video model and spatial and temporal super-resolution modules, together with progressive distillation [9]. Make-A-Video combines learning from text–image pairs with motion learning from videos without text annotations, supplementing generation with temporal and spatial modules [10]. The absence of text–video pairs in the latter approach does not mean the absence of video data. These systems demonstrate the role of dedicated motion and scaling mechanisms; the educational DDPM has neither cascades nor knowledge transfer from a text-to-image model.

VideoFusion changes the noise itself: frame noise is decomposed into a base component shared across the clip and a residual component that varies over time, with two jointly trained denoisers corresponding to these components [21]. This explicitly accounts for content redundancy between frames. In the educational DDPM, noise elements are independent, and the network learns temporal dependencies alone. Comparing these schemes requires a separate ablation study with identical data and computational budgets.

Latent Diffusion Models move iterative denoising into an autoencoder space, reducing the dimensionality of the processed data [4]. Align your Latents extends this principle to video: a pretrained image model is augmented with temporal layers, and super-resolution is coordinated across frames [11]. Computational savings in latent space should be evaluated together with encoding and decoding costs.

AnimateDiff introduces a trainable motion module that can be attached to compatible personalized image models [12]. Stable Video Diffusion investigates scaling latent video diffusion and preparing training data [6]. VideoCrafter2 studies how high-quality images can overcome limitations in video data quality [13]. Together, these works emphasize visual knowledge transfer,

motion learning, and corpus quality. The pipeline uses pretrained Stable Video Diffusion XT (SVD XT) without reproducing its training; the educational branch is trained on synthetic squares and does not establish transfer to real scenes.

VideoCrafter1 provides separate text-to-video and image-to-video models [22], while VideoCrafter2 continues investigating training data quality [13]. LaVie combines a base latent generator, temporal interpolation, and super-resolution using joint training on images and videos [23]. Show-1 builds a hybrid cascade: a pixel-space model generates low-resolution video, followed by a latent model for super-resolution [24]. These approaches show that pixel or latent representations may differ between stages of one system. In the developed pipeline, the DDPM and pretrained latent-model branches run separately and do not form such a cascade.

Diffusion Transformers investigates transformer processing of latent patches and diffusion architecture scaling [7]. CogVideoX uses a specialized transformer for text-to-video generation [8]; the developed system supports CogVideoX-2b inference. This approach differs from the compact convolutional denoiser in data representation, scale, and conditioning.

Open-Sora 2.0 treats training efficiency as the result of jointly organizing data, architecture, training strategy, and system optimizations [14]. Wan describes large-scale video generation using a diffusion transformer, a video autoencoder, and data preparation [15]. These technical reports broaden the scaling context, but their resource characteristics do not directly transfer to a CPU implementation. Running pretrained weights and fully training them are different computational tasks.

Latte extracts spatiotemporal tokens and compares transformer processing variants that factorize spatial and temporal dimensions [25]. HunyuanVideo treats generation as a combination of data, architecture, progressive scaling, and training infrastructure [26]. These studies motivate separate analyses of denoiser complexity, clip length, and resolution. Adding attention to the compact network will require measuring memory and execution time rather than merely counting parameters.

Tune-A-Video adapts a pretrained image model using a single text–video pair, employing spatiotemporal attention and DDIM inversion to guide sampling structurally [19]. Text2Video-Zero uses an image model without additional video generation training, introducing motion into latent codes and cross-frame attention to the first frame [20]. Training savings therefore come from limited adaptation or changes to the generation procedure. Both approaches rely on a previously trained image model and differ from training a compact DDPM from scratch.

Dreamix investigates text-guided editing of an input video's appearance and motion after adapting the model to that video [27]. VideoComposer combines textual, spatial, and temporal conditions, with motion vectors providing an explicit control signal [28]. These works identify controllability as a distinct characteristic of generation. The current system supports text and image conditions in pretrained models but does not support input-clip editing, specified trajectories, or a unified encoder for multiple conditions.

Fréchet Video Distance (FVD) compares real and generated video distributions in a video-network feature space [16]. This approach accounts for clip properties and does not require frame-by-frame correspondence for each pair, but it depends on the samples and feature extraction procedure. VBench divides evaluation into 16 aspects, including identity preservation, motion smoothness, and flickering [17]. Thus, a single inter-frame similarity indicator cannot capture overall video-generation quality.

The current pipeline computes accessible diagnostic characteristics of frames and inter-frame changes; FVD and VBench are not implemented. Low inter-frame error may correspond to a static video, while apparent detail may result from noise. The review motivates these indicators with motion checks, condition adherence, and perceptual evaluation.

The reviewed studies cover probabilistic formulation, accelerated sampling, temporal architecture, knowledge transfer, and evaluation. This work occupies an engineering and methodological niche: it combines training a small pixel-space DDPM with pretrained SVD and CogVideoX models while saving outputs and resource reports. Its contribution is a transparent

implementation and a reproducible research protocol; superiority over large models requires separate experimental verification.

3. Problem Formulation. Let $X=(I_1,\dots,I_F)$ denote a video clip containing F RGB frames, each of size $H\times W$. The clip is represented as a tensor $X\in\mathbb{R}^{(3\times F\times H\times W)}$; a batch has shape $[B,3,F,H,W]$. The index f denotes a frame, whereas t denotes a diffusion timestep. These temporal indices are distinct.

The task is to construct a distribution $p_\theta(X|c)$, where c is absent for the educational DDPM, is an image for SVD, or is text for CogVideoX. Evaluation criteria include frame properties, temporal dynamics, and computational cost. Quality metrics and resource indicators are considered separately.

4. Mathematical Diffusion Model. Define timesteps $t=1,\dots,T$, variances $\beta_t\in(0,1)$, coefficients $\alpha_t=1-\beta_t$, and $\bar{\alpha}_t=\prod_{i=1}^t\alpha_i$. The forward process for a clean video tensor x_0 is defined by Gaussian transitions [1]:

$$q(x_t|x_{t-1})=N\left(\sqrt{\alpha_t}x_{t-1},\beta_tI\right). \quad (1)$$

A noisy sample can be obtained directly without computing all preceding transitions:

$$x_t=\sqrt{\bar{\alpha}_t}x_0+\sqrt{1-\bar{\alpha}_t}\varepsilon, \quad \varepsilon\sim N(0,I). \quad (2)$$

The identity matrix I acts in the space of all video-tensor elements. One timestep t is selected for each clip in the batch, and noise values are generated for all its pixels. The code uses indexing $0,\dots,T-1$, equivalent to the notation adopted here after shifting the index.

The denoiser ε_θ estimates noise from x_t and t . Training minimizes the mean error over tensor elements:

$$L(\theta)=E_{x_0,t,\varepsilon}\left[\frac{\|\varepsilon-\varepsilon_\theta(x_t,t)\|_2^2}{d}\right], \quad s=3FWH. \quad (3)$$

The expectation in (3) is taken over clean clips, a random timestep, and Gaussian noise. At each iteration, it is estimated using the current batch. Low training MSE alone does not establish realism, diversity, or correct generated dynamics.

The cosine schedule is constructed using a normalized function $g(u)$, where $u\in[0,1]$ and $s=0.008$ [3]:

$$g(u)=\frac{\cos^2\left[\pi(u+s)/(2(1+s))\right]}{\cos^2\left[\pi s/(2(1+s))\right]}. \quad (4)$$

The software computes $\beta_t=\text{clip}(1-g(t/T)/g((t-1)/T),10^{-5},0.999)$ and then recomputes $\bar{\alpha}_t$ as a product of coefficients. After clipping β_t , the product may differ from the original analytical curve g . The linear option uses β values from 10^{-4} to 0.02.

At a reverse step, the noise estimate is used to form a clipped estimate of the clean clip:

$$\hat{x}_0=\text{clip}\left[\frac{x_t-\sqrt{1-\bar{\alpha}_t}\varepsilon_\theta(x_t,t)}{\sqrt{\bar{\alpha}_t}},-1,1\right]. \quad (5)$$

For DDIM with $\eta=0$ and the next selected timestep $j<t$, a corrected noise direction is used [2]:

$$g_t = \frac{x_t - \sqrt{\alpha_t} \hat{x}_0}{\sqrt{1 - \alpha_t}}. \quad (6)$$

$$x_j = \sqrt{\alpha_j} \hat{x}_0 + \sqrt{1 - \alpha_j} g_t. \quad (7)$$

After the final step, the result equals $\hat{x}_0$. The correction in (6) accounts for clipping in (5). DDPM uses a transition with a posterior mean and variance $\tilde{\beta}_t = \beta_t (1 - \alpha_{t-1}) / (1 - \alpha_t)$; no additional noise is added at the final transition. These equations describe the educational branch specifically, rather than the internal schedules of pretrained SVD and CogVideoX.

5. Architecture and Software Implementation. The educational denoiser contains an input Conv3d layer $3 \rightarrow c$, a residual block with two Conv3d layers $c \rightarrow c$, and an output Conv3d layer $c \rightarrow 3$. All kernels are $3 \times 3 \times 3$, with stride 1 and padding=1. Group Normalization uses four groups; the width c must be divisible by four. Sigmoid Linear Unit is the nonlinearity. This is a compact network without a multilevel U-Net or attention mechanism.

The timestep t is normalized by T and fed into an MLP with dimensions $1 \rightarrow c \rightarrow c$. The resulting vector is added to the input convolution features after broadcasting over the spatial and frame axes. The residual connection preserves input-feature information. Temporal convolutions link neighboring frames but impose no explicit constraint on the physical correctness of motion.

The synthetic generator creates one colored square against a background. Initial positions and velocities are random; velocity components are selected from $\{-2, -1, 1, 2\}$. The trajectory includes reflections at the boundaries. Color remains constant within each clip, and frame values are normalized to $[-1, 1]$. The square side length is $\max(2, \lfloor H/6 \rfloor)$ for the square frames used. Data are regenerated at every iteration, so the iteration count should not be interpreted as an epoch count over a fixed dataset.

Table 1. Default configurations

Characteristic	Educational DDPM	SVD XT	CogVideoX-2b
Condition	None	Image	Text
Training in the code	Yes	No.	No.
Frames and dimensions	8; 24×24	25; 1024×576	17; 720×480
Generation steps	DDIM 20 / DDPM 100	15	20
Export FPS	8	7	8
Representation	Pixels	Latents	Latents

Source: compiled by the authors

By default, DDPM training runs for 2,000 iterations with batch=4, $c=16$, $T=100$, and a learning rate of 2×10^{-4} . AdamW is used, with the gradient norm clipped to 1. The software saves execution parameters, weights, and loss history in a checkpoint and reports. When loading an older checkpoint without width or schedule information, the software uses $c=32$ and a linear schedule.

Pretrained models run on the CPU in float32. The SVD input image is converted to RGB and resized to fit, with cropping if necessary. CogVideoX uses guidance=6 and checks that the frame count is $4k+1$ and no greater than 49. The – revision and – local-only options fix the weight revision and enable local files. Exact library versions and weights should be retained to reproduce the experiment.

6. Video Data Processing Pipeline. The pipeline combines configuration, separate generation branches, export, and diagnostic evaluation. DDPM training is linked to sampling through a saved checkpoint; the two commands run separately. Application of SVD and CogVideoX does not depend on training the compact denoiser.

Fig. 1 presents the architecture of the software pipeline for generating video sequences on a CPU. The system includes three alternative branches: training and application of a compact diffusion model, image-to-video generation, and text-to-video generation. All branches share export, evaluation, and computational resource monitoring procedures.

Execution configuration. The top block specifies the command and execution parameters: PyTorch on the CPU, random seed=42, and four computational threads. The ResourceMonitor module records execution time and sampled peak resident set size (RSS). These data allow resource costs to be assessed for the selected mode.

Branch A – compact DDPM training and generation. Training data are generated synthetically as RGB video clips containing moving squares. The figure shows default settings: eight 24×24-pixel frames and a batch of four clips. Gaussian noise is added to the original video tensor:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \varepsilon, \quad \varepsilon \sim N(0, I).$$

Here, x_0 is the original clip, x_t is its noisy version, and $\bar{\alpha}_t$ determines the fraction of retained signal at diffusion timestep t . The three-dimensional convolutional network receives a video tensor of shape $[B, 3, F, H, W]$ and the timestep, then predicts the added noise. B , F , H , and W denote batch size, frame count, height, and width; 3 denotes the number of color channels. AdamW updates the network parameters by minimizing the mean squared noise-prediction error. Model weights, loss history, and a training report are saved.

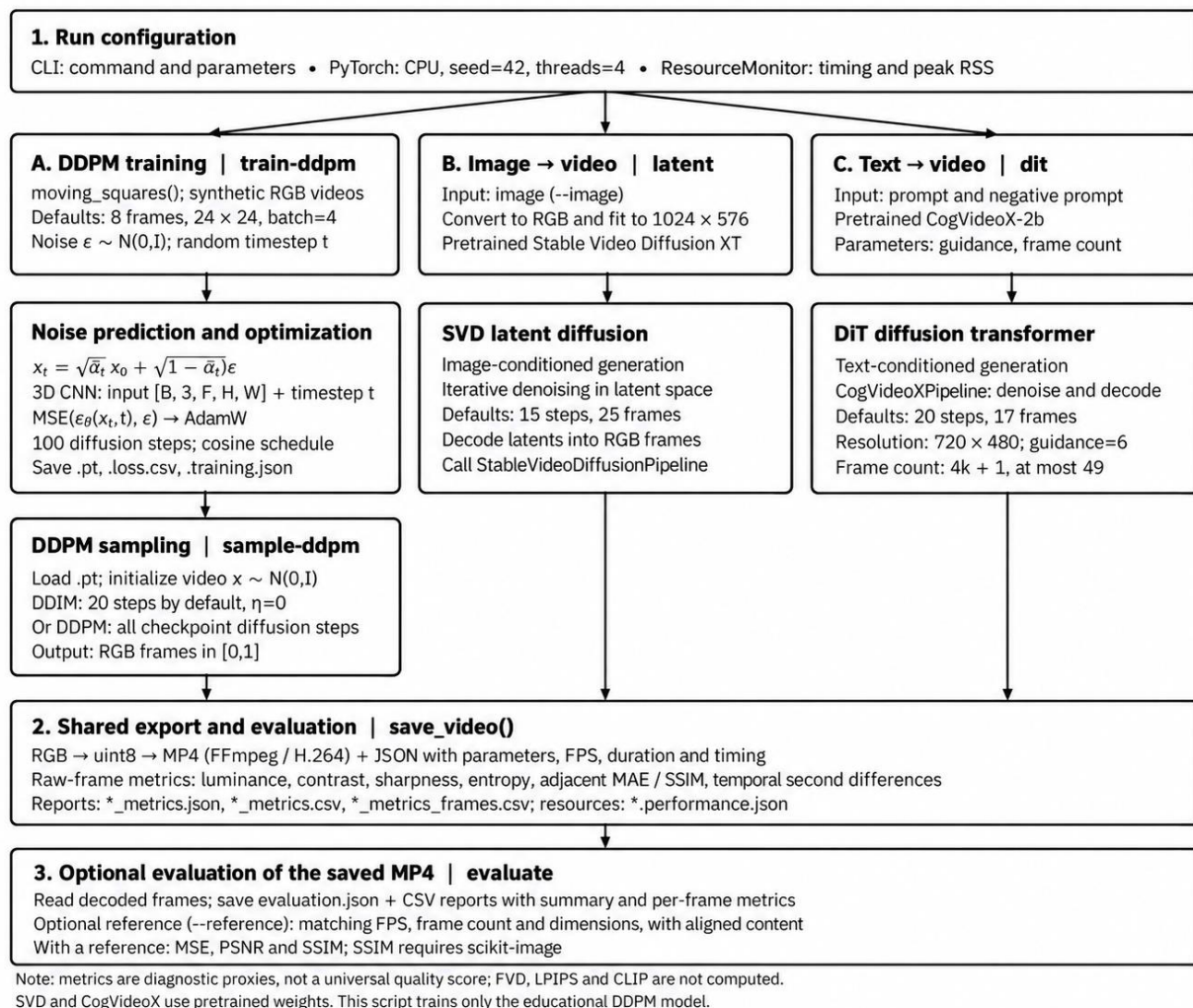


Fig. 1. Architecture of the CPU video generation software pipeline

Source: compiled by the authors

Generation runs as a separate command after the saved weights are loaded. An initial random video tensor is progressively denoised. Available sampling options are DDIM with 20 steps by

default and $\eta=0$, or DDPM using all diffusion timesteps specified in the checkpoint. The output comprises RGB frames with values in $[0,1]$. Branch B – image-to-video generation. The input image is converted to RGB and fitted to 1024×576 pixels. Pretrained Stable Video Diffusion XT uses the image as a condition for iterative denoising in latent space. The resulting latents are decoded into RGB frames. The diagram specifies 15 generation steps and 25 output frames. This branch applies a pretrained model without training it within the pipeline.

Branch C – text-to-video generation. CogVideoX-2b receives a text prompt and, optionally, a negative prompt. The diffusion transformer produces a video representation that is then decoded into frames. Default parameters are 20 steps, 17 frames, a resolution of 720×480 pixels, and a guidance scale of 6. The frame count must satisfy $F=4k+1$ and must not exceed 49.

Common export and initial evaluation. Outputs from each branch are passed to `save_video()`. Frames are converted to uint8 and saved as MP4 using the H.264 codec. The code also creates JSON and CSV reports with execution parameters, duration, frame rate, execution time, and diagnostic indicators. The code computes brightness, contrast, sharpness, entropy, inter-frame MAE and SSIM, and second temporal difference indicators for the original frames before video compression.

Additional evaluation of the saved video. The bottom block reads frames from the MP4 again, enabling analysis after compression and decoding. MSE, PSNR, and SSIM can be computed when a semantically aligned reference video is available; comparison requires matching frame rates, frame counts, dimensions, and temporal correspondence.

Common export saves an H.264 MP4 and a JSON file with parameters. Metrics for original frames before compression are stored separately from the results of the `evaluate` command, which reads the decoded MP4. This separation accounts for how video compression affects numerical indicators.

Overall interpretation of the diagram. The figure shows a unified interface for three video generation approaches with common output files and diagnostic procedures. Only the compact DDPM is trained; SVD and CogVideoX use pretrained weights. The reported metrics help identify frame changes and artefacts but are not a universal quality assessment: static video can have low inter-frame error, and noise can have high sharpness and entropy. FVD, LPIPS (Learned Perceptual Image Patch Similarity), and CLIP are not computed in this pipeline.

7. Quality and Resource Evaluation Methodology. For frames $Y_f \in [0,1]^{3 \times H \times W}$ We calculate mean luminance, luminance standard deviation, Laplacian variance, and luminance-distribution entropy. Luminance is a weighted RGB sum with coefficients 0.2126, 0.7152, and 0.0722. Entropy uses 256 bins. High sharpness or entropy may result from noise, so we treat these quantities as diagnostic features.

For $d_f = Y_f - Y_{f-1}$, $f = 2, K, F$, the mean inter-frame change is defined as:

$$A = \frac{1}{F-1} \sum_{f=2}^F \frac{\|d_f\|_1}{3HW} \quad (8)$$

Changes in inter-frame dynamics are assessed using the second difference:

$$J = \frac{1}{F-2} \sum_{f=3}^F \frac{\|d_f - d_{f-1}\|_1}{3HW} \quad (9)$$

The indicator A requires $F \geq 2$, and J requires $F \geq 3$. Small values may occur with smooth motion or no motion at all. Large values may indicate artefacts or an abrupt real change in the scene. Therefore, neither A nor J should be minimized without checking content. The script also saves inter-frame SSIM, temporal pixel variance, and the fraction of transitions with $MAE \leq 1/255$.

When an aligned reference R is available, the error over all frames and pixels is used:

$$MSE_{ref} = \frac{\|Y - R\|_2^2}{3FWH}, \quad PSNR = -10 \log(MSE_{ref}) \quad (10)$$

The PSNR formula in (10) assumes a maximum signal value of 1. When MSE is zero, the result is infinite and is saved as the string Infinity. We check frame count and dimensions, and FPS when reading MP4; we also perform semantic alignment with the reference beforehand. For unconditional synthesis, comparison with an arbitrary clip is not a valid quality evaluation.

Resource evaluation includes loading, inference, export, and metric computation times, together with total process duration and sampled peak RSS. Export FPS describes video playback. Different quantities are used to assess computational performance:

$$v_{gen} = \frac{F}{\tau_{gen}}, \quad RTF = \frac{\tau_{gen}}{F / FPS} \quad (11)$$

Here, τ_{gen} duration. In the current DDPM code, generation_seconds includes checkpoint loading and preparation, whereas inference_seconds for pretrained models measures a separate inference stage. Measurement boundaries must be aligned before direct comparison. psutil and scikit-image are optional dependencies; missing data should be explicitly recorded.

8. Results. Results were obtained from smoke.training.json, smoke.loss.csv, smoke.json, smoke.performance.json, smoke_metrics.json, and two CSV metric reports. Checkpoint metadata and smoke.mp4 decoding were also checked. One CPU smoke test of the educational branch was examined; no SVD or CogVideoX results are included in the supplied files. The test configuration differs from the defaults in Table 1.

Table 2. Training configuration and results

Indicator	Value
Network width; trainable parameters	c=16; 16,867
Frames and resolution	4; 8×8 pixels
Schedule; sampling	cosine; DDIM, 10 denoiser calls
Seed; generation threads	42; 4
Training iterations	2
Training time; iterations per second	1.192528 s; 1.677110
Noise MSE at iterations 1 and 2	1.047072; 1.163834
Mean MSE over two iterations	1.105453

Source: compiled by the authors

The parameter count corresponds to $P(16)=55 \cdot 16^2 + 174 \cdot 16 + 3 = 16,867$. MSE increased by 0.116763, or 11.15 % relative to the first iteration. Two values do not establish a convergence trend because the training batch, diffusion timestep, and noise change. The final_100_mean_noise_mse field actually averages the two available iterations rather than 100 observations.

The sum of generation, export, and metric computation times is 0.403480 s; the difference from total time is 0.028324 s. Metric computation accounts for approximately 60.09 % of total time, export for 29.50 %, and generation for 3.85 %. In the educational branch, generation_seconds includes checkpoint loading and preparation and is not the denoiser-only time. CPU time may exceed wall-clock time under multithreaded execution.

Table 3. Measured generation resources

Indicator	Value
Generation; export; metric computation	0.016628; 0.127372; 0.259480 s
Total process time	0.431804 s
Process CPU time	1.000000 s
Sampled peak RSS; polling interval	216.484375 MB; 0.1 s
Playback FPS; duration	8 frames/s; 0.5 s
F / generation_seconds; RTF	240.559541 frames/s; 0.033256

Source: compiled by the authors

The reports describe original frames before MP4 compression. We computed mean inter-frame indicators over three transitions and second-difference indicators over two frame triplets. These are descriptive characteristics of one clip, not estimates of the generator's quality distribution.

Table 4. Diagnostic clip characteristics

Indicator	Value
Mean luminance; contrast	0.482328; 0.364846
Laplacian variance	2.382919
Luminance entropy	3.365255 bits
Mean inter-frame MAE	0.505269
Mean second-difference MAE	1.030731
Mean inter-frame SSIM	−0.046246
Mean temporal pixel variance	0.179113
Fraction of nearly identical transitions	0.000000 at MAE threshold 1/255

*Source: compiled by the authors***Table 5. Indicators for individual transitions**

Transition	MAE	SSIM	Second difference
0→1	0.440158	0.115037	Undefined
1→2	0.560842	−0.154808	0.997549
2→3	0.514805	−0.098965	1.063914

Source: compiled by the authors

A mean MAE of approximately 0.505 for RGB values in [0,1] indicates large changes between neighboring frames. The mean SSIM is negative, and two of the three transitions are negative. The second difference indicates substantial changes in inter-frame increments. These characteristics are consistent with weak temporal consistency but do not measure the physical correctness of motion.

Visual inspection reveals changing color structures without a clearly persistent object or coherent trajectory. We obtained the image from the decoded MP4; values in Table 4 and Table 5 come from original-frame reports. The MP4 contains H.264 video with four 8×8 frames, a frame rate of 8 frames/s, and a duration of 0.5 s; its file size is 2,030 bytes. Decoding completed successfully.

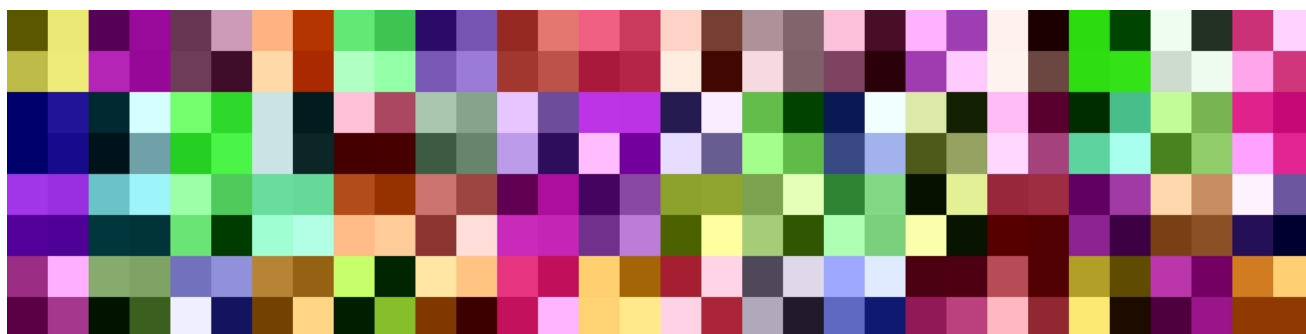


Fig. 2. Four decoded frames from smoke.mp4, shown from left to right and enlarged for viewing

Source: compiled by the authors

9. Discussion. The main smoke-test result confirms the sequence of training, checkpoint saving, DDIM sampling, export, and diagnostic report creation. Two parameter updates are insufficient to conclude that the synthetic video-data distribution was learned. The increase in training MSE does not prove generator degradation; it may reflect differences between two random training examples. However, the supplied data also provide no evidence of established convergence.

Inter-frame MAE and SSIM, together with the appearance of the decoded frames, indicate weak temporal consistency in this clip. High Laplacian variance and entropy should not be interpreted as great detail: irregular color changes also increase these indicators. The absence of nearly identical transitions confirms frame changes but not meaningful motion. The second-difference indicator is not bounded by 1, even for normalized frames, so 1.030731 is not a range error.

The values of 240.56 frames/s and $RTF=0.03326$ refer to `generation_seconds` and the extremely small 8×8 resolution. They do not reflect realistic video-generation speed. For the entire measured process, the time-to-video-duration ratio is $0.431804/0.5=0.863608$; this was also obtained in a single run. The dominance of export and evaluation shows that overhead operations exceed sampling costs for such a small tensor. RSS sampling at 0.1-s intervals may miss brief peaks. Missing processor details and environment versions limit reproducibility of the resource results.

One seed, one clip, three transitions, and two training iterations do not support justified quality confidence intervals, statistical significance tests, or comparison of DDIM with DDPM. The supplied files contain no independent validation, reference trajectories, FVD, perceptual evaluation, or expert assessment. This conclusion is consistent with the need for multidimensional video evaluation discussed in [16], [17]. SVD and CogVideoX results require separate experimental series.

The next stage should include full training, error recording on an independent dataset, and repeated generation using a fixed set of seeds. Comparisons of step counts should use one checkpoint and identical conditions; architectural comparisons should use aligned training budgets. For synthetic squares, preservation of object color and shape, velocity, and boundary reflections should be checked separately. Store loading, denoising, export, and evaluation measurements separately.

10. Conclusions. The smoke test of a compact video diffusion model with 16,867 parameters completed two training iterations in 1.192528 s and produced a clip of four 8×8 -pixel frames using DDIM with ten denoiser calls. The exported MP4 decodes correctly; the total measured generation-process time was 0.431804 s, and sampled peak RSS was 216.48 MB.

The mean training MSE was 1.105453; the two observations showed no error reduction. The mean inter-frame MAE (0.505269) and SSIM (- 0.046246), along with visual inspection, do not confirm coherent dynamics. The results demonstrate the tested pipeline's functionality but not its

quality after full training, superiority over other models, or efficiency at high resolution. Further conclusions require training to verifiable convergence and an independent series of experiments.

Conflict of interests: Authors declare that they do not have conflicts of interest, in particular financial, personal, copyright or any of a different nature, which could have influenced the research, as well as the results published in this article

Financing: Research was carried out without financial support

Data availability: all data are available in digital or graphical form in the main text of the manuscript

Use of artificial intelligence: during the preparation of the manuscript the author used generative artificial-intelligence tools for language editing, formatting of the text and preparation of the illustrations; after using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of this publication

REFERENCES

1. Ho, J., Jain, A. & Abbeel, P. “Denoising diffusion probabilistic models”. *NeurIPS*. 2020. DOI: <https://doi.org/10.48550/arXiv.2006.11239>.
2. Song, J., Meng, C. & Ermon, S. “Denoising diffusion implicit models”. *ICLR*. 2021. DOI: <https://doi.org/10.48550/arXiv.2010.02502>.
3. Nichol, A. Q. & Dhariwal, P. “Improved denoising diffusion probabilistic models”. *Proceedings of the 38th International Conference on Machine Learning. PMLR*. 2021; 139: 8162–8171. DOI: <https://doi.org/10.48550/arXiv.2102.09672>.
4. Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. “High-resolution image synthesis with latent diffusion models”. *CVPR*, 2022. DOI: <https://doi.org/10.48550/arXiv.2112.10752>.
5. Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M. & Fleet, D. J. “Video diffusion models”. *NeurIPS*. 2022. DOI: <https://doi.org/10.48550/arXiv.2204.03458>.
6. Blattmann, A., Dockhorn, T., Kulal, S. et al. “Stable video diffusion: Scaling latent video diffusion models to large datasets”. *arXiv*. 2023. DOI: <https://doi.org/10.48550/arXiv.2311.15127>.
7. Peebles, W. & Xie, S. “Scalable diffusion models with transformers”. *ICCV*. 2023. DOI: <https://doi.org/10.48550/arXiv.2212.09748>.
8. Yang, Z., Teng, J., Zheng, W., et al. “CogVideoX: Text-to-Video diffusion models with an expert transformer”. *ICLR*, 2025. DOI: <https://doi.org/10.48550/arXiv.2408.06072>.
9. Ho, J., Chan, W., Saharia, C., et al. “Imagen video: High definition video generation with diffusion models”. *arXiv*. 2022. DOI: <https://doi.org/10.48550/arXiv.2210.02303>.
10. Singer, U., Polyak, A., Hayes, T., et al. “Make-A-Video: Text-to-Video generation without Text-Video Data”. *arXiv*. 2022. DOI: <https://doi.org/10.48550/arXiv.2209.14792>.
11. Blattmann, A., Rombach, R., Ling, H. et al. “Align your latent’s: High-resolution video synthesis with latent diffusion models”. *CVPR*. 2023. DOI: <https://doi.org/10.48550/arXiv.2304.08818>.
12. Guo, Y., Yang, C., Rao, A. et al. “AnimateDiff: Animate your personalised Text-to-Image diffusion models without specific tuning”. *arXiv*. 2023. DOI: <https://doi.org/10.48550/arXiv.2307.04725>.
13. Chen, H., Zhang, Y., Cun, X., et al. “VideoCrafter2: Overcoming data limitations for high-quality video diffusion models”. *CVPR*. 2024. DOI: <https://doi.org/10.48550/arXiv.2401.09047>.
14. Zheng, Z., Peng, X., Lou, Y., et al. “Open-Sora 2.0: Training a commercial-level video generation model in \$200k”. *arXiv*. 2025. DOI: <https://doi.org/10.48550/arXiv.2503.09642>.
15. Wan, T., Wang, A. & Ai, B., et al. “Wan: Open and advanced large-scale video generative models”. *arXiv*. 2025. DOI: <https://doi.org/10.48550/arXiv.2503.20314>.
16. Unterthiner, T., van Steenkiste, S., Kurach, K., et al. “Towards accurate generative models of video: A new metric & challenges”. *arXiv*. 2018. DOI: <https://doi.org/10.48550/arXiv.1812.01717>.
17. Huang Z., He, Y., Yu, J. et al. “VBench: Comprehensive benchmark suite for video generative models”. *arXiv*, 2023. DOI: <https://doi.org/10.48550/arXiv.2311.17982>.

18. Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J. “DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps”. *NeurIPS*. 2022. DOI: <https://doi.org/10.48550/arXiv.2206.00927>.
19. Wu, J. Z., Ge, Y., Wang, X., et al. “Tune-A-Video: One-shot tuning of image diffusion models for Text-to-Video generation”. *ICCV*, 2023. DOI: <https://doi.org/10.48550/arXiv.2212.11565>.
20. Khachatryan, L., Movsisyan, A., Tadevosyan, V., et al. “Text2Video-Zero: Text-to-Image diffusion models are Zero-Shot video generators”. *arXiv*. 2023. DOI: <https://doi.org/10.48550/arXiv.2303.13439>.
21. Luo, Z., Chen, D., Zhang, Y., et al. “VideoFusion: Decomposed diffusion models for high-quality video generation”. *CVPR*. 2023. DOI: <https://doi.org/10.48550/arXiv.2303.08320>.
22. Chen, H., Xia, M., He, Y., et al. “VideoCrafter1: Open diffusion models for high-quality video generation”. *arXiv*. 2023. DOI: <https://doi.org/10.48550/arXiv.2310.19512>.
23. Wang, Y., Chen, X., Ma, X., et al. “LAVIE: High-quality video generation with cascaded latent diffusion models”. *arXiv*. 2023. DOI: <https://doi.org/10.48550/arXiv.2309.15103>.
24. Zhang, D. J., Wu, J. Z., Liu, J.-W., et al. “Show-1: Marrying pixel and latent diffusion models for Text-to-Video gGeneration”. *arXiv*. 2023. DOI: <https://doi.org/10.48550/arXiv.2309.15818>.
25. Ma, X., Wang, Y., Chen, X., et al. “Latte: Latent diffusion transformer for video generation”. *Transactions on Machine Learning Research*. 2025. DOI: <https://doi.org/10.48550/arXiv.2401.03048>.
26. Kong, W., Tian, Q., Zhang, Z., et al. “HunyuanVideo: A systematic framework for large video generative models”. *arXiv*. 2024. DOI: <https://doi.org/10.48550/arXiv.2412.03603>.
27. Molad, E., Horwitz, E., Valevski, D., et al. “Dreamix: Video diffusion models are general video editors”. *arXiv*. 2023. DOI: <https://doi.org/10.48550/arXiv.2302.01329>.
28. Wang, X., Yuan, H., Zhang, S., et al. “VideoComposer: Compositional video synthesis with motion controllability”. *NeurIPS*. 2023. DOI: <https://doi.org/10.48550/arXiv.2306.02018>.

Received 03.09.2026

Received after revision 11.10.2026

Accepted 14.10.2026

DOI: <https://doi.org/10.15276/ict.03.2026.93>

УДК 004.032.26:004.932

Розробка моделей дифузії для генерації відеопослідовностей

Литвиненко Андрій Володимирович¹⁾

ORCID: <https://orcid.org/0000-0002-4611-7832>; akoshka1@gmail.com

Вороненко Марія Олександрівна¹⁾

ORCID: <https://orcid.org/0000-0002-5392-5125>; mary.voronenko@gmail.com

¹⁾ Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна

АНОТАЦІЯ

У цій статті представлено програмну реалізацію для генерації коротких відеопослідовностей на центральному процесорі (ЦП). Розроблено уніфікований інтерфейс для навчання компактної дифузійної моделі в піксельному просторі та застосування попередньо навчених моделей Stable Video Diffusion і CogVideoX. Навчальна модель використовує тривимірні згортки, обумовлення за кроком дифузії та мінімізацію помилки прогнозування шуму. Реалізовано методи вибірки на основі стохастичної ймовірнісної моделі дифузії з усуненням шуму (DDPM) та детермінованої неявної моделі дифузії з усуненням шуму (DDIM). Описано математичне формулювання, архітектуру мережі, параметри виконання, процедури експорту та діагностичне оцінювання. Представлено архітектуру програмного конвеєра для генерації відеопослідовностей на центральному процесорі. Система включає три альтернативні гілки: навчання та застосування компактної дифузійної моделі, генерацію відео із зображення та генерацію відео за текстовим описом. Усі гілки використовують спільні процедури експорту, оцінювання та моніторингу обчислювальних ресурсів. Аналітично виведено залежність кількості параметрів від ширини мережі та порівняно кількість викликів модуля усунення шуму для різних схем вибірки. Запропоновано протокол

оцінювання компромісу між обчислювальними витратами та характеристиками відео. Наведено результати попереднього тестування («smoke test») компактної моделі, що включало дві ітерації навчання та генерацію чотирьох кадрів розміром 8×8 пікселів. Тест підтвердив успішне виконання конвеєра, проте не продемонстрував формування узгодженої динаміки чи переваг над іншими моделями. Ця робота займає інженерно-методологічну нішу: вона поєднує навчання невеликої ймовірнісної моделі дифузії з усуненням шуму в піксельному просторі із застосуванням попередньо навчених моделей Stable Video Diffusion і CogVideoX, забезпечуючи при цьому збереження результатів і звітів про використання ресурсів. Її внеском є прозора реалізація та відтворюваний дослідницький протокол; питання переваги над великими моделями потребує окремої експериментальної перевірки.

Ключові слова: дифузійна модель; генерація відео; відеопослідовності; тривимірна згортка; латентна дифузія; CogVideoX; часова узгодженість; програмний конвеєр

ІНФОРМАЦІЯ ПРО АВТОРІВ



Andrey V. Lytvynenko - Junior Research Fellow, Institute of Computer Systems, Odesa Polytechnic National University, 1, Shevchenko Avenue. Odesa, 65044, Ukraine

ORCID: <https://orcid.org/0000-0002-4611-7832>; akoshka1@gmail.com

Research field: generative artificial intelligence models

Литвиненко Андрій Володимирович - молодший науковий співробітник, Інститут комп'ютерних систем. Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна

Галузь досліджень: генеративні моделі штучного інтелекту



Mariia O. Voronenko - Associate Professor, Department of Information Systems, Odesa Polytechnic National University, 1 Shevchenko Avenue. Odesa, 65044, Ukraine

ORCID: <https://orcid.org/0000-0002-5392-5125>; mary.voronenko@gmail.com

Research field: computer science, artificial intelligence, probabilistic models

Вороненко Марія Олександрівна - кандидат техніч. наук, доцент, доцент каф. Інформаційних систем. Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна

Галузь досліджень: комп'ютерні науки, штучний інтелект, ймовірнісні моделі