El proyecto **SarcasmDiffusion (Generador de Memes Sarcásticos)** es una idea espectacular para tu portafolio — mezcla **visión, NLP y generación multimodal**, algo que pocos proyectos personales exploran.  
Vamos paso a paso: te explico **el enfoque conceptual, el stack técnico y la arquitectura general** que usaremos 👇

**🧩 1. Enfoque general del proyecto**

**🎯 Objetivo**

Generar memes *originales y coherentes* (imagen + texto) que transmitan **sarcasmo o ironía**, condicionados por un texto de entrada (prompt).

**Ejemplo de entrada:**

“When your boss says ‘it’s just a small project’”

**Salida esperada:**  
Una imagen tipo meme con una situación visual sarcástica y un caption humorístico generado automáticamente.

**🧠 2. Etapas del pipeline**

**Fase A — Preprocesamiento del dataset**

Usaremos el **Hateful Memes Dataset (Facebook AI)** que contiene:

* Imagen del meme (image\_path)
* Texto embebido en el meme (text)
* Etiquetas (label → 0: no odio, 1: odio)

**Pasos:**

1. Extraer el texto de los memes (OCR con easyocr o pytesseract si no está en los metadatos).
2. Clasificar el tono del texto (sarcasmo, ironía, humor negro) usando un modelo preentrenado de emociones (GoEmotions o RoBERTa-sarcasm).
3. Guardar pares:
4. text,emotion,image\_path
5. "When you fail an exam but smile anyway","sarcasm","img\_0453.png"

**Fase B — Modelo de texto a imagen (Diffusion Model)**

Usaremos un modelo de **Text-to-Image Diffusion** preentrenado y lo ajustaremos (*fine-tuning ligero o LoRA*) al estilo de memes.

**Opciones base:**

* 🧩 Stable Diffusion v1.5 (versátil y rápido)
* 🧠 Stable Diffusion XL (mejor coherencia entre texto e imagen)
* ⚡ Kandinsky 2.2 (buena comprensión semántica)
* 💬 PixArt-α o FLUX (más recientes, más eficientes)

**Estrategia:**

1. Entrenar un *LoRA* (Low-Rank Adapter) con tus imágenes del dataset (centrado en estilo “meme”).
2. Fine-tuning del *text encoder* para captar el tono sarcástico de los captions.
3. Añadir capa de texto (caption overlay) en post-procesamiento automático.

**Fase C — Post-procesamiento (Overlay Generator)**

Una vez generada la imagen, añadiremos el texto al estilo “meme”:

* Tipografía: **Impact**, color blanco, borde negro.
* Posición automática: arriba o abajo según densidad visual.
* Uso de **Pillow** o **OpenCV** para colocar el texto con sombreado.

**Fase D — Interfaz (App Visual)**

Se implementará una app en **Streamlit o Gradio**, con un flujo así:

1️⃣ Usuario escribe un *prompt* (ej. “When your computer crashes during training”).  
2️⃣ Se genera una imagen con el modelo Diffusion.  
3️⃣ El sistema añade el caption sarcástico.  
4️⃣ Se muestra el resultado + botón “Regenerar con otro tono”.

**⚙️ 3. Stack tecnológico completo**

| **Componente** | **Tecnología** | **Propósito** |
| --- | --- | --- |
| **Preprocesamiento** | pandas, easyocr, transformers, matplotlib | Limpieza y extracción de texto/emociones |
| **Modelo base** | diffusers (HuggingFace), transformers | Stable Diffusion + LoRA |
| **Fine-tuning** | accelerate, peft, xformers, bitsandbytes | Ajuste eficiente del modelo |
| **Caption overlay** | Pillow, opencv-python, numpy | Generar texto visual estilo meme |
| **Interfaz visual** | Streamlit o Gradio | UI de generación en tiempo real |
| **Dataset** | Hateful Memes Dataset | Datos multimodales reales |
| **Opcional** | WandB o TensorBoard | Tracking de fine-tuning |

**🧱 4. Estructura recomendada del repositorio**

SarcasmDiffusion/

├── data/

│ ├── hateful\_memes/

│ └── processed/

├── models/

│ ├── lora\_weights/

│ └── text\_encoder/

├── src/

│ ├── preprocess.py

│ ├── train\_lora.py

│ ├── generate\_meme.py

│ ├── overlay\_text.py

│ └── app\_streamlit.py

├── notebooks/

│ └── exploration.ipynb

├── requirements.txt

└── README.md

**📈 5. Métricas y evaluación**

| **Tipo** | **Métrica** | **Descripción** |
| --- | --- | --- |
| **Visual** | CLIP similarity | Qué tan coherente es la imagen con el texto |
| **Textual** | BLEU / ROUGE | Coherencia entre caption original y generado |
| **Humana** | Survey Score (1–5) | Cuánto sarcasmo percibe el usuario |
| **Eficiencia** | Latencia de generación | Tiempo promedio por meme |

**🧠 6. Ideas de expansión futura**

* Entrenar un *classifier de sarcasmo visual-textual* para refinar prompts.
* Incluir “modos” (irónico, auto-sarcástico, humor oscuro).
* Usar embeddings de CLIP para sugerir imágenes más coherentes con el texto.
* Añadir *re-ranking* con un modelo evaluador de humor (LLM).