Guía de Parámetros de KSampler

El KSampler es vuestra varita mágica en el mundo de la generación de imágenes con IA (como Stable Diffusion en ComfyUI). Entender sus parámetros os permitirá controlar con precisión cómo la IA transforma una idea (vuestros prompts) y un lienzo de ruido digital en la imagen que tenéis en mente. A continuación, se detallan los parámetros más comunes que encontrarán en un KSampler:

Entradas Principales:

1. **model (Modelo):**
    - **Qué es:** Es el modelo de difusión principal que se utilizará para generar la imagen. Contiene el "conocimiento" de cómo transformar el ruido en imágenes coherentes.
    - **Qué controla:** El estilo general, la calidad y el tipo de contenido que se puede generar. Diferentes modelos están entrenados en diferentes conjuntos de datos y producen resultados distintos.
2. **positive (Positivo):**
    - **Qué es:** Aquí se introduce el "prompt" positivo. Es una descripción textual de lo que **quieres** ver en la imagen generada.
    - **Qué controla:** Los elementos, sujetos, estilos, colores y la composición general que deseas en la imagen. Cuanto más descriptivo seas, más probable es que la IA genere lo que tienes en mente.
3. **negative (Negativo):**
    - **Qué es:** Aquí se introduce el "prompt" negativo. Es una descripción textual de lo que **NO quieres** ver en la imagen.
    - **Qué controla:** Ayuda a refinar la imagen eliminando artefactos no deseados, estilos, colores o elementos. Por ejemplo, "manos feas, borroso, mala calidad".
4. **latent_image (Imagen Latente):**
    - **Qué es:** Es la representación de la imagen en el "espacio latente" que sirve como punto de partida para el KSampler.
        - Para **txt2img** (texto a imagen): Normalmente se conecta a un nodo "Empty Latent Image" que define las dimensiones de la imagen a generar.
        - Para **img2img** (imagen a imagen) o **inpainting** (rellenar partes de una imagen): Se conecta a un VAE Encode que convierte una imagen existente a su forma latente.
    - **Qué controla:** Las dimensiones de la imagen de salida (si se parte de un latente vacío) o la imagen base que se va a modificar (si se parte de una imagen existente).

Parámetros de Configuración:

5. **seed (Semilla):**
    - **Qué es:** Un número que inicializa el generador de ruido aleatorio.
    - **Qué controla:** La aleatoriedad inicial. Si usas la misma semilla con los mismos parámetros y el mismo modelo, obtendrás exactamente la misma imagen. Cambiar la semilla, incluso ligeramente, producirá una imagen diferente. Es útil para reproducir resultados o explorar variaciones.
6. **control_after_generate (Control después de generar):**
    - **Qué es:** Define qué sucede con el valor de la semilla después de que se genera una imagen.
    - **Qué controla:**
        - randomize: Se elige una nueva semilla aleatoria para la siguiente generación.
        - increment: La semilla aumenta en 1 para la siguiente generación.
        - decrement: La semilla disminuye en 1 para la siguiente generación.
        - fixed: La semilla permanece igual para la siguiente generación.
    - **Utilidad:** randomize es bueno para explorar muchas ideas diferentes. increment o fixed son útiles si te gusta un resultado y quieres hacer pequeñas variaciones o generar una serie.
7. **steps (Pasos):**
    - **Qué es:** El número de pasos de muestreo (o "denoising") que el sampler realizará.
    - **Qué controla:** La cantidad de "refinamiento" que se aplica a la imagen.
        - **Pocos pasos (ej. 10-15):** Generación más rápida, pero la imagen puede parecer inacabada o con menos detalles.
        - **Pasos moderados (ej. 20-30):** Generalmente un buen equilibrio entre calidad y tiempo de generación para muchos samplers.
        - **Muchos pasos (ej. 50+):** Puede añadir más detalles finos o coherencia, pero también aumenta significativamente el tiempo de generación y puede no siempre mejorar el resultado (a veces puede "sobrecocinar" la imagen).
    - Nota: El número óptimo de pasos puede depender del sampler_name elegido. Si bien 20-30 pasos es un buen punto de partida general, algunos samplers modernos (especialmente los de la familia DPM++) pueden lograr resultados excelentes y detallados con menos pasos (incluso 15-25). Experimentar es clave, ya que demasiados pasos con ciertos samplers no solo aumentan el tiempo, sino que podrían no mejorar la imagen o incluso introducir artefactos no deseados.

8. **cfg (Classifier Free Guidance scale / Escala de Guía Libre de Clasificador):**
    - **Qué es:** Un valor que controla qué tan estrictamente el modelo debe seguir el prompt positivo. Imagina que es como un "volumen de obediencia" al prompt. Un valor más alto le dice a la IA: "¡Sigue mis instrucciones al pie de la letra!", mientras que uno más bajo le da más espacio para la "interpretación artística" y la creatividad.
    - **Qué controla:** La fidelidad al prompt.
        - **Valores bajos (ej. 1-5):** El modelo tiene más libertad creativa, puede desviarse más del prompt.
        - **Valores medios (ej. 6-10):** Un buen equilibrio entre seguir el prompt y permitir cierta creatividad. Un valor común es 7 u 8.
        - **Valores altos (ej. 12-20+):** El modelo se adhiere muy estrictamente al prompt, lo que a veces puede llevar a imágenes "quemadas" o con artefactos si es demasiado alto.
    - **Recomendación:** Experimentar con valores entre 5 y 12 suele ser un buen punto de partida.
9. **sampler_name (Nombre del Sampler):**
    - **Qué es:** El algoritmo específico que se utiliza para el proceso de muestreo (denoising) en cada paso.
    - **Qué controla:** El "camino" que toma el modelo para pasar del ruido a una imagen clara. Diferentes samplers pueden producir resultados con diferentes características (ej. más nítidos, más suaves, más rápidos, mejores con ciertos tipos de imágenes).
    - **Ejemplos comunes:** euler, euler_ancestral (o euler_a), dpmpp_2m_karras, dpmpp_sde_gpu, lms, ddim.
    - **Nota:** No hay un "mejor" sampler universal; depende de la preferencia personal, el modelo y el tipo de imagen deseada. dpmpp_3m_sde_gpu (como el de la imagen) es un sampler moderno y popular que suele dar buenos resultados.
10. **scheduler (Programador):**
    - **Qué es:** Trabaja en conjunto con el sampler_name. Define cómo se distribuye y reduce el nivel de "ruido" a lo largo de los steps. Algunos samplers funcionan de manera óptima o están diseñados para usarse con programadores específicos. Por ejemplo, el programador "karras" suele mejorar la calidad de detalle con samplers de la familia DPM++ porque ajusta la reducción de ruido de una manera que complementa cómo operan estos samplers.
    - **Qué controla:** La progresión del proceso de denoising. Diferentes programadores pueden afectar la convergencia de la imagen y cómo se ven

los detalles en diferentes etapas.
    - **Ejemplos comunes:** normal, karras, exponential, sgm_uniform.
    - **Nota:** El programador karras (como el de la imagen) es muy popular y a menudo se recomienda para muchos samplers DPM++ porque puede ayudar a producir imágenes más detalladas y estables, especialmente con menos pasos.
11. **denoise (Reducción de Ruido / Denoise Strength):**
    - **Qué es:** Un valor (generalmente entre 0.0 y 1.0) que se usa principalmente en flujos de trabajo de **img2img** o **inpainting**.
    - **Qué controla:** Cuánto de la latent_image original se va a modificar o "ignorar" para dar paso a la nueva generación basada en los prompts.
        - 1.0: En la generación de texto a imagen (txt2img), donde se parte de una imagen latente vacía o puro ruido, este valor asegura que la IA genere una imagen completamente nueva basándose únicamente en tus prompts. Es el ajuste estándar y recomendado para txt2img, ya que no hay una imagen previa que preservar o modificar parcialmente.
        - **Valores más bajos (ej. 0.5 - 0.75) en img2img:** Conservan más de la estructura y contenido de la imagen original, aplicando los cambios del prompt de forma más sutil.
        - **Valores más altos (ej. 0.8 - 0.95) en img2img:** Permiten cambios más drásticos en la imagen original, acercándose más a una generación desde cero pero aún influenciada por la composición inicial.

Conclusión:

Entender y experimentar con estos parámetros es clave para dominar la generación de imágenes con IA. No hay configuraciones "mágicas" que funcionen para todo. La mejor manera de aprender es probar diferentes combinaciones, observar los resultados y tomar notas sobre cómo cada parámetro afecta la imagen final.