📘 Project 1: RectifiedHR – High-Resolution Diffusion via Energy Profiling and Scheduling

---

🧠 1. Background & Motivation

Diffusion models like **Stable Diffusion (SD)** have shown impressive capabilities in high-resolution image synthesis. However, when generating images at **1024×1024 or higher**, users often observe:

- **Blurry outputs**, especially in details
- **Over-saturation** or washed-out colors
- **Artifacts due to classifier-free guidance instability**

Recent work like **RectifiedFlow**, **HiResFix**, and **RectifiedHR** show that even in **pre-trained models**, tweaking **the noise schedule**, **guidance strength**, or **latent sampling energy** can significantly improve quality — **without retraining**.

🎯 2. Goal

To analyze and improve **high-resolution image quality** generated by latent diffusion (SD 1.5 or SDXL) via:

- Measuring **latent energy evolution** across denoising steps
- Proposing a **rectified guidance schedule** (dynamic CFG)
- Exploring **noise refresh** and **intermediate clipping**
- Producing visually and quantitatively better high-res images

---

📦 3. Deliverables

Item	Description
📁 GitHub Repo	With `generate.py`, `metrics.py`, `plot_energy.py`, and sample runs
📄 Paper Draft	4–6 page LaTeX paper with analysis, visuals, and comparisons
📊 Visual Results	Before/after grids, energy trend plots, latent heatmaps
📈 Metrics	CLIP similarity, MS-SSIM, optional FID (if resources allow)

🧱 4. Methodology

🧪 Step 1: Baseline Generation

- Generate high-resolution images (`768×768`, `1024×1024`) using:
    - **StableDiffusionPipeline** from `diffusers`
    - CFG scales: [3, 5, 7, 10]
    - Samplers: DDIM, Euler A
- Save intermediate latents (`x_t`) during sampling

📊 Step 2: Latent Energy Profiling

For each timestep `t`, compute:

python
CopyEdit
```
E_t = torch.norm(latents[t])**2 / latent.shape[1:].numel()
```

-
- Plot energy curves:

- With different CFG values
- For SD 1.5 vs SDXL
- With and without `noise_refresh` (reset `x_t` to x_0 + noise halfway)

🧪 Step 3: Rectified Guidance Scheduling

- Implement **adaptive CFG** strategy:
  - Lower CFG at early steps, increase near end (or vice versa)
  - Try smooth cosine ramp or linear increasing schedule
- Observe if:
  - High-frequency detail improves
  - Latent energy stabilizes
  - Visual quality improves

🧪 Step 4: Noise Refresh + Clipping

At step `t = T // 2`, reset latent:

python
CopyEdit
```
latents[t] = predicted_x0 + torch.randn_like(predicted_x0) * sqrt(1 -
alpha_bar[t])
```

-
- Optional: clip `x_t` norm if it exceeds a threshold

📊 Step 5: Evaluation

- For each condition (baseline, rectified, adaptive CFG), evaluate:
    - CLIP score similarity to original prompt
    - MS-SSIM for image sharpness
    - Latent energy trajectory plots
    - Side-by-side image comparisons

🛠️ 5. Tools & Frameworks

Tool	Use
🤗 `diffusers`	For SD model & samplers
`matplotlib`, `seaborn`	For plotting energy trends
`CLIP` or `BLIP`	For image-text similarity
`numpy`, `scipy`	Signal smoothing, energy calc
GPU optional	Will run on MPS or Colab if needed

📅 6. Timeline (7–8 days)

Day	Task
1	Set up repo, generate baseline images, save latents
2	Plot latent energy trends across timesteps
3	Implement adaptive CFG scheduler
4	Implement noise refresh + clipping
5	Evaluate visual quality + metrics
6	Write paper (intro, method, results)

7	Final visuals + arXiv prep
8 (buffer)	Review, polish, post

📄 7. Paper Structure

Section	Description
Abstract	What you did + why it matters + result summary
1. Introduction	Problem: SD fails at high-res, our hypothesis
2. Related Work	RectifiedFlow, HiResFix, DDIM, CFG
3. Method	Energy profiling, adaptive scheduling
4. Experiments	Images, graphs, metrics
5. Discussion	When it helps, when it doesn’t
6. Conclusion	Summary + future work (multi-resolution fusion?)