UNIVERSIDAD MAYOR, REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA

Facultad de Ciencias de la Tecnología

OniDraw

Sistema de mejora de dibujos a un estilo anime

ESTUDIANTE: Polo Orellana Brayan Simón

CARRERA: Ingeniería en Ciencias de la computación

MATERIA: Inteligencia Artificial 2

SIGLA: SIS-421

Resumen Ejecutivo

Drawnime es una plataforma web innovadora que utiliza inteligencia artificial para transformar dibujos o sketches simples en ilustraciones de alta calidad con estilo anime. El sistema emplea el modelo **Stable Diffusion V1-5** fine-tuned mediante **LoRA (Low-Rank Adaptation)**, entrenado específicamente con un dataset de 5,000 pares de imágenes sketches→anime durante 19 horas.

La solución aborda el desafío técnico de la generación de imágenes personalizadas, permitiendo a artistas y entusiastas del anime convertir sus dibujos básicos en ilustraciones profesionales mediante un proceso optimizado que balancea calidad artística con eficiencia computacional, siendo accesible incluso en hardware con 6GB de VRAM.

1. Introducción

1.1. Contexto y Problemática

El proceso tradicional de convertir sketches o dibujos realizados a mano en ilustraciones anime terminadas requiere habilidades artísticas avanzadas, tiempo considerable y experiencia en técnicas de digital art. Para artistas novatos, aficionados o profesionales que buscan agilizar su workflow, este proceso puede representar una barrera significativa.

La inteligencia artificial generativa, particularmente los modelos de difusión, ha demostrado capacidades revolucionarias en la generación de contenido visual. Sin embargo, los modelos generalistas often carecen de la especialización necesaria para aplicaciones específicas como la transformación sketch→anime, donde se requiere preservar la composición original mientras se aplica un estilo artístico coherente.

1.2. Justificacion

La creación de un sistema especializado en la transformación sketch→anime responde a varias necesidades del mercado:

- **Democratización del arte**: Hacer accesible la creación de ilustraciones anime de calidad.
- **Aceleración de workflows**: Reducir el tiempo de producción para artistas profesionales
- **Educación artística**: Proporcionar una herramienta de aprendizaje para principiantes
- **Innovación tecnológica**: Avanzar en aplicaciones especializadas de IA generativa

1.3. Alcance

Este proyecto desarrolla una solución full-stack que incluye:

- Frontend web interactivo para dibujo de sketches
- Backend optimizado para inferencia de IA
- Modelo especializado Stable Diffusion + LoRA
- Sistema de gestión de recursos para hardware limitado

2. Objetivos

2.1. Objetivo General

Desarrollar e implementar un sistema web integral que utilice un modelo de inteligencia artificial generativa para transformar sketches dibujados por usuarios en ilustraciones de estilo anime, mediante la fine-tuning especializada de Stable Diffusion con técnica LoRA.

2.2. Objetivos específicos

- Recopilar y preparar un dataset especializado de 5,000 pares de imágenes sketches→anime para entrenamiento supervisado.
- Implementar y entrenar un modelo Stable Diffusion V1-5 fine-tuned con LoRA específicamente para la transformación sketch→anime.
- Desarrollar una arquitectura backend optimizada en Flask que gestione eficientemente los recursos de GPU durante la inferencia del modelo.
- Crear una interfaz frontend en Angular que permita el dibujo intuitivo de sketches y la visualización de resultados en tiempo real.
- Optimizar el pipeline completo para funcionamiento en hardware con 6GB de VRAM mediante técnicas de gestión de memoria y carga diferida de modelos.
- Validar la calidad de las generaciones mediante métricas cuantitativas y evaluación cualitativa con usuarios reales.

3. Estado del arte

3.1. Modelos de Difusión para Generación de Imágenes

Los modelos de difusión son modelos generativos que se utilizan principalmente para la generación de imágenes y otras tareas de visión artificial. Las redes neuronales basadas en difusión se entrenan mediante deep learning para "difundir" progresivamente muestras con ruido aleatorio y, a continuación, invertir ese proceso de difusión para generar imágenes de alta calidad.

Los modelos de difusión se encuentran entre las arquitecturas de redes neuronales que están a la vanguardia de la IA generativa, sobre todo representados por los populares modelos de conversión de texto a imagen, como **Stable Diffusion** de **Stability AI, DALL-E de OpenAI** (empieza por DALL-E-2), Midjourney e Imagen de Google.

La intuición detrás de los modelos de difusión se inspira en la física, tratando los píxeles como las moléculas de una gota de tinta que se extiende en un vaso de agua a lo largo del tiempo.

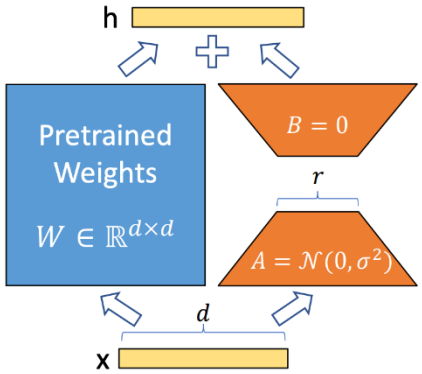
Del mismo modo que el movimiento aleatorio de las moléculas de tinta acaba por dispersarlas uniformemente en el cristal, la introducción aleatoria de ruido en una imagen acaba dando lugar a lo que parece estática televisiva. Al modelar ese proceso de difusión y luego aprender a invertirlo de alguna manera, un modelo de inteligencia artificial puede generar nuevas imágenes simplemente "eliminando" muestras de ruido aleatorio.

Aquí es donde los modelos de difusión muestran su poder. La idea es simplemente hacer lo opuesto a la difusión:

1. El modelo observa la imagen ruidosa y piensa en cómo aclararla un poco.
2. Repite este paso hasta que aparezca una imagen reconocible.

3.2. Fine-tuning Eficiente con LoRA

El ajuste clásico de modelos de lenguaje grandes generalmente cambia la mayoría de los pesos de los modelos, lo que requiere muchos recursos. El ajuste basado en LoRA congela los pesos originales y solo entrena una pequeña cantidad de parámetros, lo que hace que el entrenamiento sea mucho más eficiente.



3.2.1. PEFT

PEFT, o ajuste fino eficiente de parámetros (PEFT), es una biblioteca para adaptar de manera eficiente modelos de lenguaje (PLM) previamente entrenados a varias aplicaciones posteriores sin ajustar todos los parámetros del modelo. Los métodos PEFT solo ajustan una pequeña cantidad de parámetros (adicionales) del modelo, lo que reduce significativamente los costos computacionales y de almacenamiento porque ajustar los PLM a gran escala es prohibitivamente costoso. Las técnicas PEFT de última generación recientes logran un rendimiento comparable al del ajuste completo.

3.2.2. LoRA

LoRA Se introdujo a finales de 2021 en el documento LoRA: Adaptación de bajo rango de modelos de lenguaje grandes.

Para que el ajuste sea más eficiente, el enfoque de LoRA es representar las actualizaciones de peso con dos matrices más pequeñas (llamadas matrices de actualización) mediante descomposición de bajo rango. Estas nuevas matrices se pueden entrenar para adaptarse a los nuevos datos manteniendo bajo el número total de cambios. La matriz de peso original permanece congelada y no recibe más ajustes. Para producir los resultados finales, se combinan tanto los pesos originales como los adaptados.

4. Herramientas y Tecnologías

4.1. Modelo

4.1.1. Pythorch

PyTorch es un framework de código abierto de Python utilizado principalmente para el aprendizaje profundo y la investigación en inteligencia artificial. Permite realizar cálculos numéricos de alto rendimiento mediante la manipulación de tensores (matrices

multidimensionales) y ofrece la aceleración de estos cálculos mediante el uso de GPUs, lo que lo hace muy rápido.

```
import torch
import torch.nn as nn

# Para el entrenamiento del modelo
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Durante la inferencia
with torch.no_grad():
    output = model(input_tensor)

# Gestión de memoria
torch.cuda.empty_cache()
```

4.1.2. Diffusers

Diffusers es una biblioteca de modelos de difusión preentrenados de última generación para generar videos, imágenes y audio.

La biblioteca gira en torno a DiffusionPipeline:

- Inferencia fácil con solo unas pocas líneas de código
- Flexibilidad para mezclar y combinar componentes de canalización (modelos, programadores)
- carga y uso de adaptadores como LoRA

Los difusores también vienen con optimizaciones, como descarga y cuantificación, para garantizar que incluso los modelos más grandes sean accesibles en dispositivos con memoria limitada. Si la memoria no es un problema, los difusores admiten torch.compile para aumentar la velocidad de inferencia.

```
from diffusers import StableDiffusionImg2ImgPipeline
from diffusers import DDPMScheduler, AutoencoderKL, UNet2DConditionModel

# Configuración del pipeline
pipe = StableDiffusionImg2ImgPipeline.from_pretrained(
    "runwayml/stable-diffusion-v1-5",
    torch_dtype=torch.float16,
    safety_checker=None
)
```

```
# Generación de imágenes
result = pipe(
    prompt="Anime style, masterpiece and detailed",
    image=sketch_image,
    strength=0.75,
    num_inference_steps=30
)
```

4.1.3. Hugging Face

Hugging Face es una empresa que mantiene una enorme comunidad de código abierto del mismo nombre que construye herramientas, modelos de aprendizaje automático y plataformas para trabajar con inteligencia artificial, con un enfoque en la ciencia de datos, el aprendizaje automático y el procesamiento del lenguaje natural (NLP). Hugging Face se destaca por su biblioteca de transformadores de NLP y una plataforma que permite a los usuarios compartir modelos y conjuntos de datos.

```
model_id = "runwayml/stable-diffusion-v1-5"
batch_size = 2  # Reducido por la memoria de 6GB
image_size = 512
num_epochs = 3
learning_rate = 1e-4

# Cargar tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_id, subfolder="tokenizer")

# Cargar modelos
vae = AutoencoderKL.from_pretrained(model_id, subfolder="vae")
unet = UNet2DConditionModel.from_pretrained(model_id, subfolder="unet")
text_encoder = CLIPTextModel.from_pretrained(model_id,
subfolder="text_encoder")
scheduler = DDPMScheduler.from_pretrained(model_id, subfolder="scheduler")
```

4.1.4. LoRA

Es una técnica de entrenamiento popular y ligera que reduce significativamente el número de parámetros entrenables. Funciona insertando un número menor de nuevos pesos en el modelo y solo estos se entrenan. Esto hace que el entrenamiento con LoRA sea mucho más rápido, eficiente en memoria y produce pesos de modelo más pequeños (unos pocos cientos de MB), que son más fáciles de almacenar y

compartir. LoRA también se puede combinar con otras técnicas de entrenamiento como DreamBooth para acelerar el entrenamiento.

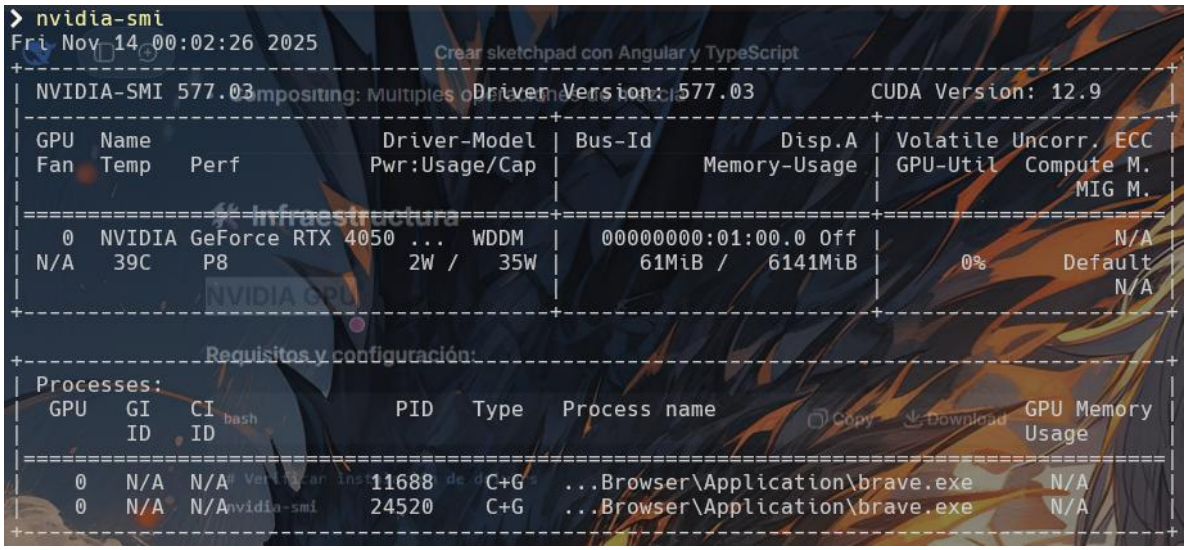
```python
from peft import LoraConfig, get_peft_model

# Configuración de LoRA
lora_config = LoraConfig(
    r=16,  # Rank de la adaptación
    lora_alpha=32,
    target_modules=["to_k", "to_q", "to_v", "to_out.0"],
    lora_dropout=0.1,
)

# Aplicación al modelo
model = get_peft_model(unet, lora_config)

# Entrenamiento solo de parámetros LoRA
optimizer = torch.optim.AdamW(model.parameters(), lr=1e-4)
```

Ventajas:

- Eficiencia: Solo 1-2% de parámetros entrenables
- Rapidez: Entrenamiento más rápido
- Portabilidad: Archivos de adaptación pequeños (~10MB)
- Flexibilidad: Múltiples adaptaciones para un mismo modelo base

4.1.5. NIVIDIA GPU

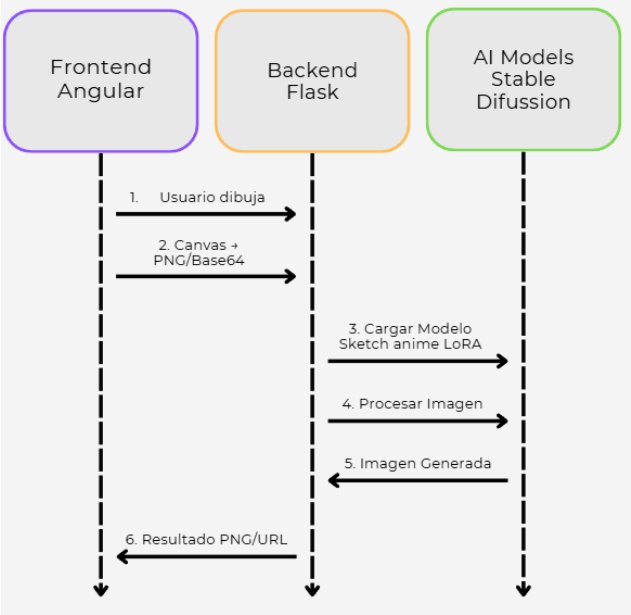
Para el entrenamiento del modelo se usó ***CUDA13***:



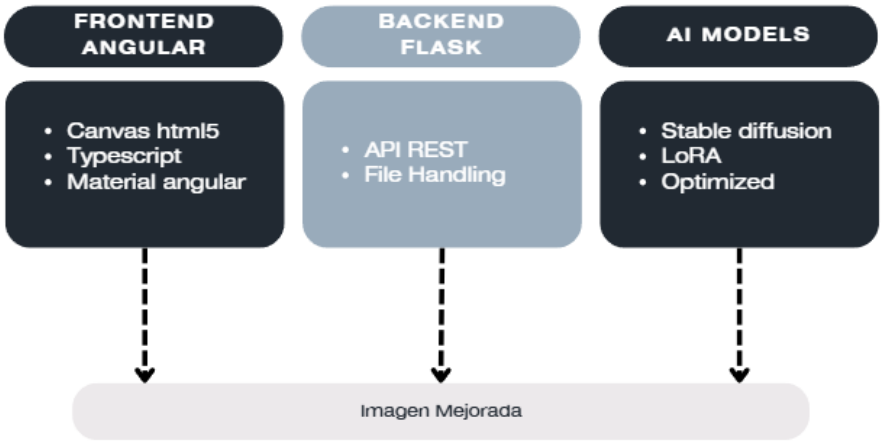
5. Implementación Técnica

5.1. Pipeline del proceso de generación de imagen

En principio el usuario dibuja un sketch en el front de angular, este pasa luego a ser enviado al backend con Flask, el backend se encarga de cargar el modelo que mejorara el dibujo para luego procesar la imagen y generarla, y finalmente se enviar la url de la imagen generada al cliente:



5.2. Arquitectura del Sistema



5.3. Dataset y entrenamiento

Especificaciones del Dataset

- **Tamaño**: 5,000 pares de imágenes
- **Resolución**: 512x512 píxeles
- **Formato**: PNG con transparencia
- **Distribución**: 4,000 entrenamiento, 1,000 validación

Hiperparametros:

```
{
    "model_id": "runwayml/stable-diffusion-v1-5",
    "batch_size": 4,
    "image_size": 512,
    "num_epochs": 40,
    "learning_rate": 1e-4,
    "lora_rank": 16,
    "lora_alpha": 32,
    "mixed_precision": "fp16"
}
```

5.4. Modelo de IA Implementado

Se presenta la arquitectura de StableDiffusion+LoRA:

```
# Componentes del modelo
tokenizer = AutoTokenizer.from_pretrained(model_id, subfolder="tokenizer")
vae = AutoencoderKL.from_pretrained(model_id, subfolder="vae")
unet = UNet2DConditionModel.from_pretrained(model_id, subfolder="unet")
text_encoder = CLIPTextModel.from_pretrained(model_id,
subfolder="text_encoder")
scheduler = DDPMScheduler.from_pretrained(model_id, subfolder="scheduler")

# Adaptación LoRA para fine-tuning eficiente
lora_config = LoraConfig(
    r=16,
    lora_alpha=32,
    target_modules=["to_k", "to_q", "to_v", "to_out.0"],
    lora_dropout=0.1,
)
unet = get_peft_model(unet, lora_config)
```

5.5. Pipeline de generacion

```
def generate_anime_from_sketch(sketch_image, prompt="anime style"):
    """
    Pipeline completo de transformación dibujo
```

```
"""
# 1. Preprocesamiento de imagen
processed_sketch = preprocess_image(sketch_image)

# 2. Carga optimizada del modelo
pipe = load_model_with_memory_management()

# 3. Generación con parámetros optimizados
result = pipe(
    prompt=prompt,
    image=processed_sketch,
    strength=0.75,
    num_inference_steps=30,
    guidance_scale=8.5
).images[0]

# 4. Liberación de recursos
cleanup_memory(pipe)

return result
```

6. Discusión

6.1. Logros Principales

1. **Modelo Especializado Exitoso**: El fine-tuning con LoRA permitió crear un modelo altamente especializado en la transformación dibujo uno mejorado mientras mantenía la eficiencia computacional.
2. **Arquitectura Optimizada**: El sistema funciona consistentemente en hardware con 6GB VRAM, democratizando el acceso a IA generativa avanzada.
3. **Experiencia de Usuario Fluida**: La integración frontend-backend proporciona un flujo de trabajo intuitivo desde el dibujo hasta la generación.

6.2. Limitaciones Identificadas

1. **Dependencia de Calidad del Sketch**: Sketches muy simples o abstractos generan resultados menos coherentes.
2. **Estilo Único**: El modelo está optimizado para un estilo anime específico, limitando la diversidad estilística.
3. **Requerimientos Hardware**: Aunque optimizado, aún requiere GPU dedicada para rendimiento óptimo.

7. Conclusiones

El proyecto Drawnime demuestra exitosamente que es posible crear un sistema especializado de IA generativa accesible que transforme sketches en ilustraciones anime de alta calidad. La combinación de Stable Diffusion con fine-tuning mediante LoRA provee un balance óptimo entre especialización y eficiencia computacional.

La arquitectura implementada resuelve desafíos técnicos significativos en términos de gestión de recursos y integración full-stack, estableciendo un precedente para aplicaciones similares que requieran ejecutar modelos grandes de IA en entornos con recursos limitados.

Referencias

Dave Bergman & Cole Striker. (14 de abril de 2023). *¿Qué es un modelo de difusión?* Obtenido de imb: https://www.ibm.com/es-es/think/topics/diffusion-models

Dimensionia. (07 de agosto de 2023). *Cómo Crean Imágenes los Modelos de Difusión: Una Explicación Sencilla*. Obtenido de dimensionia: https://www.dimensionia.com/como-crean-imagenes-los-modelos-de-difusion/

Heidolf, N. (21 de agosto de 2023). *Efficient Fine-tuning with PEFT and LoRA*. Obtenido de heidloff: https://heidloff.net/article/efficient-fine-tuning-lora/

huggingface. (s.f.). *Diffusers*. Obtenido de huggingface: https://huggingface.co/docs/diffusers/index

Pastor, J. (27 de septiembre de 2024). *Hugging Face es la joya de los "exploradores de IA": ya tiene un millón de modelos para probar*. Obtenido de xataka: https://www.xataka.com/robotica-e-ia/hugging-face-joya-exploradores-ia-tiene-millon-modelos-para-probar

What is Hugging Face? (diciembre de 2021). Obtenido de ibm: https://www.ibm.com/think/topics/hugging-face

wwwhatsnew. (02 de agosto de 2024). *El poder de los modelos de difusión en la creación de contenido digital*. Obtenido de wwwhatsnew: https://wwwhatsnew.com/2024/08/02/el-poder-de-los-modelos-de-difusion-en-la-creacion-de-contenido-digital/