Quick Guide to Deforum v06

-EnzymeZoo -foxxie -huemin

Art by: neuro @ https://twitter.com/neurodiculous

This quick user guide is intended as a LITE reference for different aspects and items found within the Deforum notebook. It is intended for version 06, which was released 11/15/2022

While this reference guide includes different explanations of parameters, it is not to be used as a complete troubleshooting resource. The user is encouraged to explore and create their own style, using this guide as a compass to help better their inspiration. The best way to make this guide effective is to share your findings and experiences with the community! **-ScottieFox**

The AI art scene is evolving rapidly. Take this guide lightly. Methods, models, and notebooks will change. All the info in this guide will become irrelevant. Sad but true :'(**-huemin**

Table of Contents

Terms of Use

By using this Notebook, you agree to the following Terms of Use, and license:

This model is open access and available to all, with a CreativeML OpenRAIL-M license further specifying rights and usage.

The CreativeML OpenRAIL License specifies:

You can't use the model to deliberately produce nor share illegal or harmful outputs or content
CompVis claims no rights on the outputs you generate, you are free to use them and are accountable for their use which must not go against the provisions set in the license
You may re-distribute the weights and use the model commercially and/or as a service. If you do, please be aware you have to include the same use restrictions as the ones in the license and share a copy of the CreativeML OpenRAIL-M to all your users (please read the license entirely and carefully)

Please read the full license here:
https://huggingface.co/spaces/CompVis/stable-diffusion-license

Change Log

What's Changed

* Conditioning by @enzymezoo-code in https://github.com/deforum/stable-diffusion/pull/[55,56,114,115,117,124,131]
* Local dev by @deforum in https://github.com/deforum/stable-diffusion/pull/109
* Xformers efficient attention by @kabachuha in https://github.com/deforum/stable-diffusion/pull/108
* Aesthetic conditioning by @deforum in https://github.com/deforum/stable-diffusion/commit/81fc4f1b41825beb554df016057fe4a3db2b88f5
* k-samplers 0.10 by @XmYx in https://github.com/deforum-art/deforum-stable-diffusion/pull/14
* Masking for 2D and 3D video modes by @enzymezoo-code
* Added Disconnect for Colab by @makeitrad in https://github.com/deforum/stable-diffusion/pull/110
* Gif video creation by @vinliao
* Update README.md by @carriemorrison in https://github.com/deforum/stable-diffusion/pull/112
* Fixed typo in line 319 in generate.py by @Limbicnation in https://github.com/deforum/stable-diffusion/pull/127

New Contributors
* @carriemorrison made their first contribution in https://github.com/deforum/stable-diffusion/pull/112
* @Limbicnation made their first contribution in https://github.com/deforum/stable-diffusion/pull/127
* @XmYx made their first contribution in https://github.com/deforum-art/deforum-stable-diffusion/pull/14

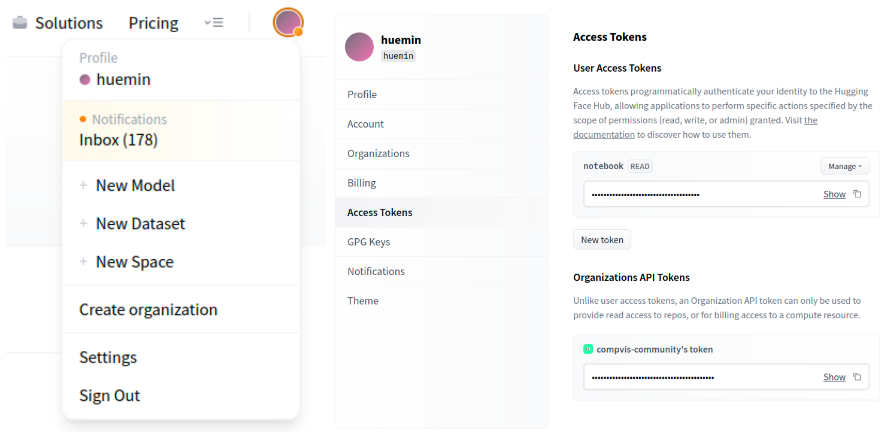
Models

Automatic Download

In Deforum there is an automatic model download feature. All model weights, when selected, are downloaded from huggingface and placed in the appropriate model folder.

To download official model weights within Deforum you will need to have an account on huggingface and provide your username and an access token within the notebook.

You can make an access token by going to your profile > settings > access tokens > new token



Access Tokens

User Access Tokens

Access tokens programmatically authenticate your identity to the Hugging Face Hub, allowing applications to perform specific actions specified by the scope of permissions (read, write, or admin) granted. Visit the documentation to discover how to use them.

Organizations API Tokens

Unlike user access tokens, an Organization API token can only be used to provide read access to repos, or for billing access to a compute resource.

Provide your username and access token when prompted in Deforum. Afterwards, the notebook will attempt to download the model. If you run into errors automatically downloading models, please try again, sometime the colab session times out.

Deforum v05 has the following models configured for automatic download:

Official Stable Diffusion Weights (requires huggingface login and token)

Unofficial Nousr Robo Diffusion
- robo-diffusion-v1.ckpt

Unofficial Waifu Diffusion v3
- waifu-diffuion-v3.ckpt

Manual Download

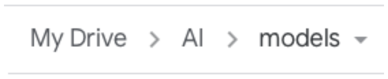
Make an account on huggingface, download the .ckpt file, and place the file in Google Drive. The Deforum Stable Diffusion notebook requires the user to download model weights (~4GB) and correctly link the model weights to the Colab Notebook. The following steps will walk you through downloading model weights and uploading them to google drive:

1. Go to https://huggingface.co and sign up to create an account

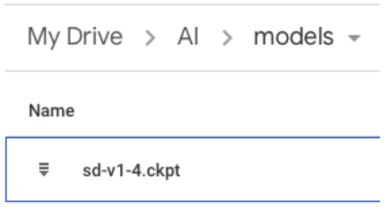
2. Once signed into your account, navigate to https://huggingface.co/CompVis. Here you will see all the checkpoints available for download marked with "-original".

3. Select an "-original" model from the "CompVis" library (stable-diffusion-v-1-4-original) and download the weights. You will need to accept the terms of use. At the time of this writing sd-v1-4.ckpt is the best model.

4. While you are waiting for the model to download to your computer, open the Deforum Notebook and run the "Model and Output Paths" cell by clicking the play button. Running this cell will configure your Google Drive with the correct folder and file structure. Alternatively, you can create the following folders on your google drive:



5. Once the download is complete you will need to upload the model weights to this models folder.

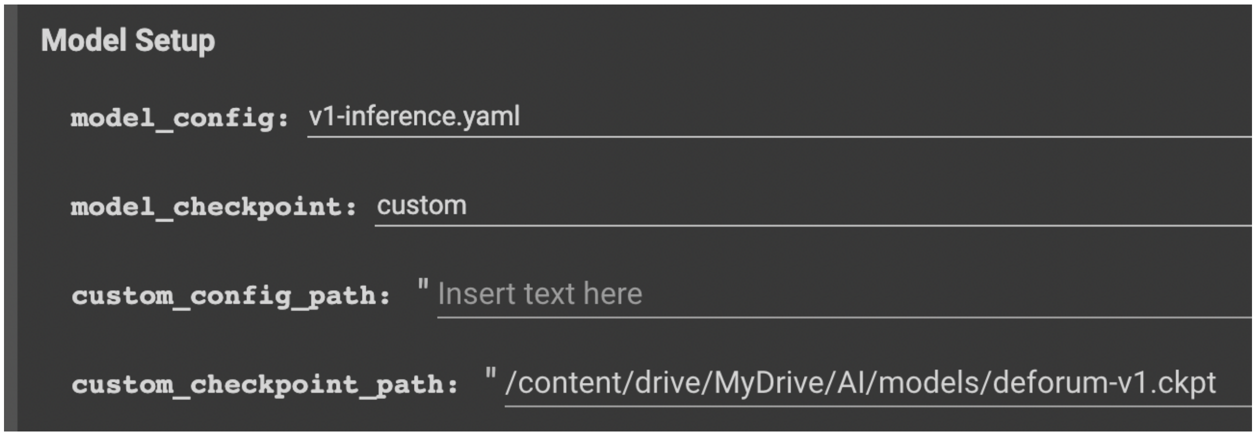


6. Ready to go!

Custom Models

Custom model checkpoints (.ckpt) can be loaded into deforum either by choosing “custom” in the model checkpoint drop down and specifying the full model path or by placing the model .ckpt in the model folder.

Choosing “custom” in the model_checkpoint drop down will enable the custom_checkpoint_path



Model Setup

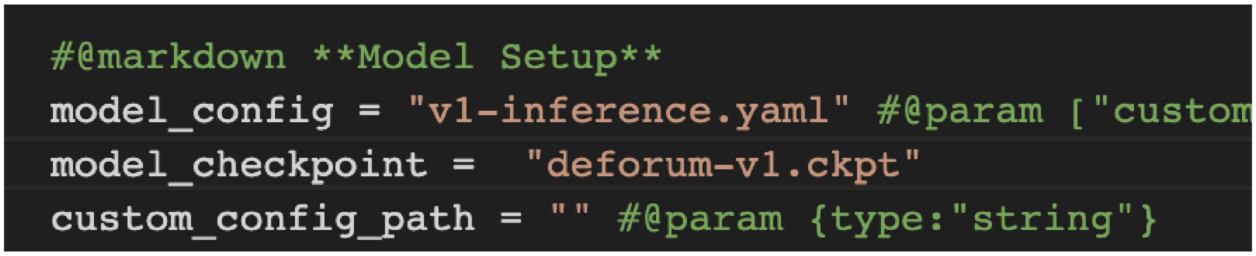
model_config: v1-inference.yaml

model_checkpoint: custom

custom_config_path: " Insert text here

custom_checkpoint_path: " /content/drive/MyDrive/AI/models/deforum-v1.ckpt

You can also replace the model_checkpoint with the name of any .ckpt file. The model will only load if the .ckpt file is in the models_path folder (the “models” folder by default).



```
#@markdown **Model Setup**
model_config = "v1-inference.yaml" #@param ["custom
model_checkpoint =  "deforum-v1.ckpt"
custom_config_path = "" #@param {type:"string"}
```

Notebook Overview

NVIDIA GPU:

- will display info about available GPUs on your system or session.
  This cell will give you information regarding the gpu you have connected to in the run session. Diffusion in general makes heavy use of VRAM (video RAM) to render images. Colab GPU tier list from best to worst: A100 (40GB VRAM), V100 (16GB VRAM), P100 (16GB VRAM), T4, K80.

Setup

Model and Output Paths:

- models_path: -looks in runtime for uploaded model
- output_path: -directs images/file to a place in the runtime
-

Google Drive Path Variables (Optional):

- mount_google_drive: when selected will redirect paths to drive instead of runtime
- models_path_gdrive: location of model on Google Drive
  (default /content/drive/MyDrive/AI/models)
- output_path_gdrive: location of images/file to be output in Google Drive
  The notebook expects the following path variables to be defined: models_path and output_path. These locations will be used to access the Stable Diffusion .pth model weights and save the diffusion output renders, respectively. There is the option to use paths locally or on Google Drive. If you desire to use paths on Google drive, mount_google_drive must be True. Mounting Gdrive will prompt you to access your Drive, to read/write/save images.

Setup Environment:

- setup_enviroment: when checked will build environment to handle pip/installs/imports
- print_subprocess: choose to show items being pulled and built
  Running this cell will download github repositories, import python libraries, and create the necessary folders and files to configure the Stable Diffusion model. Sometimes there may be issues where the Setup Environment cells do not load properly and you will encounter errors when you start the run. Verify the Setup Environment cells have been run without any errors.

Select and Load Model:

- model_config: type of instruction file: default .yaml, or custom option
- model_checkpoint: the dataset to auto downloaded
- custom_config_path: blank unless intending to use a custom .yaml file
- custom_checkpoint_path: blank unless using a .cpkt file not listed
- load_on_run_all: when checked will be an include cell for RUN ALL function
- check_sha256: will perform comparison against checksum (check hash for file integrity)

- map_location: utilizes CUDA cores on GPU[default], or uses CPU[slow]
  In order to load the Stable Diffusion model, Colab needs to know where to find the model_config file and the model_checkpoint. The model_config file contains information about the model architecture. The model_checkpoint contains model weights which correspond to the model architecture. For troubleshooting verify that both the config and weight path variables are correct. By default the notebook expects the model config and weights to be located in the model_path. You can provide custom model weights and config paths by selecting “custom” in both the model_config and model_checkpoint dropdowns. Sometimes there are issues with downloading the model weights and the file is corrupt. The check_sha256 function will verify the integrity of the model weights and let you know if they are okay to use. The map_location allows the user to specify where to load model weights. For most colab users, the default “GPU” map location is best.

settings on next page →

Settings

Animation modes:

- **NONE**, When selected, will ignore all functions in animation mode and will output batches of images coherently unrelated to each other, as specified by the prompts list. The prompts used will follow the non-scheduled, non-animation list. The number of images that are to be produced is defined in a later cell under “n_batches”.
- **2D**: When selected will ignore the “none mode” prompts and refer to the prompts that are scheduled with a frame number before them. 2D mode will attempt to string the images produced in a sequence of coherent outputs. The number of output images to be created is defined by “max_frames”. The motion operators that control 2D mode are as follows: “Border, angle, zoom, translation_x, translation_y, noise_schedule, contrast_schedule, color_coherence, diffusion_cadence, and save depth maps”. Other animation parameters have no effect during 2D mode. Resume_from_timestring is available during 2D mode. (more details below)
- **3D**, When selected will ignore the “none mode” prompts and refer to the prompts that are scheduled with a frame number before them. 3D mode will attempt to string the images produced in a sequence of coherent outputs. The number of output images to be created is defined by “max_frames”. The motion operators that control 3D mode are as follows: “Border, translation_x, translation_y, rotation_3d_x, rotation_3d_y, rotation_3d_z, noise_schedule, contrast_schedule, color_coherence, diffusion_cadence, 3D depth warping, midas_weight, fov, padding_mode, sampling_mode, and save_depth_map. Resume_from_timestring is available during 3D mode. (more details below)
- **video_input**, When selected, will ignore all motion parameters and attempt to reference a video loaded into the runtime, specified by the video_init_path. Video Input mode will ignore the “none mode” prompts and refer to the prompts that are scheduled with a frame number before them. “Max_frames” is ignored during video_input mode, and instead, follows the number of frames pulled from the video’s length. The notebook will populate images from the video into the selected drive as a string of references to be impacted. The number of frames to be pulled from the video is based on “extract_nth_frame”. Default of 1 will extract every single frame of the video. A value of 2 will skip every other frame. Values of 3 and higher will effectively skip between those frames yielding a shorter batch of images. Currently, video_input mode will ignore all other coherence parameters, and only affect each frame uniquely. Resume_from_timestring is NOT available with Video_Input mode.
- **interpolation_mode**, When selected, will ignore all other motion and coherence parameters, and attempt to blend output frames between animation prompts listed with a schedule frame number before them. If interpolate_key_frame mode is checked, the number of output frames will follow your prompt schedule. If unselected, the interpolation mode will follow an even schedule of frames as specified by “interpolate_x_frames”, regardless of prompt numbering. A default value of 4 will yield four frames of interpolation between prompts.

Animation Parameters:

- **animation_mode**, selects type of animation (see above)
- **max_frames**, specifies the number of 2D or 3D images to output
- **border**, controls handling method of pixels to be generated when the image is smaller than the frame. “Wrap” pulls pixels from the opposite edge of the image, while “Replicate” repeats the edge of the pixels, and extends them. Animations with quick motion may yield “lines” where this border function was attempting to populate pixels into the empty space created.

Motion Parameters:
motion parameters are instructions to move the canvas in units per frame
- **angle**, 2D operator to rotate canvas clockwise/anticlockwise in degrees per frame
- **zoom**, 2D operator that scales the canvas size, multiplicatively [static = 1.0]
- **translation_x**, 2D & 3D operator to move canvas left/right in pixels per frame
- **translation_y**, 2D & 3D operator to move canvas up/down in pixels per frame
- **translation_z**, 3D operator to move canvas towards/away from view [speed set by FOV]
- **rotation_x**, 3D operator to tilt canvas up/down in degrees per frame
- **rotation_y**, 3D operator to pan canvas left/right in degrees per frame
- **rotation_z**, 3D operator to roll canvas clockwise/anticlockwise
- **flip_2D_perspective**, enables 2D mode functions to simulate “faux” 3D movement
- **perspective_flip_theta**, the “roll” effect angle
- **perspective_flip_phi,** the “tilt” effect angle
- **perspective_flip_gamma**, the “pan” effect angle
- **perspective_flip_fv**, the 2D vanishing point of perspective (rec’d range 30-160)
- **noise_schedule**, amount of graininess to add per frame for diffusion diversity
- **strength_schedule**, amount of presence of previous frame to influence next frame, also controls steps in the following formula [steps - (strength_schedule * steps)] (more details under: “steps”)
- **contrast_schedule**, adjusts the overall contrast per frame [default neutral at 1.0]

Coherence:
- **color_coherence**, select between NONE, LAB, HSV, RGB
    - **LAB**: Perceptual **L**ightness* **A** * **B** axis color balance (search “cielab”)
    - **HSV**: **H**ue **S**aturation & **V**alue color balance.
    - **RGB**: **R**ed **G**reen & **B**lue color balance.

  The color coherence will attempt to sample the overall pixel color information, and trend those values analyzed in the 0th frame, to be applied to future frames. LAB is a more linear approach to mimic human perception of color space - a good default setting for most users.

  HSV is a good method for balancing presence of vibrant colors, but may produce unrealistic results - (ie.blue apples) RGB is good for enforcing unbiased amounts of color in each red, green and blue channel - some images may yield colorized artifacts if sampling is too low.

- **diffusion_cadence**, controls the frequency of frames to be affected by diffusion [1-8]
  The diffusion cadence will attempt to follow the 2D or 3D schedule of movement as per specified in the motion parameters, while enforcing diffusion on the frames specified. The default setting of 1 will cause every frame to receive diffusion in the sequence of image outputs. A setting of 2 will only diffuse on every other frame, yet motion will still be in effect. The output of images during the cadence sequence will be automatically blended, additively and saved to the specified drive. This may improve the illusion of coherence in some workflows as the content and context of an image will not change or diffuse during frames that were skipped. Higher values of 4-8 cadence will skip over a larger amount of frames and only diffuse the "Nth" frame as set by the diffusion_cadence value. This may produce more continuity in an animation, at the cost of little opportunity to add more diffused content. In extreme examples, motion within a frame will fail to produce diverse prompt context, and the space will be filled with lines or approximations of content - resulting in unexpected animation patterns and artifacts. Video Input & Interpolation modes are not affected by diffusion_cadence.

3D Depth Warping:

- **use_depth_warping**, enables instructions to warp an image dynamically in 3D mode only.
- **midas_weight**, sets a midpoint at which a depthmap is to be drawn: range [-1 to +1]
- **fov**, adjusts the scale at which a canvas is moved in 3D by the translation_z value
  FOV (field of view/vision) in deforum, will give specific instructions as to how the translation_z value affects the canvas. Range is -180 to +180. The value follows the inverse square law of a curve in such a way that 0 FOV is undefined and will produce a blank image output. A FOV of 180 will flatten and place the canvas plane in line with the view, causing no motion in the Z direction. Negative values of FOV will cause the translation_z instructions to invert, moving in an opposite direction to the Z plane, while retaining other normal functions.A value of 30 fov is default whereas a value of 100 would cause transition in the Z direction to be more smooth and slow. Each type of art and context will benefit differently from different FOV values. (ex. "Still-life photo of an apple" will react differently than "A large room with plants")

  FOV also lends instruction as to how a midas depth map is interpreted. The depth map (a greyscale image) will have its range of pixel values stretched or compressed in accordance with the FOV in such a fashion that the illusion of 3D is more pronounced at lower FOV values, and more shallow at values closer to 180. At full FOV of 180, no depth is perceived, as the midas depth map has been compressed to a single value range.

- **padding_mode**, instructs the handling of pixels outside the field of view as they come into the scene. 'Border" will attempt to use the edges of the canvas as the pixels to be drawn. "Reflection" will attempt to approximate the image and tile/repeat pixels, whereas "Zeros" will not add any new pixel information.
- **sampling_mode**, choose from Bicubis, Bilinear or Nearest modes.

In image processing, bicubic interpolation is often chosen over bilinear or nearest-neighbor interpolation in image resampling, when speed is not an issue. In contrast to bilinear interpolation, which only takes 4 pixels (2×2) into account, bicubic interpolation considers 16 pixels (4×4). Images resampled with bicubic interpolation are smoother and have fewer interpolation artifacts.

- **save_depth_map**, will output a greyscale depth map image alongside the output images.

Video Input:

- **video_init_path**, the directory at which your video file is located for Video INput mode only.
- **extract_nth_frame**, during the run sequence, only frames specified by this value will be extracted, saved, and diffused upon. A value of 1 indicates that every frame is to be accounted for. Values of 2 will use every other frame for the sequence. Higher values will skip that number of frames respectively.
- **overwrite_extracted_frames**, when enabled, will re-extract video frames each run. When using video_input mode, the run will be instructed to write video frames to the drive. If you've already populated the frames needed, uncheck this box to skip past redundant extraction, and immediately start the render. If you have not extracted frames, you must run at least once with this box checked to write the necessary frames.
- **use_video_mask**, video_input mode only, enables the extraction and use of a separate video file intended for use as a mask. White areas of the extracted video frames will not be affected by diffusion, while black areas will be fully effected. Lighter/darker areas are affected dynamically.
- **video_mask_path**, the directory in which your mask video is located.

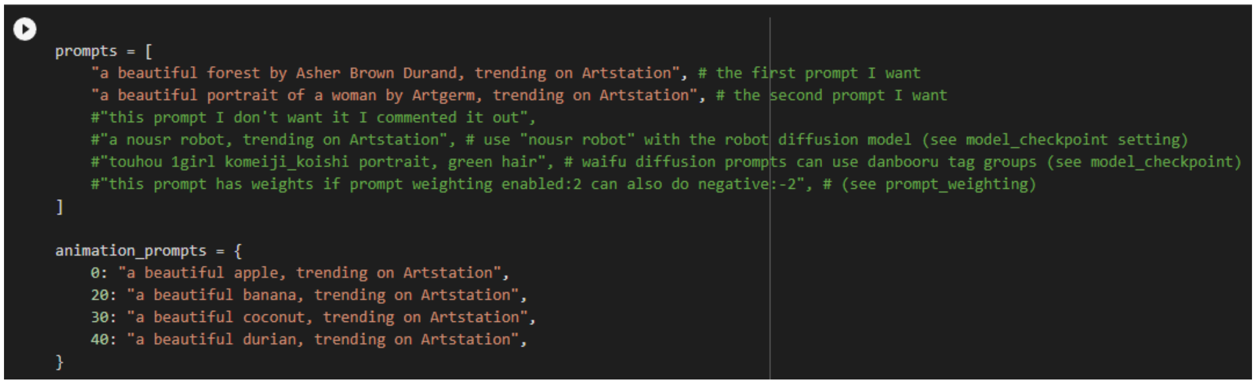
Interpolation:

- **interpolate_key_frames**, selects whether to ignore prompt schedule or _x_frames.
- **interpolate_x_frames**, the number of frames to transition thru between prompts (when interpolate_key_frames = true, then the numbers in front of the animation prompts will dynamically guide the images based on their value. If set to false, will ignore the prompt numbers and force interpole_x_frames value regardless of prompt number)

Resume Animation:

- **resume_from_timestring**, instructs the run to start from a specified point
- **resume_timestring**, the required timestamp to reference when resuming
  Currently only available in 2D & 3D mode, the timestamp is saved as the settings .txt file name as well as images produced during your previous run. The format follows: yyyymmddhhmmss - a timestamp of when the run was started to diffuse.

Prompts



```
prompts = [
    "a beautiful forest by Asher Brown Durand, trending on Artstation", # the first prompt I want
    "a beautiful portrait of a woman by Artgerm, trending on Artstation", # the second prompt I want
    #"this prompt I don't want it I commented it out",
    #"a nousr robot, trending on Artstation", # use "nousr robot" with the robot diffusion model (see model_checkpoint setting)
    #"touhou 1girl komeiji_koishi portrait, green hair", # waifu diffusion prompts can use danbooru tag groups (see model_checkpoint)
    #"this prompt has weights if prompt weighting enabled:2 can also do negative:-2", # (see prompt_weighting)
]

animation_prompts = {
    0: "a beautiful apple, trending on Artstation",
    20: "a beautiful banana, trending on Artstation",
    30: "a beautiful coconut, trending on Artstation",
    40: "a beautiful durian, trending on Artstation",
}
```

In the above example, we have two groupings of prompts: the still frames *prompts* on top, and the animation_prompts below. During the “NONE” animation mode, the diffusion will look to the top group of prompts to produce images. In all other modes, (2D, 3D etc) the diffusion will reference the second lower group of prompts.

Careful attention to the syntax of these prompts is critical to be able to run the diffusion.
For still frame image output, numbers are not to be placed in front of the prompt, since no “schedule” is expected during a batch of images. The above prompts will produce and display a forest image and a separate image of a woman, as the outputs.

During 2D//3D animation runs, the lower group with prompt numbering will be referenced as specified. In the example above, we start at frame 0: - an apple image is produced. As the frames progress, it remains with an apple output until frame 20 occurs, at which the diffusion will now be directed to start including a banana as the main subject, eventually replacing the now no longer referenced apple from previous.

Interpolation mode, however, will “tween” the prompts in such a way that firstly, 1 image each is produced from the list of prompts. An apple, banana, coconut, and a durian fruit will be drawn. Then the diffusion begins to draw frames that should exist between the prompts, making hybrids of apples and bananas - then proceeding to fill in the gap between bananas and coconuts, finally resolving and stopping on the last image of the durian, as its destination. (remember that this exclusive mode ignores max_frames and draws the interpolate_key_frame/x_frame schedule instead.

Many resources exist for the context of what a prompt should include. It is up to YOU, the dreamer, to select items you feel belong in your art. Currently, prompts weights are not implemented yet in deforum, however following a template should yield fair results:

[Medium]	[Subject]	[Artist]	[Details]	[Repository]

Ex. “A Sculpture of a Purple Fox by Alex Grey, with tiny ornaments, popular on CGSociety”,

Run

Load Settings:

- **override_settings_with_file**, when checked ,ignores all settings and refers to a .txt file
- **custom_settings_file**, location of settings file to be used for override instructions

Image settings:

- **W**, defines the output width of the final image in pixels
- **H**, defines the output height of the final image in pixels
  Dimensions in output must be multiples of 64 pixels otherwise, the resolution will be rounded down to the nearest compatible value. Proper values 128, 192, 256, 320, 384, 448, 512, 576, 640, 704, 768, 832, 896, 960, 1024. Values above these recommended settings are possible, yet may yield OOM (out of memory) issues, as well as improper midas calculations. The model was trained on a 512x512 dataset, and therefore must extend its diffusion outside of this “footprint” to cover the canvas size. A wide landscape image may produce 2 trees side-by-side as a result, or perhaps 2 moons on either side of the sky. A tall portrait image may produce faces that are stacked instead of centered.

Sampling Settings:

- **seed**, a starting point for a specific deterministic outcome, (-1 = random starting point)
  Stable Diffusion outputs are deterministic, meaning you can recreate images using the exact same settings and seed number. Choosing a seed number of -1 tells the code to pick a random number to use as the seed. When a random seed is chosen, it is printed to the notebook and saved in the image settings .txt file.
- **sampler**, method in which the image is encoded and decoded from latent space
  - **klms** = Kernel Least Mean Square
  - **dpm2** = Denoise Probabilistic Model
  - **dpm2**_Ancestral = dpm2 with reverse sampling path
  - **heun** = founded off of Euler by Karl Heun (maths & derivative solving)
  - **euler** = fractional-order anisotropic denoise (Euler-Lagrange equations)
  - **euler_ancestral** = reverse sampling path to Euler
  - **plms** = Pre-trained Language Model(s)
  - **ddim** = Denoising Diffusion Probabilistic Models
- **steps**, the number of iterations intended for a model to reach its prompt
  Considering that during one frame, a model will attempt to reach its prompt by the final step in that frame. By adding more steps, the frame is sliced into smaller increments as the model approaches completion. Higher steps will add more defining features to an output at the cost of time. Lower values will cause the model to rush towards its goal, providing vague attempts at your prompt. Beyond a certain value, if the model has achieved its prompt, further steps will have very little impact on final output, yet time will still be a wasted resource. Some prompts also require fewer steps to achieve a desirable acceptable output.

During 2D & 3D animation modes, coherence is important to produce continuity of motion during video playback. The value under Motion Parameters, “strength_schedule” achieves this coherence by utilizing a proportion of the previous frame, into the current diffusion. This proportion is a scale of 0 - 1.0 , with 0 meaning there’s no cohesion whatsoever, and a brand new unrelated image will be diffused. A value of 1.0 means ALL of the previous frame will be utilized for the next, and no diffusion is needed. Since this relationship of previous frame to new diffusion consists of steps diffused previously, a formula was created to compensate for the remaining steps to justify the difference. That formula is as such:
Target Steps - (strength_schedule * Target Steps)

Your first frame will, however, yield all of the steps - as the formula will be in effect afterwards.

- **scale**, a measurement of how much enforcement to apply to an overall prompt.
  A normal range of 7-10 is appropriate for most scenes, however some styles and art will require more extreme values. At scale values below 3, the model will loosely impose a prompt with many areas skipped and left uninteresting or simply grayed-out. Values higher than 25 may over enforce a prompt causing extreme colors of over saturation, artifacts and unbalanced details. For some use-cases this might be a desirable effect. During some animation modes, having a scale that is too high, may trend color into a direction that causes bias and overexposed output.

- **ddim_eta**, ONLY enabled in ddim sampler mode, will control a ratio of ddim to ddpm sampling methods, with a range of -1 to +1 with 0 being less randomized determinism.

Save & Display Settings:
- **save_samples**, will save output images to the specified drive, including cadence frames
- **save_settings**, will save a snapshot .txt of all settings used to start a run with a timestamp
- **display samples**, shows on-screen image of the completed output
- **save_sample_per_step**, outputs all intermediate steps of a single frame (many files)
- **show_sample_per_step**, displays all images of each step of the output

Prompt Settings:
- **prompt_weighting**, enables interpretation of weight syntax in a prompt
- **normailze_prompt**_weights, multiplies by a factor to have sum of all = 1.0
- **log_weighted_subprompts**, displays tokenization of prompt context

Batch Settings:
- **n_batch**, produces n amounts of outputs per prompt in ‘none’ animation mode
- **batch_name**, will create a folder and save output content to that directory location

- **seed behavior**, will perform progressive changes on the seed starting point based on settings:
  Iter = incremental change (ex 77, 78, 79 ,80, 81, 82, 83…)
  Fixed = no change in seed (ex 33, 33, 33, 33, 33, 33…)
  Random = random seed (ex 472, 12, 927812, 8001, 724…)
  Note: seed -1 will choose a random starting point, following the seed behavior thereafter
  Troubleshoot: a “fixed” seed in 2D/3D mode will overbloom your output. Switch to “iter”
- **make_grid**, will take take still frames and stitch them together in a preview grid
- **grid_rows**, arrangement of images set by make_grid

Init_Settings:
- **use_init**, uses a custom image as a starting point for diffusion
- **strength**, determines the presence of an init_image/video on a scale of 0-1 with 0 being full diffusion, and 1 being full init source.
  Note: even with use_init unchecked, video input is still affected.
- **init_image**, location of an init_image to be used
  Note: in ‘none’ animation mode, a folder of images may be referenced here.
- **use_mask**, adds an image for instructions as to which part of an image to diffuse by greyscale
- **mask_file**, location of the mask image to be used
- **invert_mask**, ranges the greyscale of a mask from “0 to 1” into “1 to 0”
- **mask_brightness_adjust**, changes the value floor of the mask, controlling diffusion overall
- **mask_constract_adjust**, clamps min/max values of the mask to limit areas of diffusion.
  Note: lighter areas of the mask = no diffusion, darker areas enforce more diffusion

Video

- **skip_video_for_run_all**, when running-all this notebook, video construction will be skipped until manually checked and the cell is re-run. It is off by default.
- **fps**, framerate at which the video will be rendered
- **image_path**, location of images intended to be stitched in sequence. The user must update this parameter to reflect the timestamp needed.
- **mp4_path**, location to save the resulting video to
- **max_frames**, the quantity of images to be prepared for stitching

MATHs in Deforum

This guide is an introduction to some of the parameters within deforum that can be controlled and altered using math expressions and functions. It aims to assist intermediate and advanced users of animation scheduling, by providing examples and descriptions of use cases. MATH functions are not mandatory in the functionality of deforum's parameters, and only serve as a dynamic tool to better enhance manipulation of values during an animation.

The specific tool documentation that has been added to Deforum V05 can be found here:
NumExpr 2.0 User Guide — numexpr 2.6.3.dev0 documentation

Parameters

In deforum, any parameter that accepts a string format of instructions (type = `string`) can be altered using a math expression, a schedule, or a combination of both. These parameters are typically denoted with 0:(0) where the preceding number is the frame, and the parentheses number is the value to be enforced during the designated frame. In the example of 0:(0), the render will reference frame0 and assign 0.0 as its value indefinitely unless instructed otherwise.

Parameters that are controlled by strings are as follows: *angle, zoom, translations_xyz, rotations_3D_xyz, perspective_flips_theta,phi,gamma,fv , noise_schdule, strength_schedule, and contrast_schdule.*

Scheduled values will "tween" linearly between two instructional elements in a string. In the example of 0:(-2), 100:(4) The render will start at frame0 with a value of -2 and rise up over time, increasing its value to 4 by the time it reaches frame100. During frame50 of that render, we would observe a value of 1.0 being enforced, since the midpoint between frame0 and frame100 falls on the line drawn between the two values at 1.0

When using math expressions however, the "tweening" follows an approximation of values within elements of the string in such a way that a curve is drawn between values. Consider the following example: 0:(sin(t)), 100:(4) The function at frame 0 in this case is a sine wave, where "t" represents the frame number. The value at frame 0 will start to calculate the sin(t) to produce its initial value, and quickly fluctuate causing peaks and valleys, while it slowly climbs to a constant value of 4 by frame100. We can observe an effect of the sine wave starting at full strength, and finally losing all amplitude 100 frames later - a "ripple" effect.

If a math expression is used as the sole element of a string, it will indefinitely calculate and produce its value for as long as it is defined, without interruption. If at any point, a parameter falls out of the range of acceptable values, the render will adhere to the next available calculation of that function. (ex. A value approaches infinity, asymptotic or undefined) This can sometimes be a desired effect if a "pulsing" or "sawtooth" function is to be achieved.

MATH expressions

Many combinations and complex functions can be expressed during an animation schedule to achieve patterns and motion that would otherwise take extremely long strings of manual information to achieve. Consider a sine function, where previously, we would have to enter in each frame's respective value to simulate a waving pattern. The longer our animation, the more frame instructions we'd have to manually enter. Now, with MATH functions, we can populate a never-ending list of instructions simply contained in one expression. The method that we use is to reference the variable "t". When we use that variable in our math statements, a calculation is performed such that "t" = the current frame number. Since the frame number steadily increases in increments of +1, we can now define an "x axis". With that aspect in place, we can use "t" to alter the value across the "y axis" in sequence. A frames (time) progresses forward, the MATHs performed on "t" will allow us to control what values are to be enforced at that exact snapshot in time. In the default notebook of deforumV05, the "translation_x" schedule is defined as: 0:(10*sin(2*3.14*t/10)) We can see "t" along with a sine wave (sin) being performed. This will cause the image to translate left and right over time. We will examine in more detail how this function works.

Anatomy MATH expressions

We saw the expression 0:(10*sin(2*3.14*t/10)) being used in the default notebook of deforumV05. Let's observe how it is "driving" our parameter. When we use the most simple of math expressions 0:(t) we define the value at any frame to be equal to its frame number. However, this value will soon rise off into unusable values above any recommended range within the animation parameters. At frame0, we start at 0, by frame1, we're at 1, and by frame 200, we're at 200 - so on and so forth. So a method of "containing" this value must be expressed somehow, as to prevent the number from flying off into infinity. The 2 best methods are sine/cosine functions as well as modulus functions (more info on modulus later).

So, in our example, we can see a "sin( )" being used. If we were to take the sine of our frame number, or "sin(t)", we'd generate a wave shape. The value would swing up and down quickly as each frame was calculated.

While this does keep our value from ever increasing - it is not enough to control our parameter in a realistic way. A simple sine wave is too fast, shallow and rapid. So our example includes more expressions being performed. We see that a familiar value 3.14 is multiplied by "t". This causes the period of our sine wave to fall on integers (approximately) at its wavelength. More specifically, this wavelength is 2. So our example goes further to multiply that variable by 2 also. When we take the sin(2*3.14*t) , we yield a wave that has a period of 1 and an amplitude of 1 (it peaks and valleys between -1 and 1). All that is left is to add math that will control how high the value should bounce(amplitude), and how often(frequency). So our example finally multiplies the whole expression by 10, and also divides “t” by 10. This results in a wave that will alternate between +10 and -10 and repeat every 10 frames. → cont on next page

->cont.

But what if we wanted even MORE control. We notice our example suffers the property of always passing through 0 as its baseline - but what if we wanted the baseline to start at -3? We just need to take the whole expression, and subtract 3 from it, and our new baseline is established. 0:(10*sin(2*3.14*t/10)-3) Now our wave bounces between 7 and -13, keeping its amplitude and frequency intact. More functionality can be added as we build our expression, including exponents, cosine properties, and negative amplitudes.

A recap of our examples anatomy: 0:(10*sin(2*3.14*t/10))
0: = the current frame instruction
10 = the amplitude or “height”
t = frame count
10 = the frequency or wavelength

Advanced Expressions

When constructing a complex schedule of effects during your animation, more control and special techniques will yield a better dynamic result. Let’s examine a specific use case. The artist wants to use a constant value of 0.8 as their strength schedule. However, they wish they could have more detail appear in their animation. A value of 0.45 is great for adding new enriched content to a scene, but it causes very little coherency. The artist decides that they should only introduce the value of 0.45 periodically about every 25 frames, yet keep it 0.8 for most of the sequence. How should the artist express this using MATHs?
Let’s observe the following solution, then discuss.
0:(-0.35*(cos(3.141*t/25)**100)+0.8)
A massively powerful function, with a simple elegance to it. Our artist uses this function in their example to achieve the desired result. At frame 0 and all frames after, a value is being calculated. In this expression we’re selecting a cosine function (cos) to allow our wave to have small periodic dips instead of peaks. The double asterisk acts as an exponent function and brings the cosine to the 100th power, tightening the dips into small indents along the timeline. The addition of +0.8 sets the baseline at 0.8 which the artist agreed was desirable for the animation, and starts the function with -0.35 knowing that it will dip below the established baseline from 0.8 down to 0.45 as expected. An approximation of pi is being used again (3.141) to align the frames to integers, and t is being divided by 25 to enforce the dip to occur only at frames that are multiples of 25. Our artist has achieved the schedule using one expression that will be calculated for the duration of the animation frames.

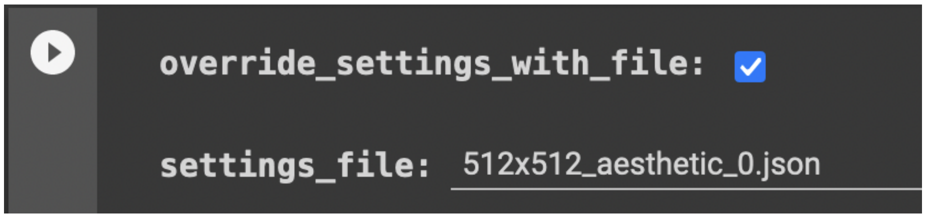
Remember that expressions can be changed along the schedule to “tween” along the frames.
0:(10*sin(2*3.14*t/10)), 50:(20*sin(4*3.14*t/40)), 100:(cos(t/10)) is an acceptable format.
Another useful tool is the modulus function. Represented by “%” is typically used to calculate the remainder of a function. In deforum, we use modulus to affect “t” frame count as a repeating limiter. Consider the following syntax:
translate_3D_z: 0:(0.375*(t%5)+15)
If “t” is the frame count, it would increase indefinitely, however in our example, we’ve set the modulus to 5. This means as the frame rises (01,2,3,4,5,6,7,8… etc) the value of “t” will repeat a sequence of 0,1,2,3,4,5,0,1,2,3,4,5,0,1… etc, without ever increasing over 5. This graphically produces a sawtooth wave. In order to bend the “blades” of the sawtooth to stretch over time, we multiply by 0.375. This acts as the slope of each line. A multiplier of 1 would yield a 45° line. Higher multipliers will increase the frequency even further, while numbers closer to 0 will lay the line near flat. Since we’re controlling the Z translate in 3D mode, we want our baseline to be at 15, hence our addition of it at the end of the syntax. The overall effect of this parameter causes our animation to consistently zoom forward, yet with pulses, similar to the perspective of nodding your head to music while riding in a car.

Many more clever approaches can be used to create elaborate functions and animations, as well as just simplifying the instruction of long frame counts. There are tools that exist such as graphing calculators to better help envision what a function would look like linearly. This can be simulated by the format of y = x instead of deforum’s 0: (t) where “y” is the frame, and “x” is t
This calculator can be used to solve similar functions, yet some syntax may vary.
Desmos | Graphing Calculator

We encourage the users to share their experiences with formulas and expressions, since there will be endless discoveries with how MATHs can work in unique applications.

Gradient Conditioning in Deforum

Having trouble finding good gradient conditioning settings? Try loading one of the settings files from the "settings" folder with override_settings_with_file.



```
override_settings_with_file: ✓

settings_file: 512x512_aesthetic_0.json
```

Here is a spreadsheet with recommended settings:
Deforum Gradient Conditioning Spreadsheet

If you want gritty details about how gradient conditioning works in general, here is an excellent notebook by Johnothan Whitaker: (colab notebook) (twitter post)

Exposure/Contrast Conditional Settings

mean_scale
  Pushes the pixel values towards middle gray
  Samples have values between -1 and 1. Mean_scale guides pixels toward 0, hence gray

var_scale
  Pushes pixels towards lower variance

exposure_scale
  Targeted mean loss. Uses exposure_target variable.
  When exposure_target == 0, exposure_scale is equivalent to mean_scale
  Use it to compensate for very high cfg scale, for example.

exposure_target [-1.0 to 1.0]
  Used with exposure_scale
  Negative values push towards a darker image, positive values toward a brighter image

Color Match Conditional Settings

colormatch_scale
  Guides the image towards a color palette.
  Palette is extracted from an input image colormatch_image
  Best matching results when decode_method == "autoencoder"
  Works best when gradient_wrt == "x0_pred", but will still work well with "x"

colormatch_image
    Image to extract a color palette from
colormatch_n_colors
    Number of colors in the palette
ignore_sat_weight (default None)
    Amount to ignore the saturation when using colormatch_scale.
    High ignore_sat_weight will allow a higher colormatch_scale without making the colors look overblown

CLIP\Aesthetics Conditional Settings

clip_name ['ViT-L/14', 'ViT-L/14@336px', 'ViT-B/16', 'ViT-B/32']
    Used with aesthetics_scale>0 and clip_scale>0
    Recommended to use with gradient_wrt='x'

aesthetics_scale
    Recommended to use with gradient_wrt='x'

clip_scale
    CLIP Guidance!

cutn
    Used when clip_scale > 0
    Number of times CLIP views the image for each step
    This variable only matters under these conditions:
        Extremely large images bigger than 2688 x 2688px, or
        decode_method = autoencoder
    This is because clip looks at the decoded image, which is small when decode_method=linear

cut_pow
    Used when clip_scale > 0
    Affects the size of the view window when CLIP sees the image
    When cut_pow is very small, the view window is large, and vise versa
    Only matters for extremely large images or autoencoder decode_method, see cutn

Other Conditional Settings

init_mse_scale
init_mse_image
    Guide the image to match init_mse_image pixel-by-pixel

Conditional Gradient Settings

gradient_wrt ["x", "x0_pred"]

"x0_pred"

- Applies the gradient calculation only to the denoised image at each step
- Much faster
- May require higher *_scale values
- Does not work very well with clip_scale or aesthetic_scale guidance

"x"

- Applies the gradient calculation through the unet
- Much slower
- Takes the content of the image more into consideration because the grad uses the unet

gradient_add_to ["cond", "uncond", "both"]

Gradient is added to the cfg cond or uncond or both

"cond" or "uncond" only applies if cond_uncond_sync == False

decode_method ["autoencoder","linear"]

Gradient conditioning decodes the latent at every step.

- Linear is a shortcut that quickly translates the latent to a small (default 64x64) image. linear decoding is not implemented for every model, so **if your results with conditioning look noisy, try "autoencoder".**
- Autoencoder is much slower, but higher quality. It uses the same decoding method the model already uses after the last step.

Clamping

Keeps the gradient from getting too big and making the image all washed out.
If you find that a conditioning setting isn't changing an image much no matter how high the conditioning scale, try increaseing one of these clamp numbers or using grad_threshold_type = "dynamic".

grad_threshold_type ["dynamic", "static", "mean", "schedule"]

The final gradient is clamped before it's added to the image. This is to avoid the overexposed look.

- dynamic: thresholding from the imagen paper (May 2022)
- static: simple clamping
- mean: rescales out of range values based on the mean
- schedule: uses clamp_start and clamp_stop to set a threshold value that changes linearly at each step. Like "mean" but with changing clamp_grad_threshold over time

clamp_grad_threshold

used with grad_threshold_types “static” “mean” and “dynamic”

clamp_start

used with grad_threshold_type “dynamic”
clamp_grad_threshold value at the first step, linearly changes to clamp_stop at the last step

clamp_stop

used with grad_threshold_type “dynamic”
clamp_grad_threshold value at the last step, linearly changes from clamp_start from the first step

Inject Timing

grad_inject_timing

Applies the gradient only at a limited number of steps
Interpreted differently based on the type of grad_inject_timing
if grad_inject_timing is:

- **int**: compute every inject_timing steps, eg. 2 mean apply grad every other step
- **list of floats**: compute on these decimal fraction steps (eg, [0.5, 1.0] for 50 steps would be at steps 25 and 50)
- **list of ints**: compute on these steps
- **None**: Compute on all steps

This can be used to generate an image faster, or apply conditioning to different sections of the gen. For example, overall composition is usually decided in the first 20% of steps, so if colormatch conditioning is only applied in the last half of steps it will result in practically the same image but with different colors.

Speed vs VRAM Settings

cond_uncond_sync

Syncs cond an uncond so they are generated in parallel.
Faster if True, Slower if False
False uses less vram
gradient_add_to = "cond" or "uncond" only applies if cond_uncond_sync == False