Package 'sd2R'

April 9, 2026

Type Package

Title Stable Diffusion Image Generation

Version 0.1.9

Description Provides Stable Diffusion image generation in R using the 'ggmlR' tensor library. Supports text-to-image and image-to-image generation with multiple model versions (SD 1.x, SD 2.x, 'SDXL', Flux). Implements the full inference pipeline including CLIP text encoding, 'UNet' noise removal, and 'VAE' encoding/decoding. Unified sd_generate() entry point with automatic strategy selection (direct, tiled sampling, high-resolution fix) based on output resolution and available 'VRAM'. High-resolution generation (2K, 4K+) via tiled 'VAE' decoding, tiled diffusion sampling ('MultiDiffusion'), and classic two-pass refinement (text-to-image, then upscale with image-to-image). Multi-GPU parallel generation via sd_generate_multi_gpu(). Multi-GPU model parallelism via 'device_layout' in sd_ctx(): distribute diffusion, text encoders, and 'VAE' across separate 'Vulkan' devices. Built-in profiling (sd_profile_start(), sd_profile_summary()) for per-stage timing of text encoding, sampling, and 'VAE' decode. Interactive Shiny GUI via sd_app() with non-blocking asynchronous generation (C++ std::thread), live progress bar, auto-detection of model architecture, and ETA display. Supports CPU and 'Vulkan' GPU. No 'Python' or external API dependencies required. Cross-platform: Linux, macOS, Windows.

SystemRequirements GNU make, curl or wget (for downloading vocabulary files during installation)

License MIT + file LICENSE

URL https://github.com/Zabis13/sd2R

BugReports https://github.com/Zabis13/sd2R/issues

Depends R (>= 4.1.0)

Encoding UTF-8

Imports Rcpp (>= 1.0.0), ggmlR (>= 0.5.0)

LinkingTo Rcpp, ggmlR

Suggests testthat (>= 3.0.0), callr, png, shiny, base64enc, plumber, jsonlite

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation yes

Author Yuri Baramykov [aut, cre],
Georgi Gerganov [ctb, cph] (Author of the GGML library),
leejet [ctb, cph] (Author of stable-diffusion.cpp),
stduhpf [ctb] (Core contributor to stable-diffusion.cpp),
Green-Sky [ctb] (Contributor to stable-diffusion.cpp),
wbruna [ctb] (Contributor to stable-diffusion.cpp),
akleine [ctb] (Contributor to stable-diffusion.cpp),
Martin Raiber [cph] (Copyright holder in miniz.h),
Rich Geldreich [cph] (Author of miniz.h),
RAD Game Tools [cph] (Copyright holder in miniz.h),
Valve Software [cph] (Copyright holder in miniz.h),
Alex Evans [cph] (PNG writing code in miniz.h),
Sean Barrett [cph] (Author of stb_image.h),
Jorge L Rodriguez [cph] (Author of stb_image_resize.h),
Niels Lohmann [cph] (Author of json.hpp (nlohmann/json)),
Susumu Yata [cph] (Author of darts.h (darts-clone)),
Kuba Podgorski [cph] (Author of zip.h/zip.c (kuba--/zip)),
Meta Platforms Inc. [cph] (rng_mt19937.hpp (ported from PyTorch)),
Google Inc. [cph] (Sentencepiece tokenizer code in t5.hpp)

Maintainer Yuri Baramykov <lbsbmsu@mail.ru>

Repository CRAN

Date/Publication 2026-04-09 12:00:02 UTC

Contents

`LORA_APPLY_MODE` *LoRA apply modes*

Description

LoRA apply modes

Usage

```
LORA_APPLY_MODE
```

Format

An object of class `list` of length 3.

PREDICTION *Prediction types*

Description

Prediction types

Usage

```
PREDICTION
```

Format

An object of class `list` of length 6.

RNG_TYPE *RNG types*

Description

RNG types

Usage

```
RNG_TYPE
```

Format

An object of class `list` of length 3.

SAMPLE_METHOD *Sampling methods*

Description

Sampling methods

Usage

```
SAMPLE_METHOD
```

Format

An object of class `list` of length 12.

SCHEDULER *Schedulers*

Description

Schedulers

Usage

```
SCHEDULER
```

Format

An object of class `list` of length 10.

sd_api_start *Start sd2R REST API server*

Description

Launches a plumber-based REST API for image generation. Optionally pre-loads a model at startup.

Usage

```
sd_api_start(
  model_path = NULL,
  model_type = "sd1",
  model_id = NULL,
  vae_decode_only = TRUE,
  host = "0.0.0.0",
  port = 8080L,
  api_key = NULL,
  ...
)
```

Arguments

`model_path`	Optional path to model file to load at startup
`model_type`	Model type for the pre-loaded model (default "sd1")
`model_id`	Identifier for the pre-loaded model (default: basename of model_path)
`vae_decode_only`	VAE decode only for the pre-loaded model (default TRUE)
`host`	Host to bind to (default "0.0.0.0")

port	Port to listen on (default 8080)
api_key	Optional API key string. When set, non-localhost requests must include X-API-Key or Authorization: Bearer <key> header. Default NULL (no auth).
...	Additional arguments passed to sd_ctx for the pre-loaded model

Value

Invisibly returns the plumber router object

Examples

```
## Not run:
# Start with a pre-loaded model
sd_api_start("model.safetensors", model_type = "flux", port = 8080)

# Start empty, load models via API
sd_api_start(port = 8080)

# With API key
sd_api_start("model.safetensors", api_key = "my-secret-key")

## End(Not run)
```

---

sd_api_stop *Stop sd2R REST API server*

Description

Stops the running plumber server and unloads all models.

Usage

```
sd_api_stop()
```

Value

No return value, called for side effects.

sd_app *Launch sd2R Shiny GUI*

Description

Opens an interactive Shiny application for text-to-image generation. Requires the **shiny** and **base64enc** packages.

Usage

```
sd_app(model_dir = NULL, launch.browser = TRUE, port = NULL, ...)
```

Arguments

`model_dir`	Path to folder with model files. If provided, the app scans the folder on startup and auto-assigns model roles.
`launch.browser`	Open in browser (default TRUE)
`port`	Port number (default NULL = random)
`...`	Additional arguments passed to `runApp`

Value

This function does not return; it runs the Shiny app until stopped.

Examples

```
## Not run:
sd_app()
sd_app(model_dir = "/path/to/models")

## End(Not run)
```

SD_CACHE_MODE *Cache modes*

Description

Cache modes

Usage

```
SD_CACHE_MODE
```

Format

An object of class `list` of length 6.

sd_cache_params *Create cache configuration for step caching*

Description

Constructs a list of cache parameters for fine-tuning step caching behavior. Pass the result as `cache_config` to generation functions.

Usage

```
sd_cache_params(
  mode = SD_CACHE_MODE$EASYCACHE,
  threshold = 1,
  start_percent = 0.15,
  end_percent = 0.95
)
```

Arguments

`mode`	Cache mode integer from `SD_CACHE_MODE` (default EASYCACHE)
`threshold`	Reuse threshold (default 1.0). Lower = more aggressive caching
`start_percent`	Start caching after this fraction of steps (default 0.15)
`end_percent`	Stop caching after this fraction of steps (default 0.95)

Value

Named list of cache parameters

sd_convert *Convert model to different quantization format*

Description

Convert model to different quantization format

Usage

```
sd_convert(
  input_path,
  output_path,
  output_type = SD_TYPE$F16,
  vae_path = NULL,
  tensor_type_rules = NULL
)
```

Arguments

input_path	Path to input model file
output_path	Path for output model file
output_type	Target quantization type (see SD_TYPE)
vae_path	Optional path to separate VAE model
tensor_type_rules	Optional tensor type rules string

Value

TRUE on success

---

sd_ctx *Create a Stable Diffusion context*

Description

Loads a model and creates a context for image generation.

Usage

```
sd_ctx(
  model_path = NULL,
  vae_path = NULL,
  taesd_path = NULL,
  clip_l_path = NULL,
  clip_g_path = NULL,
  t5xxl_path = NULL,
  diffusion_model_path = NULL,
  control_net_path = NULL,
  n_threads = 0L,
  wtype = SD_TYPE$COUNT,
  vae_decode_only = TRUE,
  free_params_immediately = FALSE,
  keep_clip_on_cpu = FALSE,
  keep_vae_on_cpu = FALSE,
  diffusion_flash_attn = TRUE,
  rng_type = RNG_TYPE$CUDA,
  prediction = NULL,
  lora_apply_mode = LORA_APPLY_MODE$AUTO,
  flow_shift = 0,
  model_type = "sd1",
  vram_gb = NULL,
  device_layout = "mono",
  diffusion_gpu = -1L,
```

```
  clip_gpu = -1L,
  vae_gpu = -1L,
  verbose = FALSE
)
```

Arguments

model_path	Path to the model file (safetensors, gguf, or checkpoint)
vae_path	Optional path to a separate VAE model
taesd_path	Optional path to TAESD model for preview
clip_l_path	Optional path to CLIP-L model
clip_g_path	Optional path to CLIP-G model
t5xxl_path	Optional path to T5-XXL model
diffusion_model_path	Optional path to separate diffusion model
control_net_path	Optional path to ControlNet model
n_threads	Number of CPU threads (0 = auto-detect)
wtype	Weight type for quantization (see SD_TYPE)
vae_decode_only	If TRUE, only load VAE decoder (saves memory)
free_params_immediately	Free model params after first computation. If TRUE, the context can only be used for a single generation — subsequent calls will crash. Set to TRUE only when you need to save memory and will not reuse the context. Default is FALSE.
keep_clip_on_cpu	Keep CLIP model on CPU even when using GPU
keep_vae_on_cpu	Keep VAE on CPU even when using GPU
diffusion_flash_attn	Enable flash attention for diffusion model (default TRUE). Set to FALSE if you experience issues with specific GPU drivers or backends.
rng_type	RNG type (see RNG_TYPE)
prediction	Prediction type override (see PREDICTION), NULL = auto
lora_apply_mode	LoRA application mode (see LORA_APPLY_MODE)
flow_shift	Flow shift value for Flux models
model_type	Model architecture hint: "sd1", "sd2", "sdxl", "flux", or "sd3". Used by sd_generate to determine native resolution and tile sizes. Default "sd1".
vram_gb	Override available VRAM in GB. When set, disables auto-detection and uses this value for strategy routing. Default NULL (auto-detect from Vulkan device).
device_layout	GPU layout preset for multi-GPU systems. One of:

"mono" All models on one GPU (default).

"split_encoders" Text encoders (CLIP/T5) on GPU 1, diffusion + VAE on GPU 0.

"split_vae" Text encoders + VAE on GPU 1, diffusion on GPU 0. Maximizes VRAM for diffusion.

"encoders_cpu" Text encoders on CPU, diffusion + VAE on GPU. Saves GPU memory at the cost of slower text encoding.

Ignored when diffusion_gpu, clip_gpu, or vae_gpu are explicitly set (>= 0).

diffusion_gpu Vulkan GPU device index for the diffusion model. Default -1 (use SD_VK_DEVICE env or device 0). Overrides device_layout.

clip_gpu Vulkan GPU device index for CLIP/T5 text encoders. Default -1 (same device as diffusion). Overrides device_layout.

vae_gpu Vulkan GPU device index for VAE encoder/decoder. Default -1 (same device as diffusion). Overrides device_layout.

verbose If TRUE, print model loading progress and sampling steps. Default FALSE.

Value

An external pointer to the SD context (class "sd_ctx") with attributes model_type, vae_decode_only, vram_gb, vram_total_gb, and vram_device.

Examples

```
## Not run:
ctx <- sd_ctx("model.safetensors")
imgs <- sd_txt2img(ctx, "a cat sitting on a chair")
sd_save_image(imgs[[1]], "cat.png")

## End(Not run)
```

---

sd_generate *Generate images (unified entry point)*

Description

Automatically selects the best generation strategy based on output resolution and available VRAM (set via vram_gb in sd_ctx). For txt2img, routes between direct generation, tiled sampling (MultiDiffusion), or highres fix. For img2img (when init_image is provided), routes between direct and tiled img2img.

Usage

```
sd_generate(
  ctx,
  prompt,
  negative_prompt = "",
  width = 512L,
  height = 512L,
  init_image = NULL,
  strength = 0.75,
  sample_method = SAMPLE_METHOD$EULER,
  sample_steps = 20L,
  cfg_scale = 7,
  seed = 42L,
  batch_count = 1L,
  scheduler = SCHEDULER$DISCRETE,
  clip_skip = -1L,
  eta = 0,
  hr_strength = 0.4,
  vae_mode = "auto",
  vae_tile_size = 64L,
  vae_tile_overlap = 0.25,
  cache_mode = c("off", "easy", "ucache"),
  cache_config = NULL
)
```

Arguments

ctx	SD context created by sd_ctx
prompt	Text prompt describing desired image
negative_prompt	Negative prompt (default "")
width	Image width in pixels (default 512)
height	Image height in pixels (default 512)
init_image	Optional init image for img2img. If provided, runs img2img instead of txt2img. Requires vae_decode_only = FALSE.
strength	Denoising strength for img2img (default 0.75). Ignored for txt2img.
sample_method	Sampling method (see SAMPLE_METHOD)
sample_steps	Number of sampling steps (default 20)
cfg_scale	Classifier-free guidance scale (default 7.0)
seed	Random seed (-1 for random)
batch_count	Number of images to generate (default 1)
scheduler	Scheduler type (see SCHEDULER)
clip_skip	Number of CLIP layers to skip (-1 = auto)
eta	Eta parameter for DDIM-like samplers

hr_strength	Denoising strength for highres fix refinement pass (default 0.4). Only used when auto-routing selects highres fix.
vae_mode	VAE processing mode: "normal", "tiled", or "auto" (VRAM-aware: queries free GPU memory and enables tiling only when estimated peak VAE usage exceeds available VRAM minus a 50 MB reserve). Default "auto".
vae_tile_size	Tile size for VAE tiling (default 64)
vae_tile_overlap	Overlap for VAE tiling (default 0.25)
cache_mode	Step caching mode: "off" (default), "easy" (EasyCache), or "ucache" (UCache).
cache_config	Optional fine-tuned cache config from sd_cache_params.

Details

When vram_gb is not set on the context, defaults to direct generation (equivalent to calling sd_txt2img or sd_img2img directly).

Value

List of SD images (or single image for highres fix path).

Examples

```
## Not run:
# Simple — auto-routes based on detected VRAM
ctx <- sd_ctx("model.safetensors", model_type = "sd1",
              vae_decode_only = FALSE)
imgs <- sd_generate(ctx, "a cat", width = 2048, height = 2048)

# Manual override — force 4 GB VRAM limit
ctx4 <- sd_ctx("model.safetensors", model_type = "sd1",
               vram_gb = 4, vae_decode_only = FALSE)
imgs <- sd_generate(ctx4, "a cat", width = 2048, height = 2048)

## End(Not run)
```

---

sd_generate_multi_gpu *Parallel generation across multiple GPUs*

Description

Distributes prompts across available Vulkan GPUs, running one process per GPU via callr. Each process creates its own sd_ctx and calls sd_generate. Requires the callr package.

Usage

```
sd_generate_multi_gpu(
  model_path = NULL,
  prompts,
  negative_prompt = "",
  devices = NULL,
  seeds = NULL,
  width = 512L,
  height = 512L,
  model_type = "sd1",
  vram_gb = NULL,
  vae_decode_only = TRUE,
  progress = TRUE,
  diffusion_model_path = NULL,
  vae_path = NULL,
  clip_l_path = NULL,
  t5xxl_path = NULL,
  ...
)
```

Arguments

`model_path`	Path to the model file (single-file models like SD 1.x/2.x/SDXL)
`prompts`	Character vector of prompts (one image per prompt)
`negative_prompt`	Negative prompt applied to all images (default "")
`devices`	Integer vector of Vulkan device indices (0-based). Default `NULL` auto-detects all available devices.
`seeds`	Integer vector of seeds, same length as `prompts`. Default `NULL` generates random seeds.
`width`	Image width (default 512)
`height`	Image height (default 512)
`model_type`	Model type (default "sd1")
`vram_gb`	VRAM per GPU for auto-routing (default NULL)
`vae_decode_only`	VAE decode only (default TRUE)
`progress`	Print progress messages (default TRUE)
`diffusion_model_path`	Path to diffusion model (Flux/multi-file models)
`vae_path`	Path to VAE model
`clip_l_path`	Path to CLIP-L model
`t5xxl_path`	Path to T5-XXL model
`...`	Additional arguments passed to `sd_generate`

Value

List of SD images, one per prompt, in original order.

Note

Release any existing SD context (`rm(ctx); gc()`) before calling this function. Holding a Vulkan context in the main process while subprocesses try to use the same GPU can produce corrupted (grey) images.

Examples

```
## Not run:
# Single-file model (SD 1.x/2.x/SDXL)
imgs <- sd_generate_multi_gpu(
  "model.safetensors",
  prompts = c("a cat", "a dog", "a bird", "a fish"),
  devices = 0:1
)

# Multi-file model (Flux)
imgs <- sd_generate_multi_gpu(
  diffusion_model_path = "flux1-dev-Q4_K_S.gguf",
  vae_path = "ae.safetensors",
  clip_l_path = "clip_l.safetensors",
  t5xxl_path = "t5-v1_1-xxl-encoder-Q5_K_M.gguf",
  prompts = c("a cat", "a dog"),
  model_type = "flux", devices = 0:1
)

## End(Not run)
```

---

sd_image_to_array *Convert SD image to R numeric array*

Description

Converts the raw uint8 SD image format to a [height, width, channels] numeric array with values in [0, 1] suitable for R image processing.

Usage

```
sd_image_to_array(image)
```

Arguments

`image`	SD image list (width, height, channel, data)

Value

3D numeric array [height, width, channels] in [0, 1]

`sd_img2img` *Generate images with img2img*

Description

Generate images with img2img

Usage

```
sd_img2img(
  ctx,
  prompt,
  init_image,
  negative_prompt = "",
  width = NULL,
  height = NULL,
  sample_method = SAMPLE_METHOD$EULER,
  sample_steps = 20L,
  cfg_scale = 7,
  seed = 42L,
  batch_count = 1L,
  scheduler = SCHEDULER$DISCRETE,
  clip_skip = -1L,
  strength = 0.75,
  eta = 0,
  vae_mode = "auto",
  vae_auto_threshold = 1048576L,
  vae_tile_size = 64L,
  vae_tile_overlap = 0.25,
  vae_tile_rel_x = NULL,
  vae_tile_rel_y = NULL,
  vae_tiling = NULL,
  cache_mode = c("off", "easy", "ucache"),
  cache_config = NULL
)
```

Arguments

`ctx`	SD context created by `sd_ctx`
`prompt`	Text prompt describing desired image
`init_image`	Init image in sd_image format. Use `sd_load_image` to load from file.
`negative_prompt`	Negative prompt (default "")
`width`	Image width in pixels (default 512)
`height`	Image height in pixels (default 512)

sample_method	Sampling method (see SAMPLE_METHOD)
sample_steps	Number of sampling steps (default 20)
cfg_scale	Classifier-free guidance scale (default 7.0)
seed	Random seed (-1 for random)
batch_count	Number of images to generate (default 1)
scheduler	Scheduler type (see SCHEDULER)
clip_skip	Number of CLIP layers to skip (-1 = auto)
strength	Denoising strength (0.0 = no change, 1.0 = full denoise, default 0.75)
eta	Eta parameter for DDIM-like samplers
vae_mode	VAE processing mode: "normal" (no tiling), "tiled" (always tile), or "auto" (VRAM-aware: queries free GPU memory via Vulkan and compares against estimated peak VAE usage; tiles only when VRAM is insufficient). Default "auto".
vae_auto_threshold	Pixel area fallback threshold for vae_mode = "auto" when VRAM query is unavailable (no Vulkan, CPU backend, etc.). Tiling activates when width * height exceeds this value. Default 1048576L (1024x1024 pixels).
vae_tile_size	Tile size in latent pixels for tiled VAE (default 64). Ignored when vae_tile_rel_x/vae_tile_rel_y are set.
vae_tile_overlap	Overlap ratio between tiles, 0.0-0.5 (default 0.25)
vae_tile_rel_x	Relative tile width as fraction of latent width (0-1) or number of tiles (>1). NULL = use vae_tile_size. Takes priority over vae_tile_size.
vae_tile_rel_y	Relative tile height as fraction of latent height (0-1) or number of tiles (>1). NULL = use vae_tile_size. Takes priority over vae_tile_size.
vae_tiling	**Deprecated.** Use vae_mode instead. If TRUE, equivalent to vae_mode = "tiled".
cache_mode	Step caching mode: "off" (default), "easy" (EasyCache — skips redundant denoising steps), or "ucache" (UCache). Can speed up sampling 20-40% with minor quality impact.
cache_config	Optional fine-tuned cache config from sd_cache_params. Overrides cache_mode when provided.

Value

List of SD images

sd_list_models *List registered models*

Description

Returns a data frame of all models recorded in the sd2R model registry, with a column indicating which are currently loaded in memory.

Usage

```
sd_list_models()
```

Value

Data frame with columns: id, model_type, loaded, diffusion_path

sd_load_image *Load image from file as SD image*

Description

Reads a PNG file and converts it to the SD image format (list with width, height, channel, data) suitable for img2img.

Usage

```
sd_load_image(path, channels = 3L)
```

Arguments

path	Path to image file (PNG)
channels	Number of output channels (3 for RGB, default)

Value

SD image list (width, height, channel, data as raw vector)

`sd_load_model` *Load a registered model*

Description

Loads a model by its registry id. Returns a cached context if already loaded, otherwise creates a new `sd_ctx`. Additional arguments override registry defaults.

Usage

```
sd_load_model(id, ...)
```

Arguments

`id`	Model identifier from registry
`...`	Additional arguments passed to `sd_ctx`, overriding registry defaults (e.g. `vae_decode_only = FALSE`)

Details

If loading fails due to insufficient VRAM, automatically unloads the least recently used model and retries.

Value

SD context (external pointer)

Examples

```
## Not run:
ctx <- sd_load_model("flux-dev")
imgs <- sd_txt2img(ctx, "a cat in space")

# Override defaults
ctx <- sd_load_model("flux-dev", vae_decode_only = FALSE, verbose = TRUE)

## End(Not run)
```

sd_load_pipeline *Load pipeline from JSON*

Description

Load pipeline from JSON

Usage

```
sd_load_pipeline(path)
```

Arguments

path	Path to a JSON file saved by sd_save_pipeline.

Value

An sd_pipeline object.

sd_node *Create a pipeline node*

Description

Create a pipeline node

Usage

```
sd_node(type, ...)
```

Arguments

type	Node type: "txt2img", "img2img", "upscale", or "save".
...	Parameters for the node (passed to the corresponding function).

Value

A list with class "sd_node".

sd_pipeline *Create a pipeline from nodes*

Description

Nodes are executed sequentially. The image output of each node is passed as input to the next node.

Usage

```
sd_pipeline(...)
```

Arguments

`...`	`sd_node` objects in execution order.

Value

A list with class `"sd_pipeline"`.

sd_profile_get *Get raw profile events*

Description

Returns a data frame of captured events with columns `stage`, `kind` (`"start"`/`"end"`), and `timestamp_ms`.

Value

Data frame of profile events.

sd_profile_start *Start profiling*

Description

Clears the event buffer and begins capturing stage timings from sd.cpp.

Value

No return value, called for side effects.

sd_profile_stop *Stop profiling*

Description

Stops capturing stage events. Call `sd_profile_get` to retrieve.

Value

No return value, called for side effects.

sd_profile_summary *Build a profile summary from raw events*

Description

Matches start/end events by stage and computes durations.

Usage

```
sd_profile_summary(events)
```

Arguments

`events`	Data frame from `sd_profile_get()` with columns `stage`, `kind`, `timestamp_ms`.

Value

Data frame with columns `stage`, `start_ms`, `end_ms`, `duration_ms`, `duration_s`. Has class `"sd_profile"` for pretty printing.

sd_register_model *Register a model in the sd2R model registry*

Description

Adds or updates a model entry in the sd2R model registry file. The registry lives in `tools::R_user_dir("sd2R", "config")` by default and can be overridden via the `SD2R_REGISTRY_DIR` environment variable. The directory is created only when a model is actually registered. Paths and defaults are stored for later use by `sd_load_model`.

Usage

```
sd_register_model(id, model_type, paths, defaults = list(), overwrite = FALSE)
```

Arguments

id	Unique model identifier (e.g. "flux-dev", "sd15-base")
model_type	Model architecture: "sd1", "sd2", "sdxl", "flux", "sd3"
paths	Named list of file paths. Recognized names: diffusion, model (alias for diffusion), vae, clip_l, clip_g, t5xxl, taesd, control_net.
defaults	Named list of generation defaults (optional). Recognized: steps, cfg_scale, scheduler, width, height, sample_method.
overwrite	If FALSE (default), error when id already exists

Value

Invisible model id

Examples

```
## Not run:
sd_register_model(
  id = "flux-dev",
  model_type = "flux",
  paths = list(
    diffusion = "models/flux1-dev-Q4_K_S.gguf",
    vae = "models/ae.safetensors",
    clip_l = "models/clip_l.safetensors",
    t5xxl = "models/t5xxl_fp16.safetensors"
  ),
  defaults = list(steps = 25, cfg_scale = 3.5, width = 1024, height = 1024)
)

## End(Not run)
```

---

sd_remove_model *Remove a model from the registry*

Description

Removes the model entry from the sd2R model registry and unloads it from memory if loaded.

Usage

```
sd_remove_model(id)
```

Arguments

id	Model identifier

Value

No return value, called for side effects.

sd_run_pipeline *Run a pipeline*

Description

Executes nodes sequentially. The first node must be "txt2img" (produces an image from nothing). Subsequent nodes receive the previous node's image output.

Usage

```
sd_run_pipeline(pipeline, ctx, upscaler_ctx = NULL, verbose = FALSE)
```

Arguments

pipeline	An sd_pipeline object.
ctx	A Stable Diffusion context created by sd_ctx.
upscaler_ctx	Optional upscaler context created by sd_upscale_image setup. Required if the pipeline contains an "upscale" node. Pass the result of sd_create_upscaler(path).
verbose	Logical. Print progress messages. Default FALSE.

Value

The final image (sd_image list), or the path string if the last node is "save".

sd_save_image *Save SD image to PNG file*

Description

Save SD image to PNG file

Usage

```
sd_save_image(image, path)
```

Arguments

image	SD image (list with width, height, channel, data) as returned by sd_txt2img() or sd_img2img(). Can also be a 3D numeric array [height, width, channels] with values in [0, 1].
path	Output file path (should end in .png)

Value

The file path (invisibly).

sd_save_pipeline *Save pipeline to JSON*

Description

Save pipeline to JSON

Usage

```
sd_save_pipeline(pipeline, path)
```

Arguments

pipeline	An sd_pipeline object.
path	File path (should end in .json).

Value

The file path, invisibly.

sd_scan_models *Scan a directory for models and register them*

Description

Scans for .safetensors and .gguf files, guesses component roles and model types from filenames, groups multi-file models (Flux), and registers them.

Usage

```
sd_scan_models(dir, overwrite = FALSE, recursive = FALSE)
```

Arguments

dir	Directory to scan
overwrite	If TRUE, overwrite existing entries (default FALSE)
recursive	Scan subdirectories (default FALSE)

Details

Single-file models (SD 1.5, SDXL) are registered individually. Multi-file Flux models are grouped when diffusion + supporting files (VAE, CLIP, T5) are found in the same directory.

Value

Character vector of registered model ids (invisible)

Examples

```
## Not run:
sd_scan_models("/mnt/models/")
sd_list_models()

## End(Not run)
```

`sd_system_info` *Get system information*

Description

Returns information about the stable-diffusion.cpp backend.

Usage

```
sd_system_info()
```

Value

List with system info, version, and core count

`sd_txt2img` *Generate images from text prompt*

Description

Generate images from text prompt

Usage

```
sd_txt2img(
  ctx,
  prompt,
  negative_prompt = "",
  width = 512L,
  height = 512L,
  sample_method = SAMPLE_METHOD$EULER,
  sample_steps = 20L,
  cfg_scale = 7,
  seed = 42L,
  batch_count = 1L,
  scheduler = SCHEDULER$DISCRETE,
  clip_skip = -1L,
  eta = 0,
```

```
  control_image = NULL,
  control_strength = 0.9,
  vae_mode = "auto",
  vae_auto_threshold = 1048576L,
  vae_tile_size = 64L,
  vae_tile_overlap = 0.25,
  vae_tile_rel_x = NULL,
  vae_tile_rel_y = NULL,
  vae_tiling = NULL,
  cache_mode = c("off", "easy", "ucache"),
  cache_config = NULL
)
```

Arguments

ctx	SD context created by sd_ctx
prompt	Text prompt describing desired image
negative_prompt	Negative prompt (default "")
width	Image width in pixels (default 512)
height	Image height in pixels (default 512)
sample_method	Sampling method (see SAMPLE_METHOD)
sample_steps	Number of sampling steps (default 20)
cfg_scale	Classifier-free guidance scale (default 7.0)
seed	Random seed (-1 for random)
batch_count	Number of images to generate (default 1)
scheduler	Scheduler type (see SCHEDULER)
clip_skip	Number of CLIP layers to skip (-1 = auto)
eta	Eta parameter for DDIM-like samplers
control_image	Optional control image for ControlNet (sd_image format)
control_strength	ControlNet strength (default 0.9)
vae_mode	VAE processing mode: "normal" (no tiling), "tiled" (always tile), or "auto" (VRAM-aware: queries free GPU memory via Vulkan and compares against estimated peak VAE usage; tiles only when VRAM is insufficient). Default "auto".
vae_auto_threshold	Pixel area fallback threshold for vae_mode = "auto" when VRAM query is unavailable (no Vulkan, CPU backend, etc.). Tiling activates when width * height exceeds this value. Default 1048576L (1024x1024 pixels).
vae_tile_size	Tile size in latent pixels for tiled VAE (default 64). Ignored when vae_tile_rel_x/vae_tile_rel_y are set.
vae_tile_overlap	Overlap ratio between tiles, 0.0-0.5 (default 0.25)

vae_tile_rel_x	Relative tile width as fraction of latent width (0-1) or number of tiles (>1). NULL = use vae_tile_size. Takes priority over vae_tile_size.
vae_tile_rel_y	Relative tile height as fraction of latent height (0-1) or number of tiles (>1). NULL = use vae_tile_size. Takes priority over vae_tile_size.
vae_tiling	**Deprecated.** Use vae_mode instead. If TRUE, equivalent to vae_mode = "tiled".
cache_mode	Step caching mode: "off" (default), "easy" (EasyCache — skips redundant denoising steps), or "ucache" (UCache). Can speed up sampling 20-40% with minor quality impact.
cache_config	Optional fine-tuned cache config from sd_cache_params. Overrides cache_mode when provided.

Value

List of SD images. Each image is a list with width, height, channel, and data (raw vector of RGB pixels). Use sd_save_image to save or sd_image_to_array to convert.

---

sd_txt2img_highres *High-resolution image generation via patch-based pipeline*

Description

Generates a large image by independently rendering overlapping patches at the model's native resolution, then stitching them with linear blending. An optional img2img harmonization pass can smooth seams further.

Usage

```
sd_txt2img_highres(
  ctx,
  prompt,
  negative_prompt = "",
  width = 2048L,
  height = 2048L,
  tile_size = NULL,
  overlap = 0.125,
  img2img_strength = NULL,
  sample_method = SAMPLE_METHOD$EULER,
  sample_steps = 20L,
  cfg_scale = 7,
  seed = 42L,
  scheduler = SCHEDULER$DISCRETE,
  clip_skip = -1L,
  eta = 0,
  vae_mode = "auto",
  vae_auto_threshold = 1048576L,
  vae_tile_size = 64L,
  vae_tile_overlap = 0.25
)
```

Arguments

ctx	SD context created by sd_ctx
prompt	Text prompt
negative_prompt	Negative prompt (default "")
width	Target image width in pixels
height	Target image height in pixels
tile_size	Patch size in pixels. NULL = auto-detect from model_type attribute on ctx (512 for SD1/SD2, 1024 for SDXL/Flux/SD3). Must be divisible by 8.
overlap	Overlap between patches as fraction of tile_size, 0.0-0.5 (default 0.125).
img2img_strength	If not NULL, run a final img2img pass over the stitched image at this denoising strength (e.g. 0.3) to harmonize seams. Requires vae_decode_only = FALSE in the context. Default NULL (disabled).
sample_method	Sampling method (see SAMPLE_METHOD)
sample_steps	Number of sampling steps (default 20)
cfg_scale	Classifier-free guidance scale (default 7.0)
seed	Base random seed. Each patch gets seed + patch_index. Use -1 for random.
scheduler	Scheduler type (see SCHEDULER)
clip_skip	Number of CLIP layers to skip (-1 = auto)
eta	Eta parameter for DDIM-like samplers
vae_mode	VAE tiling mode for the harmonization pass (default "auto": VRAM-aware, see sd_txt2img).
vae_auto_threshold	Pixel area fallback threshold for auto VAE tiling when VRAM query is unavailable
vae_tile_size	Tile size for VAE tiling (default 64)
vae_tile_overlap	Overlap for VAE tiling (default 0.25)

Value

SD image (list with width, height, channel, data)

Examples

```
## Not run:
ctx <- sd_ctx("sd15.safetensors", model_type = "sd1")
img <- sd_txt2img_highres(ctx, "a panoramic mountain landscape",
                          width = 2048, height = 1024)
sd_save_image(img, "panorama.png")

## End(Not run)
```

`sd_txt2img_tiled` *Tiled diffusion sampling (MultiDiffusion)*

Description

Generates images at any resolution using tiled sampling: at each denoising step the latent is split into overlapping tiles, each tile is denoised independently by the UNet, and results are merged with Gaussian weighting. VRAM usage is bounded by tile size, not output resolution.

Usage

```
sd_txt2img_tiled(
  ctx,
  prompt,
  negative_prompt = "",
  width = 2048L,
  height = 2048L,
  sample_tile_size = NULL,
  sample_tile_overlap = 0.25,
  sample_method = SAMPLE_METHOD$EULER,
  sample_steps = 20L,
  cfg_scale = 7,
  seed = 42L,
  batch_count = 1L,
  scheduler = SCHEDULER$DISCRETE,
  clip_skip = -1L,
  eta = 0,
  vae_mode = "auto",
  vae_auto_threshold = 1048576L,
  vae_tile_size = 64L,
  vae_tile_overlap = 0.25,
  vae_tile_rel_x = NULL,
  vae_tile_rel_y = NULL,
  cache_mode = c("off", "easy", "ucache"),
  cache_config = NULL
)
```

Arguments

`ctx`	SD context created by `sd_ctx`
`prompt`	Text prompt describing desired image
`negative_prompt`	Negative prompt (default "")
`width`	Target image width in pixels (can exceed model native resolution)
`height`	Target image height in pixels

`sample_tile_size`
: Tile size in latent pixels (default `NULL` = auto from `model_type`: 64 for SD1/SD2, 128 for SDXL/Flux/SD3). One latent pixel = `vae_scale_factor` image pixels (typically 8).

`sample_tile_overlap`
: Overlap between tiles as fraction of tile size, 0.0-0.5 (default 0.25).

`sample_method`
: Sampling method (see `SAMPLE_METHOD`)

`sample_steps`
: Number of sampling steps (default 20)

`cfg_scale`
: Classifier-free guidance scale (default 7.0)

`seed`
: Random seed (-1 for random)

`batch_count`
: Number of images to generate (default 1)

`scheduler`
: Scheduler type (see `SCHEDULER`)

`clip_skip`
: Number of CLIP layers to skip (-1 = auto)

`eta`
: Eta parameter for DDIM-like samplers

`vae_mode`
: VAE processing mode: `"normal"` (no tiling), `"tiled"` (always tile), or `"auto"` (VRAM-aware: queries free GPU memory via Vulkan and compares against estimated peak VAE usage; tiles only when VRAM is insufficient). Default `"auto"`.

`vae_auto_threshold`
: Pixel area fallback threshold for `vae_mode = "auto"` when VRAM query is unavailable (no Vulkan, CPU backend, etc.). Tiling activates when `width * height` exceeds this value. Default `1048576L` (1024x1024 pixels).

`vae_tile_size`
: Tile size in latent pixels for tiled VAE (default 64). Ignored when `vae_tile_rel_x`/`vae_tile_rel_y` are set.

`vae_tile_overlap`
: Overlap ratio between tiles, 0.0-0.5 (default 0.25)

`vae_tile_rel_x`
: Relative tile width as fraction of latent width (0-1) or number of tiles (>1). NULL = use `vae_tile_size`. Takes priority over `vae_tile_size`.

`vae_tile_rel_y`
: Relative tile height as fraction of latent height (0-1) or number of tiles (>1). NULL = use `vae_tile_size`. Takes priority over `vae_tile_size`.

`cache_mode`
: Step caching mode: `"off"` (default), `"easy"` (EasyCache — skips redundant denoising steps), or `"ucache"` (UCache). Can speed up sampling 20-40% with minor quality impact.

`cache_config`
: Optional fine-tuned cache config from `sd_cache_params`. Overrides `cache_mode` when provided.

Details

Requires tiled VAE (enabled automatically via `vae_mode = "auto"`).

Value

List of SD images

Examples

```
## Not run:
ctx <- sd_ctx("sd15.safetensors", model_type = "sd1")
imgs <- sd_txt2img_tiled(ctx, "a vast mountain landscape",
                         width = 2048, height = 1024)
sd_save_image(imgs[[1]], "landscape.png")

## End(Not run)
```

SD_TYPE *Weight types (ggml quantization types)*

Description

Weight types (ggml quantization types)

Usage

```
SD_TYPE
```

Format

An object of class `list` of length 15.

---

sd_unload_all *Unload all models from memory*

Description

Removes all cached contexts. Registry is preserved.

Usage

```
sd_unload_all()
```

Value

No return value, called for side effects.

sd_unload_model *Unload a model from memory*

Description

Removes the cached context for the given model id. The model remains in the registry and can be reloaded with sd_load_model.

Usage

```
sd_unload_model(id)
```

Arguments

id	Model identifier

Value

No return value, called for side effects.

sd_upscale_image *Upscale an image using ESRGAN*

Description

Upscale an image using ESRGAN

Usage

```
sd_upscale_image(esrgan_path, image, upscale_factor = 4L, n_threads = 0L)
```

Arguments

esrgan_path	Path to ESRGAN model file
image	SD image to upscale (list with width, height, channel, data)
upscale_factor	Upscale factor (default 4)
n_threads	Number of CPU threads (0 = auto-detect)

Value

Upscaled SD image

`sd_vulkan_device_count` *Get number of Vulkan GPU devices*

Description

Returns the number of Vulkan-capable GPU devices available on the system. Useful for deciding whether to use `sd_generate_multi_gpu`.

Usage

```
sd_vulkan_device_count()
```

Value

Integer, number of Vulkan devices (0 if Vulkan is not available)

Index