UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

SISTEMA PROTOTIPO DE RENDERIZACIÓN NEURAL EN LA NUBE CON TPU PARA DETERMINAR LA CAPACIDAD DE PRODUCCIÓN DE TRES ARQUITECTURAS

TESIS

PRESENTADA POR:

JEAN PIERRE SOTO CHIRINOS

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS

PUNO - PERÚ

2022

DEDICATORIA

Quiero dedicar el presente trabajo a mis padres Edgar y Maura, a mis hermanos Steven y Heiddy y en especial a mis hermosos sobrinos Mateo, Natalie y Alba.

Jean Pierre.

AGRADECIMIENTOS

A mi familia por darme su amor y apoyo incondicional, en especial a mis hermanos Heiddy y Steven.

A mi tía Nelva y primos por alentarme a seguir adelante y alcanzar mis objetivos.

Un agradecimiento especial a mi asesora Dr. Guina Guadalupe Sotomayor Alzamora por ser mi mentora, guía y apoyo para lograr el presente trabajo, asimismo por motivarme a seguir creciendo como persona y profesional.

Al programa de investigación de Google, TPU Research Cloud, que apoyo a esta investigación con 110 TPU.

Al museo Carlos Dreyer de Puno, por brindarnos y apoyarnos con las imágenes para esta investigación.

También quiero agradecer a mis amigos, Gianella, Juan Carlos, Arnold, Ronny, Marimar y Jhony por estar presentes y apoyarme en cada uno de mis logros.

Ad astra incrementis.

ÍNDICE GENERAL

Pág.

CAPÍTULO II

REVISIÓN DE LITERATURA

TEMA: Renderización Neural en la Nube

ÁREA: Ingeniería Computacional y Sistemas

FECHA DE SUSTENTACIÓN 08 DE NOVIEMBRE DEL 2022

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE ACRÓNIMOS

SPRN	Sistema Prototipo de Renderización Neural
3D	Tres Dimensiones
2D	Dos Dimensiones
NR	Neural Rendering
NN	Redes Neuronales
ML	Aprendizaje Automático
LLFF	Local Light Field Fusion
NSR	Neural Scene Representation
NVS	Novel View Synthesis
MLP	Multi-Layer Perceptron
CPU	Central Processing Unit
GPU	Graphic Processing Unit
GCP	Google Cloud Platform
AI	Artificial Intelligence
MLP	Multilayer Perceptron
GKE	Google Kubernetes Engine
TPU	Tensor Processing Unit

TRC	TPU Research Cloud
VM	Virtual Machine
RGB	Red Green Blue
NeRF	Neural Radiance Fileds
MINCUL	Ministerio de Cultura
UNESCO	The United Nations Educational, Scientific and Cultural Organization

RESUMEN

La Renderización Neural es un proceso que involucra diferentes técnicas de computación gráfica y Aprendizaje Automático de Máquinas. Estos modelos de redes neuronales son capaces de entender y codificar una escena tridimensional. Se promueve la investigación responsable de la Renderización Neural para su uso académico y cultural, con el objetivo de producir modelos sustentables que puedan ser conservados en menores espacios de almacenamiento. El presente trabajo presenta un Sistema Prototipo de Renderización Neural (SPRN) escalable basado en la nube con tres arquitecturas: (1) TPU VM, (2) TPU Node, (3) TPU Node con GKE. Con un conjunto de datos conformado por imágenes del patrimonio cultural de Puno para evaluar el rendimiento de la CPU, RAM y Tiempo de producción para determinar el perfil más óptimo de Máquina Virtual (VM) y la Arquitectura más adecuada, adicionalmente se realizó pruebas de calidad de imagen PSRN y SSIM con los renderizados. Los resultados muestran que el SPRN con arquitectura TPU VM tiene una alta capacidad de rendimiento, con una producción de 1960 modelos en 8 días con 10 TPUs. Sin embargo, no es escalable, utiliza una configuración manual, limitando el uso de las TPUs. La arquitectura TPU Node con GKE tiene una mayor escalabilidad con mínimos recursos de VM, sin embargo, reduce la capacidad de producción en un 50 % y aumentando el tiempo en un 120 %, con una producción de 1440 modelos al día con 100 TPUs. La arquitectura TPU Node tiene un rendimiento similar a la anterior; sin embargo, no es lo suficientemente escalable. Finalmente, concluimos que la arquitectura TPU VM es adecuada para la producción inmediata y experimental; y TPU Node con GKE es funcional para la producción escalable.

Palabras Clave: Campos de Radiación Neural, JAXNERF, Kubernetes, Síntesis de Vistas Novedosas, TPU Research Cloud.

ABSTRACT

Neural Rendering is a process that involves different techniques of computer graphics and Automatic Machine Learning. These neural network models can understand and encode a three-dimensional scene. Responsible research on Neural Rendering is promoted for academic and cultural use, to produce sustainable models that can be preserved in smaller storage spaces. This paper presents a scalable cloud-based Neural Rendering Prototype System (NRPS) with three architectures: (1) TPU VM, (2) TPU Node, and (3) TPU Node with GKE. The dataset is composed of the cultural heritage image's from Puno to evaluate the performance of CPU, RAM, and Throughput Time to determine the most optimal Virtual Machine (VM) profile and the most suitable Architecture, additionally, PSRN and SSIM image quality tests were performed with the renderings. The results show that the SPRN with TPU VM architecture has a high throughput capacity, with a throughput of 1960 models in 8 days with 10 TPUs. However, it is not scalable, and uses manual configuration, limiting the use of TPUs. The TPU Node architecture with GKE has higher scalability with minimal VM resources, however, it reduces throughput capacity by 50 % and increases the time by 120 %, with a throughput of 1440 models per day with 100 TPUs. The TPU Node architecture has a similar performance to the previous one; however, it is not scalable enough. Finally, we conclude that the TPU VM architecture is suitable for immediate and experimental production; and TPU Node with GKE is functional for scalable production.

Keywords: Neural Radiance Fields, JAXNERF, Kubernetes, Novel View Synthesis, TPU Research Cloud.

CAPÍTULO I
INTRODUCCIÓN

Los modelos de Redes Neuronales (NN) actualmente son capaces de entender y codificar una escena de Tres Dimensiones (3D) usando conjunto de imágenes de Dos Dimensiones (2D) con puntos de cámara dispersas, generando sintéticamente nuevos puntos de vista no conocidos de la escena. El desarrollo de nuevos métodos de digitalización une técnicas clásicas de computación gráfica con Aprendizaje Automático (ML) y NN, formando campos de estudio como la Neural Rendering (NR), con técnicas y algoritmos que dan un alcance a nuevos modelos de digitalización sustentable. Los métodos utilizados en la NR están categorizados por sus aplicaciones, estos están divididos en 6 técnicas de computación gráfica: Síntesis de fotos semánticas, Síntesis de Vistas Novedosas, Libre Visualización Objetiva de Video, Re-Iluminación, Representación Facial, Representación Corporal (Tewari *et al*., 2020).

La NR es un proceso de digitalización de escenarios 3D para la libre manipulación del escenario con el objetivo de producir novedosas imágenes 2D hiperrealistas, con un gran potencial para influir en las comunicaciones en la creación de contenido, sin embargo, este puede ser usado de forma inapropiada, es necesario divulgar su uso responsable y digno en áreas académicas, para el desarrollo digital, cultural y moral. La The United Nations Educational, Scientific and Cultural Organization (UNESCO) define patrimonio cultural como un regalo del pasado, que se vive en el presente y hereda en el futuro, considera y establece lugares de la tierra con un valor universal excepcional que pertenece al patrimonio común de la humanidad, asimismo, recursos culturales, educativos, científicos, administrativos, información técnica, jurídica, médica y de otro tipo creada digitalmente o convertida en formato digital a partir de recurso analógicos existentes, que

requieren mantenimiento y una gestión determinada para que se conserven (Batchelor *et al*., 2021; UNESCO, 2003).

Actualmente los museos cuentan con una limitada capacidad de desarrollo e innovación para fomentar la preservación digital de la cultura, además existe un riesgo de pérdida elevado del valor patrimonial si esta no se realiza correctamente. El avance en el campo de computación gráfica y NR suponen un avance en áreas como Síntesis de Vistas Novedosas Novel View Synthesis (NVS) y Representaciones Neurales de Escenas Neural Scene Representation (NSR) haciendo uso de tecnologías en la nube y recursos computacionales de alta capacidad como las TPU, que desde el punto de vista cultural es una opción sustentable.

El uso de tecnologías y servicios en la nube son clave para la producción y mantenimiento de los modelos generados por la NR, que si bien son novedosas, requieren de una gran capacidad computacional, las tecnologías en la nube son capaces de proveer una gran capacidad de recursos para el entrenamiento de NN, tal es el caso de Google Cloud Platform (GCP) y TPU Research Cloud (TRC), que permiten el uso de tecnologías como Kubernetes para gestionar y escalar proyectos de investigación, logrando integrar proyectos con TPU y diferentes servicios para mejorar su producción y escalabilidad. La presente investigación propone determinar la capacidad de producción y calidad entre el uso de arquitecturas TPU VM, TPU Nodo, TPU Nodo con Google Kubernetes Engine (GKE), aportando con una posible solución para la búsqueda de un modelo digital sustentable para la preservación de patrimonio en un Sistema Prototipo de Renderización Neural (SPRN) y soportado por la plataforma GCP gracias al programa TRC el cual brindó bajo demanda 5 Cloud TPUs v3 , 5 Cloud TPUs v2 también bajo demanda y 100 Cloud TPUs v2 interrumpibles para esta investigación.

El presente documento cuenta con cuatro capítulos: En el CAPÍTULO I, se desarrolla la introducción, planteamiento del problema, justificación y alcance abordando desde un punto de vista cultural y tecnológico, asimismo también se mencionan los objetivos e hipótesis de esta investigación. En el CAPÍTULO II, REVISIÓN DE LA LITERATURA, se desarrolla todo el Marco Teórico que soporta esta investigación, donde se detallan también los antecedentes de la investigación. En el CAPÍTULO III, MATERIALES Y MÉTODOS, se presenta la metodología usada, los lineamientos de la investigación, las principales motivaciones, técnicas e instrumentos de recolección de datos, la operacionalización de variables, los métodos y materiales empleados en la investigación, el lugar de estudio y Dataset. En el CAPÍTULO IV, RESULTADOS Y DISCUSIÓN, se exponen los resultados de la investigación, donde se detalla a profundidad las diferencias entre las arquitecturas TPU VM, TPU Nodo y TPU Nodo con GKE. Finalmente se presentan las CONCLUSIONES y RECOMENDACIONES.

1.1 Planteamiento del problema

La integración y aplicación de nuevas tecnologías basadas en NN e Artificial Intelligence (AI) está revolucionando productos y servicios, empresas como Google, Microsoft, AWS, entre otros que ofrecen sus servicios de entrenamiento en la nube poniendo a disposición del usuario herramientas y recursos necesario para el entrenamiento de modelos neuronales, estas mismas empresas brindan a sus nuevos usuarios un free trial o cupón de bienvenida, para poder hacer uso temporal de sus servicios, GCP ofrece un cupón de 300$ por 3 meses en su plataforma.

Este cuenta con restricciones, especialmente para la virtualización de Virtual Machine (VM), limitando el uso Central Processing Unit (CPU) y restringiendo el uso de Graphic Processing Unit (GPU) y Tensor Processing Unit (TPU), las cuales son necesarias para mejorar el tiempo de entrenamiento de las NN, sin embargo, se tiene a TRC, un programa creado por Google que brinda recursos TPU a investigadores para acelerar sus proyectos de AI.

Estos recursos, relativamente gratuitos, son importantes para la aplicación de tecnologías basadas en AI, en especial las orientadas a la NR, recursos que pueden ser utilizados óptimamente por parte de instituciones académicas y culturales como el Ministerio de Cultura (MINCUL). La Ley N° 28296, Ley general del patrimonio cultural de la nación, del 2016 en la cual se establece el marco legal de protección del patrimonio cultural, presentado por el MINCUL, se menciona que es responsabilidad y derecho inherente de la población y de las instituciones culturales, como los museos, a cargo del patrimonio cultural en proteger, mantener y ayudar a trascender el patrimonio cultural del Perú.

Hasta la fecha existen 56 museos registrados en el Sistema Nacional de Museos del cual un pequeño porcentaje de museos han logrado ser parte de una transformación digital, publicando museos virtuales con más de 800 modelos 3D culturales como: Cerámica, Esqueletos, sitios Arqueológicos, Instrumentos, Herramientas, etc. en su plataforma web y en la plataforma de SketchFab, haciendo uso de un presupuesto aproximado de S/.160,000.00, adquiriendo hardware y software especializado para la digitalización de patrimonio cultural.

Dichas tecnologías suponen un reto de mantenimiento y utilización, limitando su uso al resto de los museos, Apablaza (2021) informa respecto a la transformación digital en los museos del Perú, mencionando que en el Perú muchos empresarios son reticentes

ante la tecnología, lo que provoca que la transformación digital sea lenta, al tener niveles bajos de tecnología hace que sea casi imposible mantener un ritmo constante de plantear metodologías para una transformación a un medio digital, a nivel internacional Champion y Rahaman (2019) y Parry (2010) mencionan que muchos museos han cambiado en los últimos años siendo más los museos que fomentan la práctica de adoptar tecnología para brindar una mejor experiencia a los visitantes.

Finalmente se menciona que es importante poder integrar y aplicar tecnologías basadas en AI haciendo uso de recursos en la nube, optimizando el consumo computacional y maximizando la producción de modelos que representan escenarios culturales cerámicos, iniciando una propuesta de digitalización sustentable.

1.2 Formulación del problema

¿Es posible la implementación de un Sistema Prototipo de Renderización Neural en la nube para determinar la capacidad de producción de tres arquitecturas?

1.3 Justificación

En respuesta a la búsqueda de un modelo sustentable para la preservación y digitalización de patrimonio digital en los últimos años hubo grandes avances en el campo de la AI, existen tecnologías basadas en NR para representar estos modelos 3D a partir de unas cuantas imágenes funcionales para calcular la densidad y el color de una escena en función de coordenadas de 3D como es el caso de NeRF realizado por Mildenhall *et al*. (2020b) que funciona como un Perceptrón Multicapa Multilayer Perceptron (MLP) de coordenadas capaz de entender y sintetizar vistas novedosas hiperrealistas y así mismo su sucesor "*NeRF in the Wild*" de Martin-Brualla *et al*. (2020) para imágenes más complejas

y de alto tránsito como el fototurismo.

Estas tecnologías permiten la reconstrucción precisa no sólo de objetos sino también de lugares emblemáticos y famosos, abriendo campo en la NR de escenas, guardando la representación de la escena en los pesos de la red neuronal reduciendo drásticamente el peso digital de representaciones de modelos 3D. Sin limitarse a ello, en muchos de los proyectos derivados como en el trabajo de Z. Li *et al*. (2020) y Peng *et al*. (2020) se ve el potencial que puede dar esta tecnología y aumentar el valor cultural, tecnológico y educacional a los patrimonios digitales, al poder modelar y dar nuevas perspectivas de una escena en movimiento, el cual potenciaría diferentes áreas de la ingeniería, social y biomédica, con aplicaciones en reconstrucción de escenarios forenses en tiempo real y en movimiento, digitalización de flora y fauna en conservación y su interacción con el medio, aplicaciones en metodologías de medición como zoometría evitando la intervención humana, reduciendo el estrés de los animales como es en el caso de los animales marinos que requieren salir de su entorno natural para su medición. Este trabajo ofrece una herramienta con un enfoque cultural para la producción de patrimonio digital sustentable.

1.4 Alcances

El presente estudio propone tres arquitecturas para la Renderización Neural sustentable en un Sistema Prototipo, realizando una comparativa entre las tres, utilizando servicios de Compute Engine y GKE de GCP con el objetivo de aprovechar los recursos TPU brindados por TRC en las arquitecturas propuestas.

1.5 Objetivos

1.5.1 Objetivo General

Implementar un Sistema Prototipo de Renderización Neural en la nube para determinar la capacidad de producción de tres arquitecturas.

1.5.2 Objetivos Específicos

- Integrar y adaptar JAXNERF con una API/REST para el SPRN.
- Establecer perfiles VM para TPU VM, TPU Nodo y TPU Nodo en GKE.
- Evaluar el rendimiento del SPRN en TPU VM, TPU Nodo y TPU Nodo con GKE para determinar la capacidad de digitalización de cada arquitectura.

1.6 Hipótesis

1.6.1 Hipótesis General

Se determina la capacidad de producción de tres arquitecturas del Sistema Prototipo de Renderización Neural en la nube

1.6.2 Hipótesis Específicas

- JAXNERF se integra adecuadamente con una API/REST para el SPRN.
- Los perfiles VM funcionan para TPU VM, TPU Nodo y TPU Nodo en GKE.
- Se determina la capacidad de producción de las arquitecturas TPU VM, TPU Nodo y TPU Nodo con GKE evaluando el rendimiento de la SPRN.

CAPÍTULO II
REVISIÓN DE LITERATURA

2.1 Marco teórico

Este capítulo se comprende de dos partes, en la primera sección se desarrollan conceptos fundamentales para esta tesis. En la segunda sección se recopilan los antecedentes que dieron origen a esta investigación, estos son categorizados por nacionales e internacionales.

2.2 Base Teórica

En esta sección se describen fundamentos básicos de la digitalización y patrimonio cultural. Finalmente se desarrollan conceptos dedicados a las nuevas tecnologías de digitalización y su aplicación en la cultura.

2.2.1 Digitalización 3D

La digitalización se puede definir como un modelo matemático de conversión de subconjuntos continuos del plano o espacio, representando los objetos reales, a conjuntos digitales en $\mathbb{Z}^2$ o $\mathbb{Z}^3$ esto se puede ampliar a más dimensiones, donde la digitalización es la transformación de subconjuntos continuos en $\mathbb{R}^n$ a conjuntos digitales en $\mathbb{Z}^n$ (Gonzalez-Diaz *et al*., 2020).

La digitalización 3D alberga una gran cantidad de categorías para separar el tamaño de los patrimonios, para cada uno de estos tamaños existe un tipo de técnica y requerimientos específicos, existe una gran diferencia entre digitalizar un objeto y un monumento, este último, en muchos casos, usaría técnicas tradiciones de la topografía según Pavlidis *et al*. (2007).

2.2.2 Proceso de digitalización

Terras (2015) describe este proceso como un acto de traducción, desde un punto análogo hasta uno digital, sin sustituir al objeto original, con el objetivo de registrar, copiar, transmitir o analizar una señal tan compleja mediante métodos de digitalización.

Una traducción a una forma más sencilla, predecible y procesable para poder transmitirla a un público, un principio de los sistemas de telecomunicación es la de los transmisores y receptores, un ciclo que empieza desde que se quiere enviar una simple señal de bits, o algo más complejo como un documento, imagen o incluso un video a través de una red, señales que pueden transmitirse y volver a ensamblarse en la recepción, para transformarse en algo que podamos percibir como una copia fidedigna. Estos sistemas digitales se basan en los números binarios una secuencia de 1s y 0s que pueden construir documentos, imágenes, sonido, videos, modelos 3D, etc. Entre más compleja sea la representación más bits necesitarán para describirlos y muchos más mecanismos para capturar, almacenar, mostrar, procesar, analizar y convertir la información contenida en flujo de datos binarios.

2.2.3 Métodos digitalización 3D

De acuerdo al tamaño y el alcance los métodos de digitalización de acuerdo con Pavlidis *et al*. (2007), se dividen en 2 tipos y 14 categorías, la Digitalización de objetos cuenta con: Técnicas de escaneo láser; Reconstrucción a partir de luz estructurada; Reconstrucción a partir de la silueta; Reconstrucción a partir de estéreo; Reconstrucción a partir de video; Reconstrucción a partir de sombreado.

Reconstrucción a partir de fotometría; Reconstrucción a partir de enfoque; Reconstrucción a partir de sombras; Sistemas de contacto o Máquinas de medición de coordenadas. Y la Digitalización de monumentos con: Técnicas empíricas; Técnicas de topografía; Técnicas de escaneo láser; Fotogrametría.

De estas 14 categorías tomaremos el concepto de la reconstrucción a partir de video y fotogrametría y profundizaremos en este último.

- **Reconstrucción a partir de Video**

  Esta es una variante de la reconstrucción de estéreo en la cual se utilizan 2 cámaras para extrapolar la geometría, en este caso se utilizarán 2 videocámaras para capturar una secuencia de imágenes del objeto en diferentes ángulos, una limitante es que el objeto tiene que mantenerse quieto. A pesar de que el algoritmo utilizado es rápido la geometría es de muy baja calidad, que puede ser errónea.

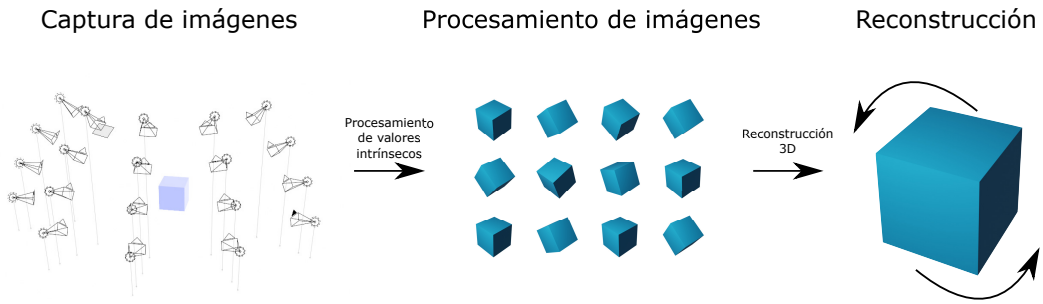
- **Fotogrametría**

  A través de la aplicación de diferentes orientaciones y transformaciones de la fotogrametría digital es posible deducir coordenadas 2D o 3D a partir de unas cuantas imágenes, que pueden ser usadas para objetos con superficies de alto detalle, como se muestra en la Figura 1, sin embargo, al ser una tecnología basada en fotos requiere espacios controlados, ocasionando dificultades con ciertos tipos de entornos no controlados, materiales reflectantes.

  La Sociedad Americana de Fotogrametría y Teledetección lo define como el arte, la ciencia y la tecnología para la obtención de información, registros, métricas e interpretación de imágenes y de patrones, también considerado como la ciencia original del análisis de fotografías (Paul R. Wolf *et al.*, 2014).

Este método permite generar modelos 3D de bienes muebles e inmuebles a partir de imágenes digitales, siendo un método más económico que los escáneres láser (Caro *et al*., 2014), las técnicas de fotogrametría han estado evolucionando hasta llegar a un nivel de digitalización decente, sin embargo, muchas de estas siguen teniendo problemas con algunos tipos de materiales, necesitando realizar procesos extra para poder modelar los objetos, entornos controlados e incluso manipulación directa, aun así estos procesos no garantizan que el modelo final genere geometrías incorrectas entre otros errores.

Figura 1: Proceso de fotogrametría

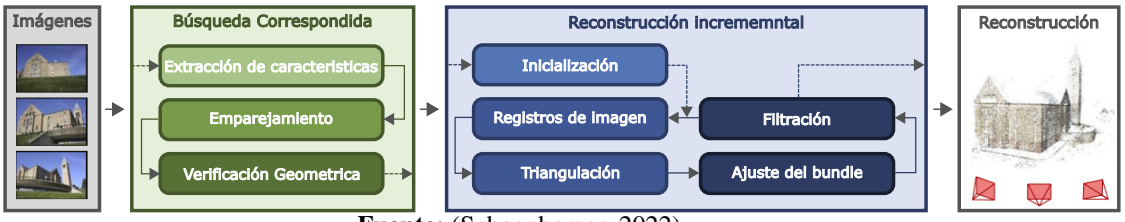


Elaboración propia

- **COLMAP**

Este programa fue escrito por Johannes L. Schönberger, como una línea de producción de estructura a partir del movimiento (SfM) y estereoscópica multivista (MVS) (COLMAP, 2016) con una gama de funciones para la reconstrucción a partir de un conjunto de imágenes ordenadas y desordenadas como se muestra en la Figura 2 .

Figura 2: Proceso de fotogrametría



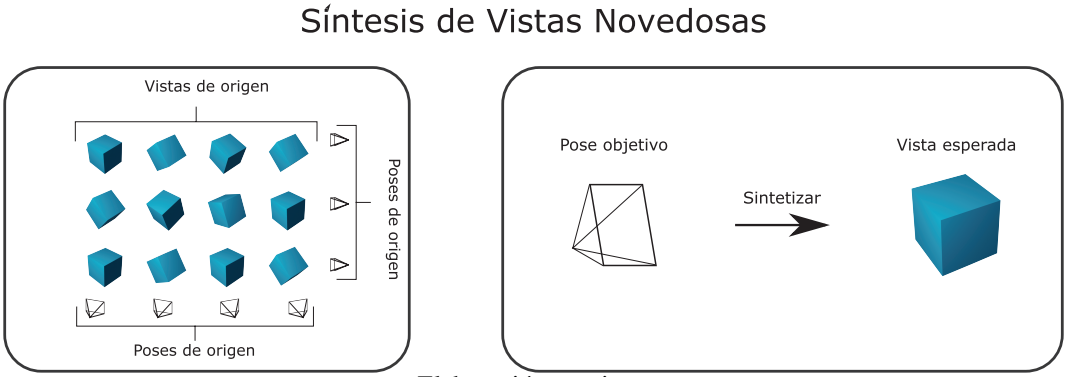
Fuente: (Schoenberger, 2022)

2.2.4 Renderización Neural

Síntesis de Vistas Novedosas

La habilidad que tiene un sistema o programa para sintetizar imágenes con nuevos ángulos y posiciones a partir de un conjunto de imágenes de referencia o entrenamiento (Avidan & Shashua, 1997) podemos remontar este concepto hasta el campo de la ciencia cognitiva, con la descripción de RN y J (1971), Shepard en su experimento rotación mental donde los sujetos de prueba demostraron una capacidad de predicción del aspecto de un objeto 3D, representado en 2D, al aplicar una rotación 3D conocida, Zhou *et al*. (2016) lo describe como el estudio computacional equivalente al experimento de rotación mental llamado NVS.

Figura 3: Proceso del NVS



Elaboración propia

También considerado como un problema de la literatura, con una gran cantidad de métodos que intentan abordar este problema en escenarios controlados, sin embargo, hasta antes de la presentación del novedoso método Neural Radiance Fileds (NeRF) y *NeRF in the wild* era un problema para capturar escenarios en ambientes naturales de alta variación, a pesar de que existen antecedentes que han logrado avances en el campo de visión y gráficos por ordenador para generar NVS a partir de un conjunto de imágenes, como se muestra en la Figura 3, es claro que el método propuesto por Mildenhall *et al*. (2020b) es

superior en los resultados y el hiperrealismo que posee los productos generados.

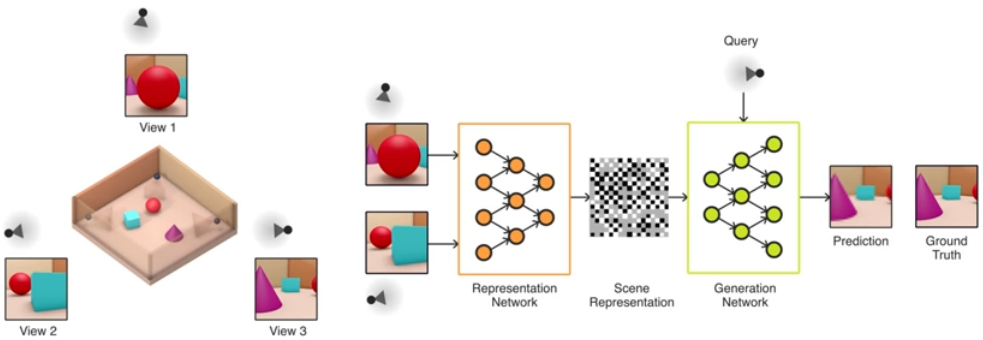
Representación neural de la escena

Este concepto según Kohli *et al*. (2020) son representaciones implícitas codifican una escena en los pesos de una red neuronal que puede ser consultada en cualquier coordenada para producir estas mismas propiedades de la escena, *i.e* es la capacidad de una red neuronal de aplicar NVS almacenando su representación en los pesos de las neuronas, a diferencia de otras formas de almacenar representaciones 3D, como son las nubes de puntos, se puede apreciar el proceso en la Figura 4.

Renderización volumétrica neural

Método que genera imágenes o videos usando el Ray Tracing dentro de una escena y tomando una integral de algún tipo sobre la longitud del rayo, usualmente es una red neuronal NSR, como un MLP de coordenadas, codifica una función a partir de las coordenadas 3D del rayo a cantidad como la densidad y el color, que se integran para obtener una imagen (Dellaert & Yen-Chen, 2020).

Figura 4: Proceso del NVS



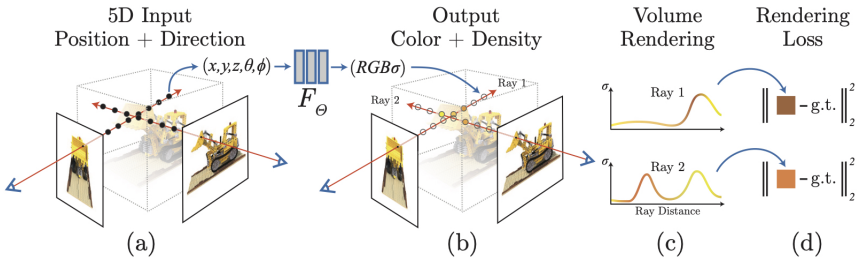
Fuente: (Eslami, 2018)

2.2.5 Campos de Radiación Neural

Este método es capaz de capturar un representación volumétrica en 3D de una escena dentro de los pesos de una red neuronal como un NSR, reduciendo drásticamente el peso de la representación multimedia, el método entrena una red totalmente conectada, usando un vector en 5D (x, y, z, θ, ϕ) de forma continua, al mapear a través de una codificación posicional, una coordenada de posición p $\rightarrow$ (x, y, z) en $\mathbb{R}^3$ y una dirección de visualización (o dirección del rayo) d $\rightarrow$ (θ, ϕ) en $\mathbb{R}^2$ de un conjunto de imágenes, para poder pasarlo por la red neuronal totalmente conectada $f_{nerf}(p, d) \rightarrow (c, \sigma)$ para obtener el color $c \rightarrow (R, G, B)$ y la densidad σ (Garbin *et al.*, 2021).

Usa la codificación posicional para poder saltar el sesgo espectral que se produce al someter MLP de coordenadas a variaciones de alta frecuencia, las cuales decaen rápidamente y tienen un tiempo de aprendizaje muy lento, NeRF tiene la asombrosa capacidad de modelar efectos dependientes del punto de vista, como reflejos, destellos, y sombras, la cual es una característica sobresaliente de esta tecnología y que mejora los métodos tradicionales de reconstrucción 3D mostrado en la Figura 5.

Figura 5: Proceso del Campo de Radiación Neural



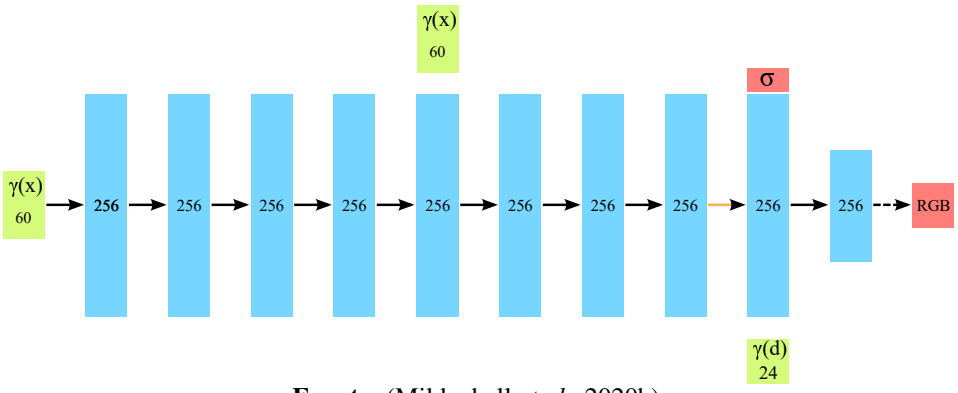
Fuente: (Mildenhall *et al.*, 2020a)

Para poder entrenar la red NeRF se requiere un conjunto de imágenes, así como los valores intrínsecos de cada una respecto al punto de vista capturado, en cada interacción se usa un subconjunto de píxeles del conjunto de entrenamiento al azar, cada pixel genera

una dirección de visualización, a su vez se restringe la red para predecir la densidad de volumen $\sigma(p)$ con una función que usa la posición p, sin embargo, que para obtener el color se usa una función $c(p,d)$ qué usa tanto la posición p como la dirección de visión, Mildenhall et al. (2020) lo describe como un MLP que procesa las coordenadas 3D de la posición en 8 capas con una activación *RELU* y 256 canales por capa, conformando 2 grupos de redes totalmente conectadas, adicionando una red que usa una función de activación Relu y 128 canales, la que da como resultado el color Red Green Blue (RGB) dependiente de la vista

Estos dos parámetros tanto como la densidad $\sigma(p)$ y el color $c(p,d)$ son entrenados para minimizar la diferencia entre lo renderizado y el valor del pixel observado sobre los rayos generados de las imágenes de entrenamiento (Rebain *et al.*, 2020a), esta red neuronal se aprecia en la Figura 6.

Figura 6: Red Neuronal NeRF



Fuente: (Mildenhall *et al.*, 2020b)

2.2.6 Kubernetes

Se refiere a un orquestador de código abierto, capaz de crear espacios seguros para aplicaciones escalables usando instancias de VM para la virtualización de un contenedor a partir de una imagen, originalmente creada por Google (Burns *et al.*, 2018). Utiliza modelos declarativos para especificar el estado esperado del sistema, implementando una

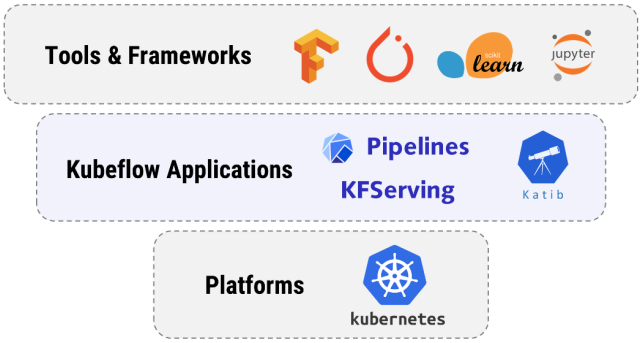
lógica de control que ayuda a mantener y conciliar el estado real del sistema con el estado esperado. Almacena, y controla los estados del sistema a través de una API/REST y utiliza los pods, que son una representación de estos estados, capaces de almacenar varias instancias de contenedores. Así, Kubernetes gestiona el ciclo de vida de estas variables y artefactos, además de otras unidades de procesamiento como el GPU y/o TPU descrito en un manifiesto de tipo YAML. El servicio de GKE de GCP es capaz de gestionar el auto escalamiento de los recursos computacionales haciendo uso de los gestores de recursos como el servicio de Artifacts (Cox *et al*., 2020; Kanso *et al*., 2021; Wu & Song, 2020).

2.2.7 Kubernetes y aprendizaje automático

Los Kubernetes crearon la base para una multitud de proyectos enfocados a proporcionar servicios básicos como el registro, las métricas, la gestión de la red y la seguridad y proporcionar un floreciente ecosistema Cloud Native (Cox *et al*., 2020).

Además de ello en los últimos años proyectos de NN, ML, entre otros se basan y utilizan Kubernetes, incluso dando como resultado una herramienta exclusiva para el uso de Kubernetes con proveedores en la nube. KubeFlow es un marco de trabajo de ML gratuito y de código abierto, que ayuda a gestionar el ciclo de vida los proyectos de ML (George & Saha, 2022), su integración con Kubernetes se muestra en la Figura 7.

Figura 7: Marco de trabajo de Kubeflow



Fuente: (Dkube, 2021)

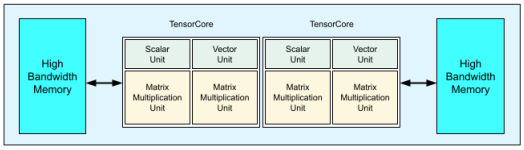
2.2.8 TPU

La TPU fue diseñada por Google para el uso exclusivo de entrenamiento de una NN profunda, acelera la inferencia 50 veces más en comparación con otras unidades de procesamiento como la CPU y GPU. El núcleo de esta tecnología utiliza una unidad de multiplicación de matrices que ocupa la mayor parte del tiempo en cada una de sus operaciones (Gundi *et al.*, 2020; N. Jouppi *et al.*, 2018; N. P. Jouppi *et al.*, 2017). A nivel de hardware y servicio se tiene:

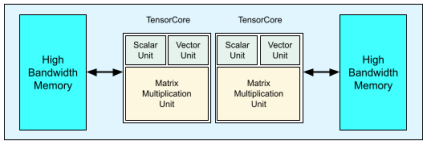
- Por versión: Se tienen 2 versiones del TPU disponibles en GCP, la V2 y la V3, que se componen de diferentes cantidades de núcleos y memoria, mostrados en la Figura 8. Google provee las cantidades y tipos de agrupación dependiendo de la zona donde se encuentren ubicados. En este trabajo se usó la versión v2-8 perteneciente al grupo Single Devices, con una sola unidad o placa de TPU, de 8 núcleos y 64GiB de memoria (Google, 2022d; Norrie *et al.*, 2021).

Figura 8: Versiones del TPU

TPU v3



TPU v2

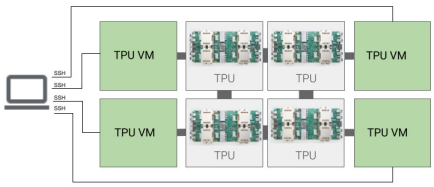


Fuente: (Google, 2022d)

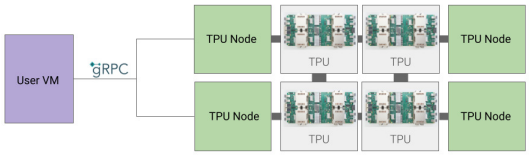
- Por servicio en la nube: Se tienen las arquitecturas TPU Nodo y TPU VM. Las TPU VM son instancias de TPU alojadas en un host VM de alta capacidad al que se puede acceder por SSH (Secure SHell) y tiene configuración propia; y las TPU Nodo son instancias de TPU alojadas en un host inaccesible, por lo que requieren instancias de VM personalizadas mostrado en la Figura 9. Su proceso de configuración depende del framework utilizado, sea TensorFlow, Pythorch o JAX, que requieren una configuración especializada. Además TPU Nodo es escalable al usar tecnologías como GKE, Artifacts o DockerHub, Cloud Storage y Github (Google, 2022a; Wudenhe & Tseng, 2021).

Figura 9: Servicios TPU

TPU VM



TPU Nodo



Fuente: (Google, 2022d)

2.2.9 JAXNERF

JAXNERF utiliza JAX para acelerar el entrenamiento de NeRF en la Nube, de ahí su nombre (JAX + NeRF), usando GPUs o TPUs para minimizar el tiempo de entrenamiento y maximizar el desempeño de NeRF. A través de los recursos y servicios de la nube como GCP, JAXNERF es capaz de usar un pod entero de TPUs de hasta 128 cores, con una capacidad de memoria de 1 TiB (Google, 2022b), reduciendo en gran medida el tiempo

de entrenamiento de un modelo en comparación con NeRF. Así, con JAXNERF se tiene un total de 2.5 horas, comparado a los 3 días que se obtendría con NeRF usando una GPU NVIDIA V100 (Deng, Barron & Srinivasan, 2020).

Tabla 1: Soporte de la implementación JAX

Plataforma	**GPU de un solo host**		**TPU multidispositivo**	
Tipo	**Dispositivo único**	**Multi Dispositivo**	**Host Único**	**Múltiples Host**
Entrenamiento	Aceptable	Aceptable	Aceptable	Aceptable
Renderización	Aceptable	Aceptable	Aceptable	Aceptable

Elaboración propia

El uso y manejo de recursos TPU por parte de GKE es un factor primordial en este trabajo, para instancias de VM personalizadas con características mínimas, a fin de crear instancias en contenedores para diversas réplicas de imágenes elaboradas a cierta medida para su uso con TPU Nodo, usando el mínimo de recursos para un entrenamiento adecuado. En la siguiente sección se realiza la descripción de todo el proceso para la elaboración del Sistema Prototipo de Renderización Neural propuesto.

2.2.10 Patrimonio Digital

He *et al*. (2017) define el patrimonio digital como la integración del patrimonio cultural con la tecnología de digitalización (”Patrimonio cultural “+”digitalización”), también lo describe como ”Recursos e información únicos con valor y significado a largo plazo producidos por medios digitales”, lo que concuerda con la descripción detallada de Manovich (2021), acerca de las bases de datos como formas simbólicas marcando una gran diferencia entre las bases de datos (información, archivos, modelos 3D, multimedia, etc.) y la narrativa (cultura). La información y la narrativa crea una trayectoria de causa y efecto de acontecimientos aparentemente desordenados, también menciona que existe

una brecha entre estas dos de carácter filosófico, y al igual que muchos otros conceptos es relativo a la perspectiva del espectador.

Las bases de datos soportan una serie de formas culturales que van desde la traducción directa hasta una forma cuya lógica es la opuesta a la lógica de la propia forma material. La UNESCO se refiere al patrimonio digital como contenido constituido por materiales informáticos de valor duradero que deben de conservarse para las generaciones futuras (UNESCO, 2003) proveniente de diferentes medios como comunidades, industria, sectores y regiones, sin embargo, no todos son de valor duradero, pero los que lo tienen, requieren enfoques de preservación activa si se quiere mantener su continuidad.

2.2.11 Digitalización del Patrimonio

Se define como la aplicación de la tecnología digital en el ámbito del patrimonio cultural, centrado en la documentación, análisis de datos, rehabilitación y revitalización a través de medios digitales con el almacenamiento permanente de la información sobre el patrimonio y la difusión pública basada en las telecomunicaciones (He *et al*., 2017).

La cantidad y mantenimiento de información en internet es muy grande, la transformación y pérdida de estas son constantes, dando origen a más información, incluso en intranets como lo son los sistemas internos de los museos, enfrentan constantemente los riesgos que implica el cambio tecnológico y su adaptación (Parry, 2010); existen varios enfoques para combatir la obsolescencia del software y hardware, que consisten en la transformación y adaptación de los archivos a nuevas plataformas, que manejan formatos estándar y no propietario, ya que facilita su mantenimiento a largo plazo y mejora su escalabilidad.

2.2.12 Desarrollo incremental o evolutivo

Sommerville (2011) explica que el desarrollo incremental o evolutivo tiene un enfoque que vincula actividades de especificación, desarrollo y validación, lo que requiere una serie de versiones con incremento de funcionalidades del mismo sistema exponiendo cada versión a los comentarios de los usuarios para su respectiva retroalimentación, este tipo de desarrollo permite reflejar la forma en la que se resolverá un problema como un historial de cambios, permitiendo retroceder cuando existe algún tipo de error, la mayor ventaja es su proceso evolutivo, que permite desarrollar con menos recursos y en menor tiempo que otros modelos de procesos de software, además que se cuenta con la versatilidad de realizar cambios de acuerdo a la evolución del mismo, lo que nos permite considerar que eventualmente podría desarrollarse un sistema final a partir de este como lo indican Basil y Turner (1975).

También se debe considerar sus desventajas, el desarrollo ágil de este modelo no permite tener procesos visibles y documentables por cada versión del sistema, su estructura al basarse en un prototipo tiende a degradarse conforme se incrementan nuevas funcionalidades, lo cual implica un gasto extra de tiempo y recursos en refactorización y documentación del sistema, lo que implica que entre más cambios tiene un sistema más grande se vuelve la curva de mantenimiento, documentación refactorización, etc.

El desarrollo de un sistema bajo este enfoque seguirá las siguientes directrices en fase de análisis para asegurar su correcto desarrollo de acuerdo a Basil y Turner (1975):

- Cualquier dificultad en el diseño, la codificación o la depuración de una modificación debería indicar la necesidad de rediseñar o recodificar los componentes existentes.

- Las modificaciones deben encajar fácilmente en módulos aislados y fáciles de encontrar. Si no es así, será necesario un rediseño.

- Las modificaciones de las tablas deben ser especialmente fáciles de realizar. Si alguna modificación de las tablas no se hace rápida y fácilmente, entonces está indicado un rediseño.

- Las modificaciones deberían ser más fáciles de hacer a medida que avanzan las iteraciones. Si no es así, entonces hay un problema de base, como un flujo de diseño o una proliferación de ”parches”.

- Normalmente debería permitirse que los ”parches” existan sólo durante una o dos iteraciones. Sin embargo, los parches deberían permitirse para evitar el rediseño durante una fase de implementación.

- La implementación existente debe ser analizada con frecuencia para determinar si está a la altura de los objetivos del proyecto.

- Siempre que se disponga de medios de análisis de programas, éstos deben ayudar a analizar las implementaciones parciales.

- La reacción de los usuarios debe ser siempre solicitada y analizada en busca de indicaciones de deficiencias en la implementación existente.

2.2.13 Evaluación de la calidad de la imagen

Una propiedad principal de las imágenes, cuya finalidad es medir la degradación y contenido de ruido que existe en una imagen, comparando, normalmente, la degradación con una imagen ideal de referencia existente. Es posible describir la calidad de una imagen técnica y objetivamente indicando la desviación en comparación con el modelo original, utilizado para la evaluación de técnicas de predicción de imágenes como NVS.

Existen muchas técnicas de calidad de imagen, tales como: MSE (Mean Square Error), UIQI (Universal Image Quality Index), PSNR (Peak Signal to Noise Ratio), SSIM (Structured Similarity Index Method), HVS (Human Vision System), FSIM (Feature Similarity Index Method), etc. Las técnicas de evaluación utilizadas para este trabajo son PSNR y SSIM (Horé & Ziou, 2010; Sara *et al*., 2019).

PSNR

El PSNR calcula la relación entre la potencia del ruido distorsionado y la máxima potencia posible de la señal que influye directamente en la calidad de la representación de la imagen. La PSNR suele calcularse como un término logarítmico en decibelios, debido a que el rango dinámico de la señal es muy alto. Este rango dinámico varía entre los valores máximos y mínimos posibles y puede ser modificado por su calidad (Sara *et al*., 2019).

SSIM

El método del índice de similitud estructural es un modelo basado en la percepción. Este método, considera un cambio en la percepción de la información estructural como degradación de la imagen. El término información estructural hace hincapié en los píxeles fuertemente interdependientes o en los píxeles espacialmente cerrados.

Estos píxeles altamente interdependientes se refieren a información más importante sobre el objeto visual dentro del área de la imagen. Combina con otros principios basados en la percepción, como el enmascaramiento de la luminancia, el enmascaramiento del contraste, etc. El enmascaramiento de luminancia es un término que designa las partes distorsionadas de una imagen que son menos visibles en los bordes de la misma.

Por otro lado, el enmascaramiento de contraste es un término que designa las distorsiones que también son menos visibles en la textura de una imagen. SSIM estima la calidad perceptiva de las imágenes y los vídeos. Mide la similitud entre dos imágenes: la imagen original y la imagen restaurada (Horé & Ziou, 2010).

2.2.14 Herramientas de desarrollo

Para integrar e implementar un Sistema Prototipo de Renderización Neural se requieren de herramientas y tecnologías basadas en la nube, así como un lenguaje de programación versátil que sea compatible con las TPUs.

Python

Diseñado por Guido van Rossun, un lenguaje de programación utilizado para el ML, con varias librerías de NN como Tensorflow así como librerías de operación de matrices como JAX, compatibles para su uso con TPUs (Van Rossum & Drake, 2009).

Miniconda

Es un entorno mínimo de Conda, el cual puede gestionar y administrar versiones de Python en entornos seguros de ejecución, con un pequeño número de paquetes útiles, este será útil para poder realizar pruebas en estado de desarrollo (Crowley, 2012), puede acceder a este software a través de este enlace.

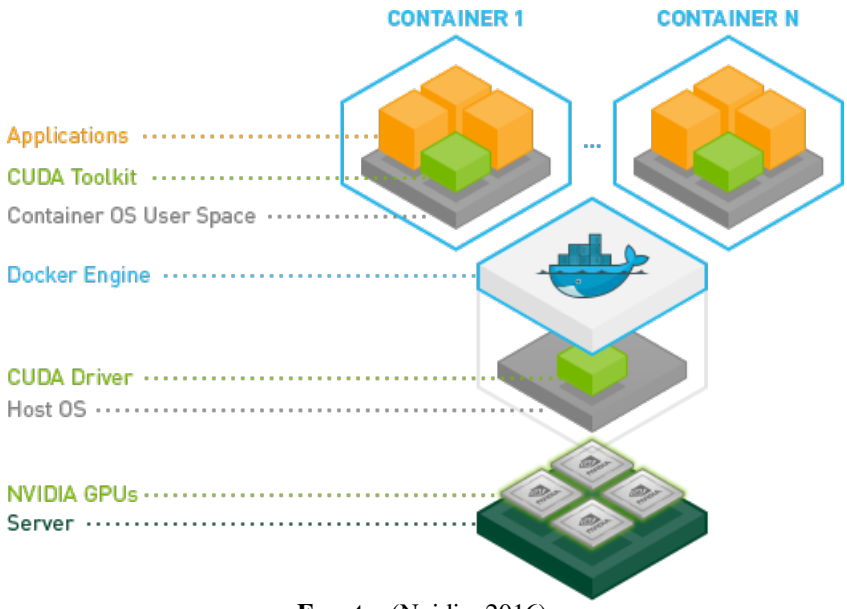
Docker

Es una plataforma abierta, para la virtualización de infraestructuras, separando en un nuevo entorno del entorno local, permitiendo gestionar las nuevas infraestructuras virtuales e incluso personalizarlas agilizando el desarrollo de las aplicaciones, puede visitar el siguiente enlace para ver su documentación: Docker.

WLS2, Nvidia y Docker

El Subsistema Windows para Linux es un producto de Windows desarrollado por años basado en subsistemas aislados para crear un entorno de prueba para Linux desacoplándolo del sistema anfitrión. La versión 2 es capaz de virtualizar la GPU Física para su uso en Docker con el objetivo de acelerar proyectos de ML, así como se muestra en la Figura 10, realizando configuraciones específicas en Windows e instalación de drivers para hardware y software de NVIDIA para su compatibilidad con el Sistema Operativo de Windows, para más información puede visitar el siguiente enlace: WLS 2 + Nvidia + Docker.

Figura 10: Contenedores acelerados por GPU



Fuente: (Nvidia, 2016)

Local Light Field Fusion (LLFF)

Local Light Field Fusion (LLFF) es una herramienta de aprendizaje profundo para la captura y renderizado de nuevas vistas de escenas complejas reales para la Libre Visualización Objetiva, utilizando COLMAP para extraer valores intrínsecos del conjunto de imágenes de entrenamiento (Mildenhall *et al.*, 2019), consulte el repositorio oficial para más información: LLFF.

Flask

Escrito en Python, es un framework minimalista que permite crear servicios API/-REST, con múltiples librerías compatibles con una variedad de base de datos y servicios incluyendo SQLALCHEMIST y GCP (Copeland, 2008), puede visitar el siguiente enlace para ver su documentación: Flask.

SQLAlchemist

Es un kit de herramientas SQL para Python diseñado para un acceso eficiente y de alto rendimiento a la base de datos, para mayor información del paquete y su instalación puede visitar el siguiente enlace.

Google Cloud Platform

GCP ofrece una amplia cantidad de servicios en la nube con el objetivo de asegurar, servir, almacenar y analizar datos, estos están en un ecosistema seguro dentro de la nube, donde el usuario puede realizar diferentes operaciones y transformaciones sobre los datos sin que estos se filtren de la nube (Bisong, 2019), dentro de los servicios utilizados en esta tesis tenemos:

- Cloud Engine: Servicio de GCP que brinda recursos computacionales de VM (Virtual Machines) también conocidos como instancias haciendo uso de CPU, RAM y discos montables principalmente, sin embargo, para trabajos y proyectos más profesionales tiene servicios como la personalización de imagen de la VM, uso de contenedores, gestión de grupos de instancias, monitoreos, signos de salud, recursos TPU y GPU, etc.

- IAM y Admin: Servicio para la gestión de roles de todos los recursos, gestión de políticas y roles para cada cuenta de servicio, scopes, etc.

- Cloud Storages: Servicio de almacenamiento de archivos, imágenes, repositorios, etc.

- Artifact Registry: Servicio de control y registro de artefactos (Imágenes Docker, Repositorios Git, versiones del programa, etc.).

- CloudBuild: Servicio de construcción de imágenes, aplicaciones, programas de alto nivel, etc.

2.3 Antecedentes

Para el apartado de antecedentes tomaremos como referencia la estructura realizada por Dellaert y Yen-Chen (2020), en su trabajo NEURAL VOLUME RENDERING: NERF AND BEYOND donde describe los avances que se obtuvo desde la publicación de Mildenhall *et al*. (2020b) con su artículo NeRF en el campo de Renderización de volúmenes neuronales (Neural Volume Rendering) y NSR donde se citan los mejores y más relevantes avances relacionados con el método de Campos de Radiación Neural (NeRF), el cual estará dividido por:

- Desempeño

- Escenas dinámicas

- Imágenes de alta varianza

Además, que en este apartado se tocaran la relevancia de estos avances no solo para la comunidad de visión y gráficos por ordenador sino también como una alternativa a la digitalización activa de patrimonio cultural, que amplía el uso de los patrimonios culturales a un área educativa y de entretenimiento, dándoles un mayor valor digital, proponiendo y sentando las bases de un medio digital sustentable para la preservación de patrimonios digitales.

2.3.1 Trabajos Internacionales

Mucho antes del método NeRF hubo varios antecedentes muy cercanos a lo que conocemos como Renderización de Volúmenes Neuronales, enfoques que usan una red neuronal para definir una representación implícita de la superficie, en el CVPR 2019 más de 3 trabajos introdujeron el uso de NN como aproximadores de funciones para definir funciones de ocupación y/o distancia de signo (Dellaert, 2020).

Entre las más resaltantes están las Redes de ocupación, IM-NET, DeepSDF y PI-Fu, introdujeron implícitamente NN con aprendizaje por coordenadas, con la función de predecir la ocupación binaria dado una coordenada 3D y vector de características, o en algunos casos solo una coordenada 3D, por consiguiente se realizaron investigaciones para poder combinar estas funciones con otras que además de dar ocupaciones puedan producir y reconstruir profundidades e imágenes RGB, proyectos como CvxNet, BSP-Net, Deep Local Shapes, Scene Representation Network, Neural Articulated Shape Aproximation

(NASA), entre otros, son trabajos dentro de la literatura con aproximaciones muy cercana a la Renderización del Volumen Neural.

El trabajo NeRF de Mildenhall toma la arquitectura de DeepSDF pero en vez de regresar una función con distancia de signo regresa la densidad y el color, capaz de retornar un renderizado volumétrico, usando un método de integración numérica, podemos considerarlo como un método de Neural Volumen Rendering, este proyecto tuvo bastante interés por que no solo era capaz de integrar un NTK estacionario para la reconstrucción de escenas con las Características de Fourier para así sintetizar vistas hiperrealistas, sino que también es capaz de generar un mapa de profundidad con un alto detalle, el gran impacto que tuvo es por su brutal simpleza, al usar solo un MLP totalmente conectado de coordenadas de 5 dimensiones para obtener el color y densidad, este artículo es de gran referencia para este proyecto, así como todas investigación realizadas desde su publicación en Agosto del 2020, como se mencionó anteriormente se mencionara rápidamente los trabajos relacionados e inspirados por este así como sus mejoras y funcionalidades.

Desempeño

A pesar de que NeRF es un método bastante novedoso, tiene un gran defecto y es que el entrenamiento de un objeto no se puede inferir para otros, además que el tiempo de entrenamiento del proyecto original puede tardar hasta días en producir un renderizado decente, dependiente de GPU, es así como nacen proyectos para superar este límite de procesamiento, haciendo uso de computación distribuida, técnicas de entrenamiento reforzado, descomposición de escenas, organización y distribución de escena en un octree de vóxeles dispersos, etc. de los cuales resalta JaxNeRF, DeRF, NERF++, Neural Sparse Voxel Fields, entre otros, esto da una oportunidad a NeRF como un medio sustentable

contra otros métodos en función del tiempo (Deng, Barron & Srinivasan, 2020; Liu *et al*., 2020; Rebain *et al*., 2020b; Zhang *et al*., 2020).

Escenas dinámicas

NeRF es capaz de entender la varianza que hay entre escenas de un objeto estático, lo cual lo orilla a funcionar mejor con entornos controlados y con imágenes, sin embargo, tiene dificultades con entender la deformación en función del tiempo, un conjunto de imágenes o video con una alta tasa de deformación no dará un buen resultado, es como nacen estos proyectos capaces de entender y encontrar la deformación y varianza entre cada frame de un video, dentro los más destacables tenemos a Nerfies y D-NeRF, Nerfies como una combinación entre NeRF y Selfie, un modelo capaz de entender la deformación de un video de un rostro con resultados hiperrealistas, asimismo tenemos proyectos como NR-NeRF, STaR, Space-Time Neural Irradiance Fields, entre otros (Park *et al*., 2020; Pumarola *et al*., 2020; Tretschk *et al*., 2020; Xian *et al*., 2020; Yuan *et al*., 2021). Lo proyectos que también calculan la varianza pero tienen como entrada un video son Neural Scene Flow, Neural Body, Neural 3D Video Syntesis y Dynamic View Synthesis from Dynamic Monocular Video, de estos resalta Neural Body, capaz de usar múltiples vistas de cámara y *motion capture* para obtener los movimientos del cuerpo humano, sin duda un proyecto con gran potencial para áreas como el estudio del comportamiento de flora y fauna, sin la necesidad de interferir, o en la preservación de danza tradicionales sin la necesidad de recrear escenas complejas en modelos 3D (Gao *et al*., 2021; T. Li *et al*., 2021; Z. Li *et al*., 2020; Peng *et al*., 2021).

Imágenes de alta varianza

Como ya mencionamos NeRF aún necesita de entornos controlados para producir modelos interesantes, sin embargo, una gran actualización agregando una red neuronal para encontrar la varianza y ruido entre imágenes de alta varianza con diferentes característica y cámara como lo es el Fototurismo, ayuda a poder generar escenas de grandes estructuras, esculturas, monumentos, etc. que están en constante cambio ambiental, así es como se define NeRF in the Wild, un proyecto realizado por los mismos autores de NeRF, que tiene como objetivo poder generar escenas de alta calidad con imágenes tomadas de internet subidas por turistas, para reconstruir patrimonios monumentales (Martin-Brualla *et al*., 2020).

2.3.2 Kubeflow en el CERN

El CERN cuenta con servicios basados en Kubeflow para la "preparación de datos y análisis interactivo, entrenamiento de modelos distribuidos a gran escala y servicio de modelos", mostrando una comparativa y evaluación de costes y tiempo del escalado en una nube pública utilizando GPUs y TPUs. Kubeflow es un proyecto basado en Kubernetes orientado al despliegue, escalado y gestión de soluciones de aprendizaje automático, ayudando a ejecutar flujos de trabajo en miniclusteres en pseudo servidores hasta ejecuciones en nubes públicas de alta capacidad, permitiendo a los desarrolladores a centrarse en el desarrollo de los modelos de aprendizaje automático (Golubovic & Rocha, 2021).

2.3.3 Trabajos Nacionales

Hasta el momento no se ha realizado ninguna investigación de algún método similar a la renderización de volúmenes neurales aplicado en patrimonio digital en el Perú o

en otras áreas, sin embargo, hay un conjunto de artículos relacionados con el uso de Inteligencia Artificial en el campo de visión y gráficos por ordenador, así como un modelo que combina los métodos de Fotogrametría y Escaneo Láser obteniendo resultados de alta calidad, esto supone un reto y una oportunidad para esta investigación, a continuación desarrollaremos los trabajos más similares al área de visión y gráficos por ordenador:

Herrera *et al*. (2019) propone una mejora de la segmentación semántica de imágenes médicas 3D en NN convolucionales 3D, este trabajo fue realizado en el Instituto de Investigaciones de la Amazonía Peruana con el objetivo de desarrollar y analizar una función de perdida para mejorar la segmentación binaria semántica de imágenes médicas volumétricas utilizando la arquitectura Dense V-Net, la cual dio excelentes resultados al mejorar la arquitectura del CNN3D adaptando una métrica de similitud, denominada coeficiente de correlación de Matthews, como función de pérdida, dando como resultado puntuaciones globales más alta de Dice para sus predicciones.

Pineda *et al*. (2020) propone un enfoque de Red Generativa Adversaria (GAN, en sus siglas en inglés) para la Superresolución (VHR) de imágenes de satélite Sentinel-2, el objetivo es entrenar una GAN para poder mejorar la resolución de 41 imágenes proporcionadas por el PeruSay-1, evaluaron la efectividad de su modelo usando métricas de PSNR y SSIM que muestra confianza del modelo para aplicarlo en tareas de mejora de calidad de imágenes.

Lambers *et al*. (2007) combina métodos como la Fotogrametría y Escaneo Láser para el registro y modelado del sitio del Periodo Intermedio Tardío de Pinchango Alto, Palpa, Perú, describiendo el modelo 3D de Pinchango Alto usando una combinación de datos de imagen y alcance, demostrando la eficiencia en el detalle incluso en condiciones desfavorables, demostrando que la aplicación de dos sistemas de digitalización de alta

versatilidad produce modelos de alta calidad y precisión.

2.3.4 Trabajos Locales

A pesar de ser la capital del Folklore no hay muchos trabajos relacionados con la digitalización de patrimonio cultural en la ciudad de Puno, lo que puede suponer un gran factor de resistencia a la implementación del SPRN, teniendo en cuenta que en la bibliografía se precisa que la principal motivación es la educación es necesario introducir modelos sustentables que valga la pena preservar.

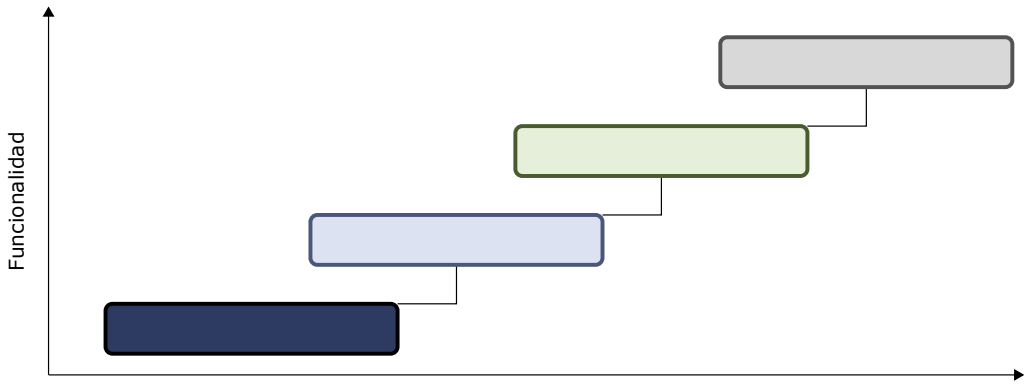
En el reporte ”Mapeo Fotogramétrico En Arqueología: Experiencias Desde El Proyecto Ramis, Cuenca Norte Del Titicaca, Perú” de Flores (2015) se realiza un registro usando mapeo digital fotogramétrico en un sistema SIG, que fue practicado en la cuenca del Titicaca, una región altiplánica, donde demuestra que el método de recolección de datos ”puede traer no solo rapidez sino también precisión en las mediciones”, se concluye del reporte que los registros que se obtienen del mapeo fotogramétrico permitirá una persistencia de datos a través del sistema SIG para futuras investigaciones, cabe aclarar que el estudio se dio en medio de un cambio a la digitalización del papel, se toma como referencia frente a nuevos métodos de digitalización y preservación de datos.

El trabajo de Ticona Quispe (2021) plantea e identifica características arquitectónicas potenciales histórico culturales que posee la ciudad de Puno, que corresponden a espacio formales, espacios funcionales y de contenido las cuales se integran en un museo virtual, como resultado se generó el concepto de un museo físico virtual con características y funcionalidades tecnológicas y arquitectónicas.

CAPÍTULO III
MATERIALES Y MÉTODOS

Este capítulo muestra los métodos, dataset y aplicación de herramientas utilizadas para integrar JAXNERF en un Sistema Prototipo de Renderización Neural. En la primera sección se describen los datos, el conjunto y configuración de imágenes utilizadas, así como el grupo de muestra y su preprocesamiento con LLFF. La segunda sección contiene las arquitecturas diseñadas con configuraciones de perfiles de VM para interactuar con SPRN, así como la configuración de la infraestructura diseñada y compilada en Cloud-Build con Docker cumpliendo con el primer objetivo específico. La tercera sección describe el diseño del SPRN y el proceso de monitorización de recursos, en la Figura 11 se muestra el proceso o pipeline de la implementación cumpliendo con el segundo objetivo específico. Finalmente se describe el proceso de evaluación de modelos con PSNR y SSIM cumpliendo con el tercer objetivo específico.

Figura 11: Flujo de Trabajo de la implementación



Elaboración propia

El estudio se ajusta a un tipo de investigación cuasiexperimental con un diseño contrabalanceado de acuerdo a Canales *et al*. (1994) y Sánchez Carlessi *et al*. (2018), este diseño divide los grupos prueba en (Xn) y los tratamientos (Otn) para la evaluación, este

proyecto contempla tres modelos de patrimonio cultural (A, B, C) con sus respectivas redimensiones, éstas son tratadas por tres tipos de arquitecturas con el SPRN (Otn).

El primer tratamiento corresponde al TPU VM, el segundo al TPU Nodo con VM y el tercero al TPU Nodo con GKE, se mide el tiempo de ejecución media por modelo, recursos RAM y CPU, costo de los servicios usados, capacidad de digitalización y alcance (Cantidad de máquinas usadas y escalabilidad) como se muestra en la Tabla 2.

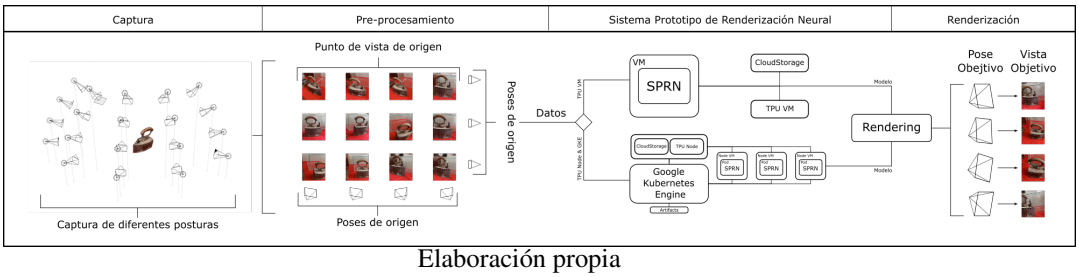
Tabla 2: Diseño Contra-balanceado

Modelos	TPU VM	TPU Nodo	TPU Nodo con GKE
A 100 %	X1a Ot1	X1a Ot2	X1a Ot3
B 100 %	X1b Ot1	X1b Ot2	X1b Ot3
C 100 %	X1c Ot1	X1c Ot2	X1c Ot3
A 50 %	X2a Ot1	X2a Ot2	X2a Ot3
B 50 %	X2b Ot1	X2b Ot2	X2b Ot3
C 50 %	X2c Ot1	X2c Ot2	X2c Ot3
A 25 %	X3a Ot1	X3a Ot2	X3a Ot3
B 25 %	X3b Ot1	X3b Ot2	X3b Ot3
C 25 %	X3c Ot1	X3c Ot2	X3c Ot3

Elaboración propia

Los procesos del SPRN son divididos en cuatro partes, primero se realiza la captura de las imágenes del escenario desde múltiples puntos de vistas, seguido del preprocesamiento de estas imágenes para capturar los datos necesarios para el SPRN, posteriormente se realiza la subida, verificación, redimensión y entrenamiento en el SPRN con servicios en la nube, finalmente se renderiza 120 *frames* de los modelos entrenados, como se muestra en la imagen 12.

Figura 12: Flujo de procesos del SPRN

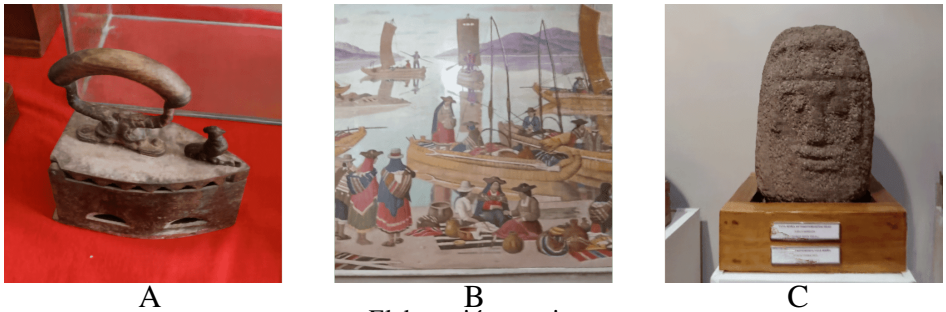


Elaboración propia

3.0.1 Datos de la investigación

Se estableció tres escenarios 3D culturales como se muestra en la Figura 13 para el entrenamiento y evaluación del SPRN, que componen conjuntos de imágenes capturadas desde una cámara móvil, el tamaño del grupo se divide en tres grupos de imágenes redimensionadas, por lo tanto cada grupo tendrá una menor dimensión a la anterior, en tamaño en MB y dimensiones Px, este se divide en: los de menor peso, los de peso medio y los de mayor peso, se usa un algoritmo de redimensionamiento escalar para cada uno de estos grupos, los de menor peso tendrán un redimensionamiento del 25 %, para los de peso medio un 50 % y los de mayor peso se utilizará el tamaño original un 100 %.

Figura 13: Patrimonio cultural



A B C

Elaboración propia

Las imágenes capturadas deben ser preprocesadas por grupos en dos fases antes de poder iniciar entrenamiento para el SPRN. Primero se preprocesarán con LLFF para extraer los valores intrínsecos necesarios para el entrenamiento, estos se componen de una

base de datos que contiene los puntos de cámara, puntos 3D, datos de las imágenes y un array Numpy que contiene las poses de cámara poses_bounds.npy, dichos datos tienen que tener compatibilidad y coherencia entre las escenas. Finalmente se aplica un algoritmo de redimensionamiento para generar tres grupos por conjunto de imágenes como se muestra en la Tabla 3 .

Tabla 3: Dataset

Modelo	Redimensión	Dimensiones	Peso Total
A	100 %	2080x4624	43,4 MB
B		3472x3472	125 MB
C		3472x4624	209 MB
A	50 %	1040x2312	44,9 MB
B		1736x1736	114 MB
C		1736x2312	192 MB
A	25 %	520x1156	12,9 MB
B		868x868	31,9 MB
C		868x1156	56,2 MB

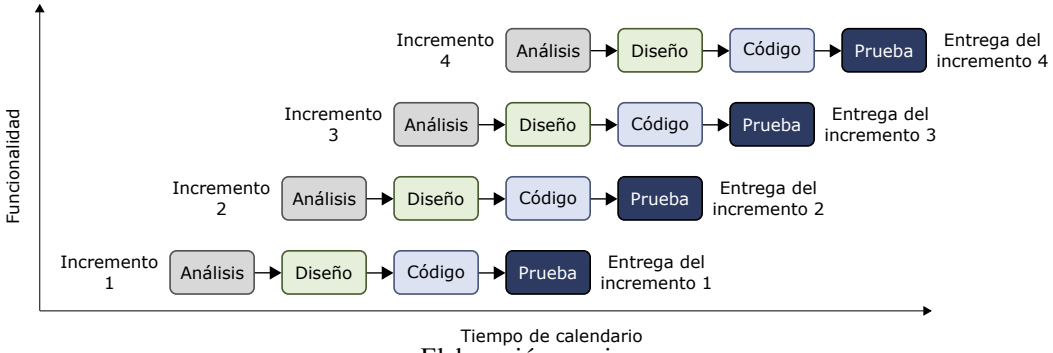
Elaboración propia

3.0.2 Diseño del SPRN

Para el modelo de desarrollo del SPRN se toma el modelo de desarrollo incremental o evolutivo (basado en prototipos de enfoque evolutivo), Sommerville (2011) menciona que un modelo de proceso de software es una representación simplificada de este proceso, cada modelo del proceso representa a otro desde una perspectiva única.

El cual ofrece sólo información particular acerca del proceso en sí tal como lo hace un *framework*, una estructura de trabajo que se utiliza en este proyecto como herramienta de perspectiva arquitectónica que ayuda a detallar más a profundidad actividades específicas, la Figura 14 muestra el ciclo de vida del SPRN.

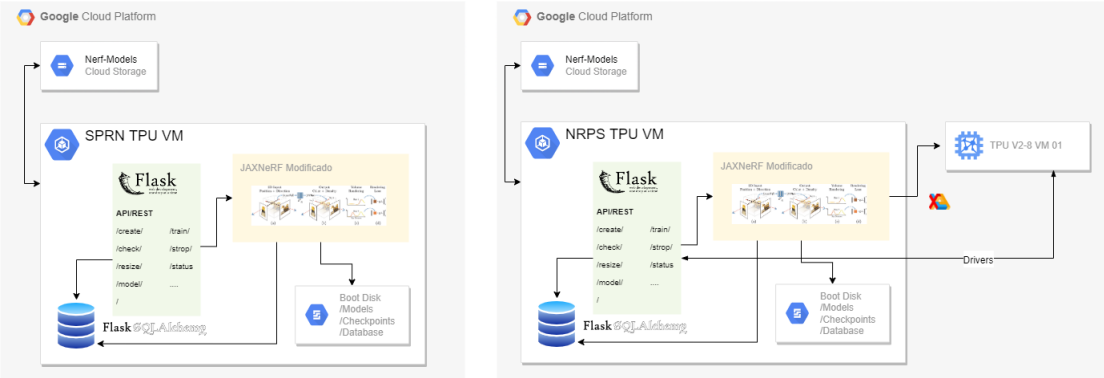
Figura 14: Ciclo de Vida del SPRN



Elaboración propia

El diseño del servicio de SPRN utiliza Flask para la construcción de una API/REST y FlaskSQLAlchemy para el registro de modelos, datos de monitoreo, la API/REST es capaz de utilizar el código de JAXNERF modificado para realizar una conexión con la TPU y empezar el entrenamiento del modelo, el código está almacenado en GitHub haciendo uso de una *branch* secundaria para hacer utilizar el código del SPRN compatible con GKE en TPU Nodo y TPU VM, el diseño es mostrado en la Figura 15.

Figura 15: Diseño SPRN para TPU VM y TPU Nodo



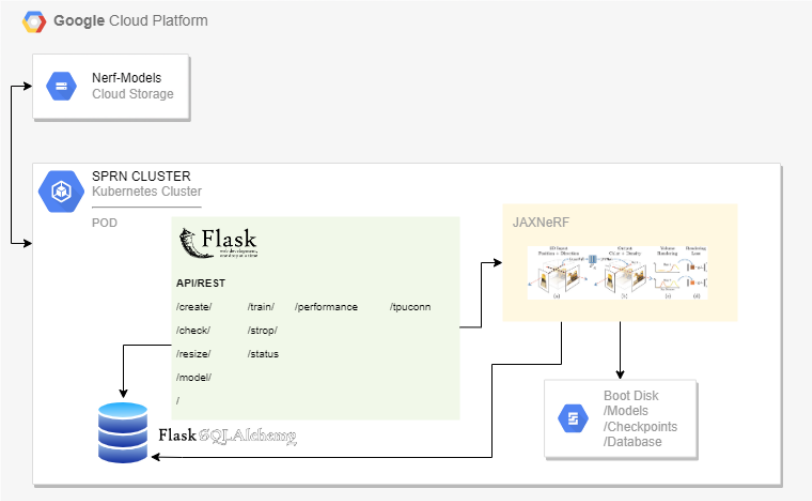
Elaboración propia

API/REST

La API/REST se utiliza para el monitoreo y control de JAXNERF, cuenta con diez endpoints y librerías para el control de subprocesos, tiempo, entorno y peticiones además de librerías personalizadas con variables de entorno y utilidades para el preprocesamien-

to, almacenamiento y estado de las imágenes, este consta de 4 partes. La primera parte está encargada de crear, verificar y redimensionar el conjunto de imágenes almacenadas en la nube, así como la disponibilidad de CPU y RAM. La Segunda está encargada de solicitar y verificar la configuración de los drivers de la TPU. La Tercera está encargada de realizar el entrenamiento haciendo uso de subprocesos, que realiza la ejecución de un comando personalizado al código de JAXNERF, esté además verifica si existe un proceso que esté utilizando la TPU para no generar conflictos, así como detener el proceso de entrenamiento. La última parte se encarga de evaluar el rendimiento y verificar si aún está en proceso el entrenamiento verificando los datos emitidos por JAXNERF como se muestra en la Figura 16.

Figura 16: Solicitudes HTTP de la API/REST

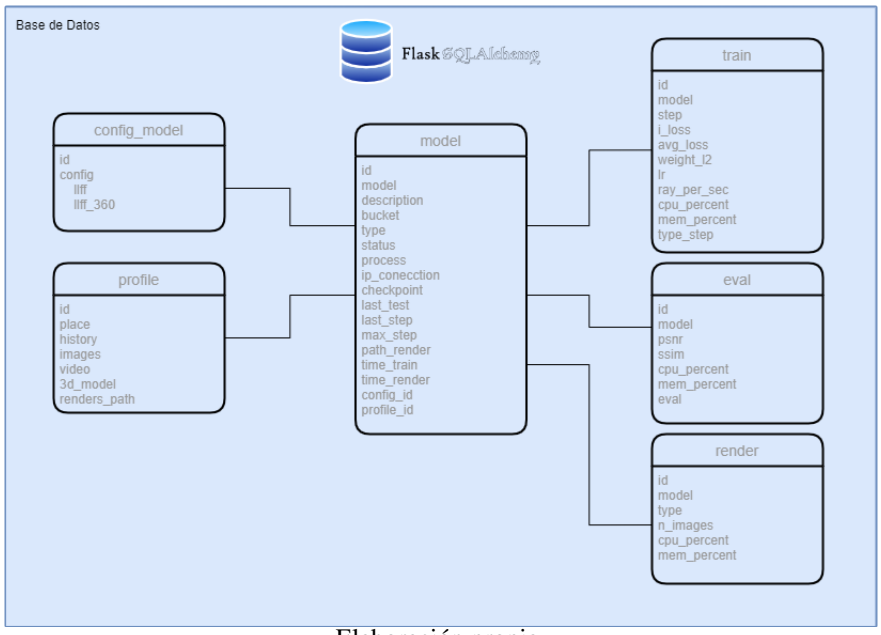


Elaboración propia

Base de Datos

Este cuenta con seis tablas relacionales como se muestra en la Figura 17, estos están diseñados para la recolección de datos, estos son utilizados para la creación de perfiles mínimos para su utilización con TPU Nodo y TPU Nodo en GKE.

Figura 17: Diseño de la Base de Datos (BD)



Elaboración propia

Máquina Virtual (VM)

Se tomó en cuenta tres entornos de pruebas para el diseño del SPRN. Primero un entorno local utilizando WSL2 en Windows 11 configurado para la virtualización de GPU con un sistema Ubuntu 20.04 LTS con 12 CPUs, 16 GB de RAM y una GPU NVIDIA GeForce GTX 1050 Ti de 4GB. Este entorno local es para el preprocesado del conjunto de las imágenes de entrenamiento y desarrollo del SPRN. El segundo en una TPU VM con infraestructura TPU v2.7.0 de 96 CPUs, 340 GB de RAM y TPU v2 – 8 cores 64 GB para el entrenamiento de los modelos. Finalmente el tercero cuenta con los perfiles VM seleccionados a partir del rendimiento del SPRN en las TPU VM, como se muestra en la Tabla 4.

3.0.3 Diseños y Arquitecturas Propuestas

El SPRN se diseñó como un servicio de renderización neural basado en la nube para su uso con las arquitecturas TPU VM, TPU Node y TPU Node con GKE. La construcción

Tabla 4: Perfiles VM

	Local	TPU Host	VM	
Características	**Personal**	**Ideal**	**Intermedio**	**Mínimo**
Imagen	Ubuntu 20.04	Tensor-Flow 2.7.0	Tensor-Flow 2.7.0	
Tipo	-	N1	N1	E2
Perfil	-	n1-356-96-tpu	n1-highmem-8	e2-highmem-4
Cores	12 CPUs	96 CPUs	8 CPUs	4 CPUs
RAM	16 GB	340 GB	52 GB	32 GB
Zona	-	us-central1-f	us-central1-f	
TPU	GPU 4GB	TPU v2-8	TPU v2-8	
Costo x hora	-	4,50$	0,44$	0,24$

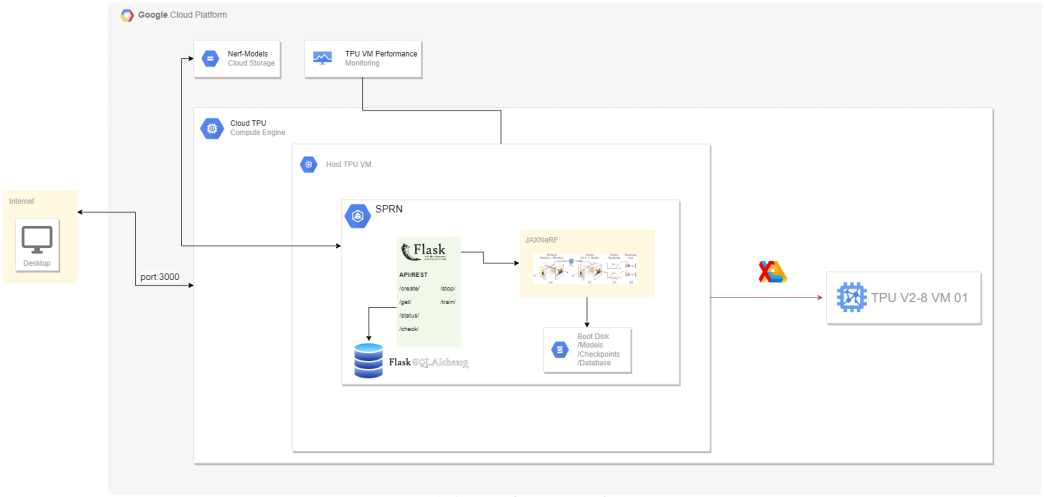
Elaboración propia

de estas tres arquitecturas usa múltiples servicios de GCP (Ifrah, 2021). Cada diseño es monitoreado para evaluar su rendimiento y será impulsado con el servicio Cloud TPU, usando una API/REST en conjunto con SPRN. A continuación, las arquitecturas utilizadas en cada caso, la construcción de la imagen y la descripción de la API/REST utilizada.

TPU VM

El diseño de esta arquitectura está orientado a un desarrollo de prueba, usando servicios como el Cloud Storage y Cloud TPU. Además, utiliza un entorno minimalista Miniconda (Inc., 2022), un entorno de desarrollo para Python, el cual facilita la instalación de los requerimientos del SPRN, en la Figura 18 se visualiza el diseño de la arquitectura, a partir de esta arquitectura se logra definir el perfil de las VMs que se utilizaran en las siguientes arquitecturas, por lo tanto esta será la primera arquitectura en probar, el proceso se muestra en la Sección 4.3 y los resultados de los perfiles en la Tabla 4.

Figura 18: Diseño TPU VM

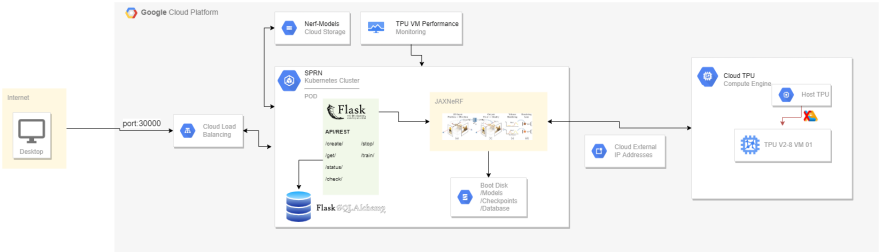


Elaboración propia

TPU Nodo

Esta arquitectura utiliza los mismos servicios que la anterior, agregando el servicio de Compute Engine para instanciar una VM, conectando los recursos TPU la librería de Colab acondicionada al código de JAXNERF utilizando los drivers del TPU. La creación de este entorno fue realizada desde la consola del GCP, creando en paralelo la VM y el TPU, la Figura 19 muestra la arquitectura de este diseño, este diseño utiliza los perfiles Intermedio y Mínimo de la Tabla 4, el propósito de esta arquitectura es establecer los requerimientos necesarios para diseñar una infraestructura y su despliegue con Kubernetes, para lo cual se utilizara Docker.

Figura 19: Diseño TPU Nodo

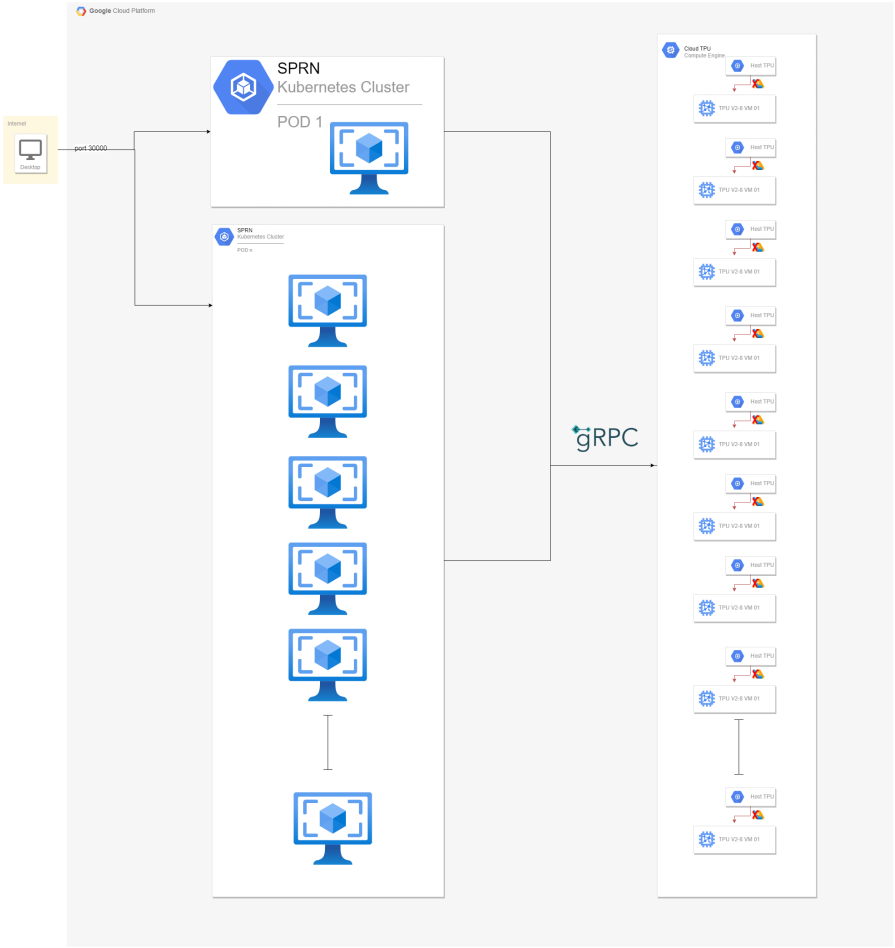


Elaboración propia

TPU Nodo con GKE

Esta arquitectura es diseñada para el uso de múltiples servicios de GCP, desde la gestión de recursos con Artifacts, GitHub, Cloud Storage y principalmente GKE, así como el uso de servicios de red para la exposición de la API/REST en el puerto 3000, creando y conectando los recursos TPU con los contenedores desplegando la aplicación. Este proceso se describe en el manifiesto para el clúster de GKE, como se muestra en la Figura 20. Esta arquitectura de GKE está configurada específicamente para instanciar un solo contenedor por Nodo debido a un requerimiento del TPU, lo que implica una restricción en la selección del Nodo en el manifiesto del clúster Kubernetes, instanciando la infraestructura registrada en Artifacts.

Figura 20: Diseño TPU Nodo con GKE

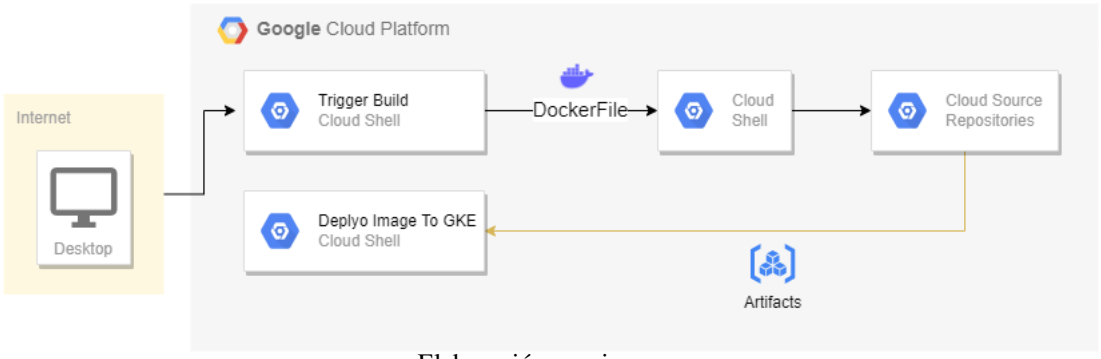


Elaboración propia

Construcción de la infraestructura

Para la construcción de la infraestructura se usó Docker, describiendo todas las características necesarias para el funcionamiento del SPRN en el manifiesto Dockerfile, obtenidas de las pruebas realizadas en la arquitectura TPU Node. La imagen construida es almacenada en Cloud Storage, administrada y distribuida por Cloud Build y Artifacts, brindando un enlace de distribución para su utilización en GKE, como se muestra al lado izquierdo de la Figura 21.

Figura 21: Diseño de la infraestructura con Cloud Build y Docker

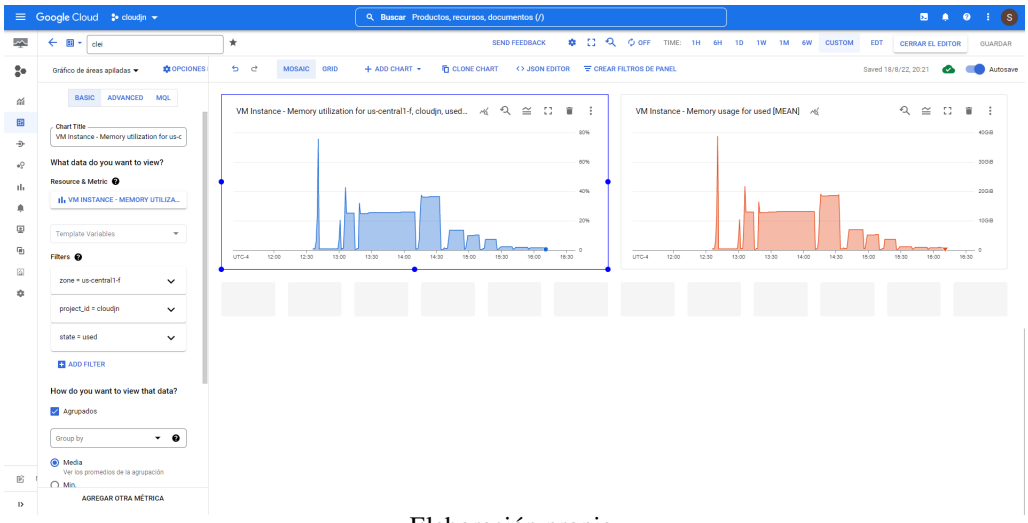


Elaboración propia

Evaluación de rendimiento de recursos

Para la evaluación del rendimiento se utilizó el servicio de monitoreo de GCP llamado Monitoring, donde se encuentra un servicio de paneles que puede filtrar la información de las TPUs y VMs utilizados en tiempo real, por los cuales se puede realizar consultas personalizadas con el objetivo de obtener datos de los recursos utilizados como: CPU y RAM así como se muestra en la Figura 22.

Figura 22: Panel de Monitoring



Elaboración propia

Evaluación de calidad de imagen

Las evaluaciones y comparación de imágenes serán al azar, se tomará una muestra de tres imágenes para realizar las pruebas PSNR y SSIM, estas serán realizadas en las TPU VM, debido a su alta capacidad de CPU, las pruebas PSNR y SSIM realizan evaluaciones con operaciones matriciales, la cual puede saturar las CPUs de los otros dos perfiles y tomar un largo tiempo, la solicitud a la red JAXNERF será tomando los valores de cámara y posición de tres imágenes al azar del conjunto de imágenes de prueba, se realizaran la prueba PSNR y SSIM para cada uno de los experimentos realizados.

Consideraciones finales

El diseño evolutivo contempla el desarrollo propuesto en este capítulo, el diseño y desarrollo de cada arquitectura está integrada con el SPRN y serán probadas y ejecutadas en el mismo orden de mención, TPU VM, TPU Nodo y TPU Nodo con GKE, siendo TPU VM la primera arquitectura en probar el prototipo del SPRN y la TPU Nodo con GKE en probar la última versión del prototipo SPRN, cumpliendo el ciclo de vida del diseño del SPRN Fig. 14 y con los objetivos de esta investigación. La monitorización de datos del

rendimiento en CPU, RAM y tiempo serán realizadas en cada una de las arquitecturas con la versión final del prototipo, así como las pruebas de calidad de imagen para determinar y resolver el objetivo final.

CAPÍTULO IV
RESULTADOS Y DISCUSIÓN

En este capítulo se desarrollan y detallan la implementación y los experimentos realizados. La implementación cumple con el primer objetivo, la Integración de JAXNERF en el SPRN, los experimentos se cumplen con el segundo y tercer objetivo respecto al diseño y experimentación de los prototipos en el siguiente orden: (A) Diseño TPU VM; (B) Diseño TPU Nodo y (C) Diseño TPU Nodo con GKE, cumpliendo todos los objetivos de esta investigación:

4.1 Desarrollo del SPRN

Para la modificación e integración de JAXNERF al SPRN se realizaron 4 configuraciones fundamentales relacionadas a los objetivos de esta tesis, sin embargo, para su integración con TPU VM, TPU Nodo y TPU Nodo con GKE se realizaron configuraciones extras para cada tipo de TPU en una *Branch* nueva en un repositorio GIT. Para la configuración de conexión con las TPUs se contemplaron 3 casos, TPU VM, TPU Nodo y TPU Nodo con GKE descritos en las 3 últimas subsecciones.

4.1.1 Primera Configuración

Se modificó el diseño de JAXNERF, de un código de entrenamiento y evaluación con métricas SSIM y PSNR a un código de entrenamiento y renderización sin métricas para mejorar el rendimiento en la CPU, además se integraron librerías personalizadas y de utilidades para cumplir con las funcionalidades de la API/REST. Los mensajes de consola son eliminados para capturar los datos en una BD, se agregó SQLAlchemist con Flask para recolectar los datos del entrenamiento descritas en el Capítulo III en la subsección

3.0.2, este actualiza los datos de la BD constantemente, lo cual es útil para verificar el rendimiento a través de la API.

Se eliminó la evaluación de cada 5000 iteraciones, aún se mantuvo la preservación de checkpoint cada 10000 It., sin embargo, esta se modificó para almacenar solo un checkpoint y no un registro completo como lo hacía originalmente, asimismo se configuro la evaluación para renderizado de un escenario completo.

4.1.2 Segunda Configuración

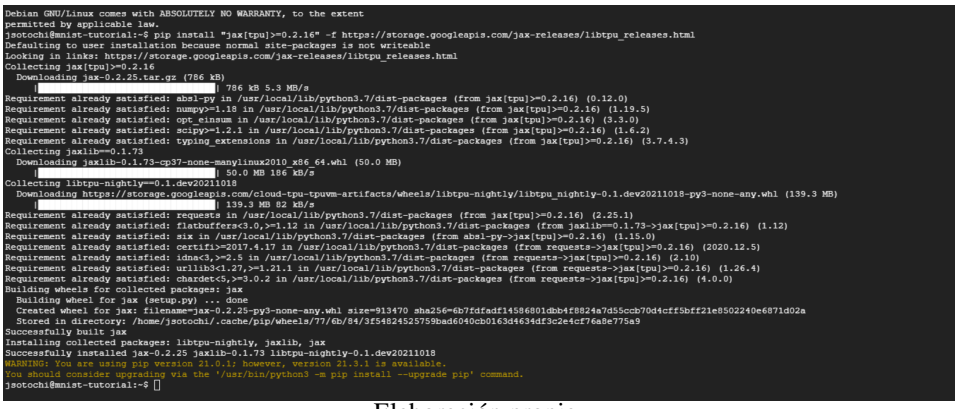
Para la configuración de la TPU VM se utilizó el entorno de pruebas en miniconda sugerido por los mismos creadores de JAXNERF, utilizando como base la versión de Python 3.6.12, la lista de requerimientos es la siguiente:

```
numpy>=1.16.4,<1.19.0
jax>=0.2.6
jaxlib>=0.1.57
flax>=0.2.2
opencv-python>=4.4.0
Pillow>=7.2.0
pyyaml>=5.3.1
tensorboard>=2.4.0
tensorflow>=2.3.1
flask
flask[async]
flask_sqlalchemy
psutil
```

Adicional a ello se requiere una dependencia en Python y en el sistema para la utilización del TPU, en la Figura 23 se muestra la consola con la instalación finalizada:

```
\$ sudo apt-get update
\$ sudo apt-get install bzip2 libxml2-dev -y
\$ curl https://packages.cloud.google.com/apt/doc/apt-key.gpg
   sudo apt-key add
\$ sudo apt-get update
\$ sudo apt-get install subversion -y
\$ sudo apt install libgl1-mesa-glx -y
\$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-
   Linux-x86_64.sh
\$ bash Miniconda3-latest-Linux-x86_64.sh
\$ rm Miniconda3-latest-Linux-x86_64.sh
\$ source .bashrc
\$ pip install --upgrade jax jaxlib==0.1.57+cuda101 -f https://
   storage.googleapis.com/jax-releases/jax\_releases.html
```

Figura 23: Instalación de librerías al TPU VM



Elaboración propia

Se realizó pruebas con el código de JAXNERF modificado para desarrollar la API/-REST, implementando módulos en Python con el algoritmo de redimensión, subfunciones para la obtención de datos para la verificación de subprocesos y recursos; subida, descarga y confirmación de archivos en la nube. Así como los archivos para la inicialización de la BD.

4.1.3 Tercera Configuración

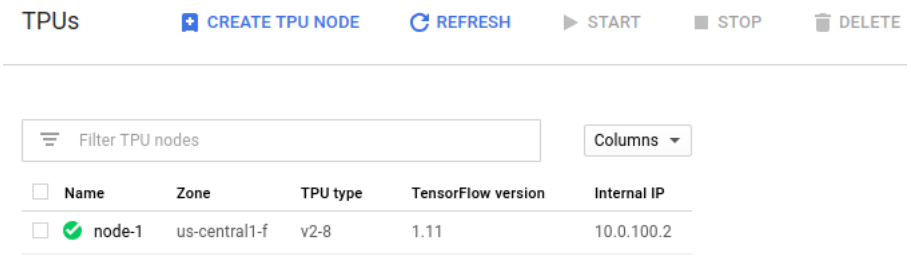
En las pruebas con TPU Nodo se creó una VM y una TPU Nodo a través de la consola del GCP y la API de Cloud TPU, ejecutando el siguiente comando:

```
\$ gcloud compute tpus execution-groups create \
 --name=mnist-tutorial \
 --zone=us-central1-f \
 --tf-version=2.6.0 \
 --machine-type=n1-standard-1 \
 --accelerator-type=v2-8
```

La conexión de TPU Nodo y JAX se realiza a través de una VM, se requieren los drivers necesarios para el TPU, estas librerías pueden ser obtenidas con una consulta por la librería requests en Python, utilizando la dirección IP interna del TPU, generada por la misma API de Cloud TPU como se muestra en la Figura 24, para consultar el driver especifico, el siguiente código representa la solicitud:

```
import requests
path = IP_TPU_NODO
url = IP_TPU_NODO+':8475/requestversion/tpu_driver_nightly'
reqq = requests.post(url)
```

Figura 24: Instalación de librerías al TPU VM



Elaboración propia

La integración con el código de JAXNERF modificado fue necesario agregar una librería de conexión segura a través de XLA, sin embargo, la alteración de variables y flags internos que se realizaron como primera prueba salieron fallidos, como se muestra en la Figura 25.

Para ello es necesario ajustar e integrar una librería de conexión utilizada en plataforma COLAB, que consulta una variable de entorno llamada COLAB_TPU_ADDR que se modificara con la IP interna del TPU Nodo, manteniendo la conexión desde el código de entrenamiento y de renderización de JAXNERF, como se muestra en el siguiente:

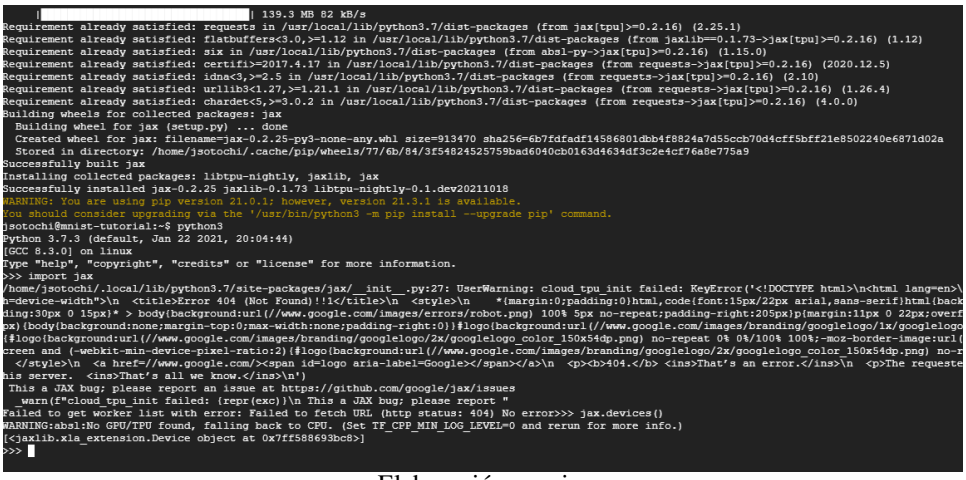
Modificación de la variable de entorno

```
\$ export \$COLAB_TPU_ADDR=\$IP_TPU_NODO:8470'
```

Integración de la librería COLAB

```
import jax
import jax.tools.colab_tpu
jax.tools.colab_tpu.setup_tpu()
```

Figura 25: Conexión de una VM a una TPU Nodo



Elaboración propia

4.1.4 Cuarta Configuración

Para la configuración de conexión con las TPU Nodo con GKE se realizó una configuración en el manifiesto YAML, tomando en cuenta los recursos en el Artifacts y los TPU disponibles, el siguiente código muestra parte del manifiesto utilizado, así como el servicio de Service para el balanceo de carga y redes.

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nerf-2
  template:
    metadata:
      name: nerf-pods-2
      annotations:
        tf-version.cloud-tpus.google.com: "2.7.0"
      labels:
        app: nerf-2
    spec:
      nodeSelector:
        cloud.google.com/gke-nodepool: e2-custom
      containers:
      - name: nerf-2
        image: gcr.io/spe3d-331118/spe3d
        command: ["/bin/bash", "-c"]
        args:
        - >
          mkdir tmp/models &&
          git -C jaxnerf/ pull origin kubectl &&
          echo $KUBE_GOOGLE_CLOUD_TPU_ENDPOINTS &&
          export TPU_MESH_CONTROLLER_ADDRESS=($(
             KUBE_GOOGLE_CLOUD_TPU_ENDPOINTS//:/)) &&
          export TPU_MESH_CONTROLLER_ADDRESS=($(
             TPU_MESH_CONTROLLER_ADDRESS//grpc\/\//)) &&
          export TPU_MESH_CONTROLLER_ADDRESS=($(
             TPU_MESH_CONTROLLER_ADDRESS//8470/)) &&
          export TPU_MESH_CONTROLLER_ADDRESS=${
             TPU_MESH_CONTROLLER_ADDRESS[0]}:8476 &&
          export COLAB_TPU_ADDR=($(
             KUBE_GOOGLE_CLOUD_TPU_ENDPOINTS//grpc:\/\//)) &&
          echo \$TPU_MESH_CONTROLLER_ADDRESS &&
          python -m jaxnerf.app
        resources:
          limits:
```

```
        cloud-tpus.google.com/v2: 8
    ports:
    - containerPort: 30000
```

Dentro de los argumentos se describen comandos que realizan una actualización del código del Github, así como la configuración de la variable COLAB_TPU_ADDR. Extrayendo la dirección IP del TPU desde la variable KUBE_GOOGLE_CLOUD_TPU_ENDPOINTS, una variable creada por GKE para ayudar a identificar a los Nodos que TPU está designada para su uso, luego se realiza la ejecución de la aplicación.

```
apiVersion: v1
kind: Service
metadata:
  name: nerf-2
spec:
  type: NodePort
  selector:
    app: nerf-2
  ports:
  - name: nerf-2
    port: 3000
    targetPort: 3000
    nodePort: 30300
```

Esta parte del manifiesto hace referencia al Service y los puertos en los que estará abierto para su comunicación. Finalmente con esta configuración se puede realizar una configuración escalable modificando el número de réplicas en la primera parte del manifiesto.

4.2 Captura y Pre-Procesamiento

La captura de las imágenes se realizó desde un Redmi Note 10S, con una cámara frontal de 64MP, los patrimonios culturales elegidos son: Una plancha antigua, un lienzo del lago Titicaca y un Monolito, el preprocesamiento se realizó en un entorno local, descrito en la Tabla 4 con LLFF, uno de los resultados del preprocesamiento se muestra en la Figura 26.

Figura 26: Pre-Procesamiento con LLFF

Elaboración propia

4.3 TPU VM

Para determinar los perfiles de VM, se evaluó el rendimiento del SPRN en una TPU VM con las características del Host de la TPU, las características se describen en la Tabla 4, se determinó un promedio de recursos tanto de CPU como de memoria RAM que serán monitoreados por el servicio de Monitoring, para encontrar el porcentaje de CPU en unidades y memoria RAM en Gigabyte (GB) para los tres tipos de dataset y sus respectivas redimensiones del experimento. La finalidad es someter las VM a diferentes tipos de carga de memoria para cada peso de archivo del dataset y sus redimensiones, para lo cual se instanciaron 5 TPU VM v2-8, registrando el tiempo y los recursos utilizados para los 9 modelos.

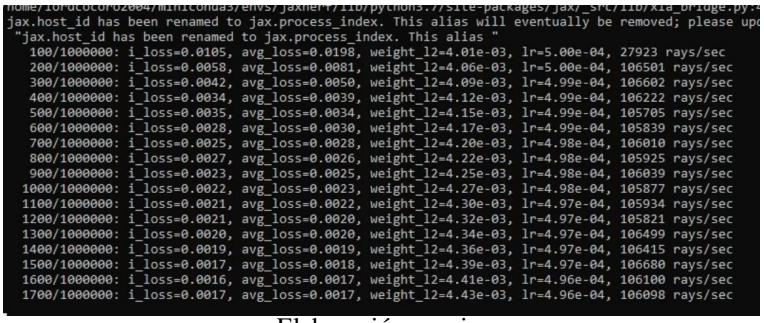
Los resultados de la experimentación muestran el rendimiento de la TPU VM, considerando los tres modelos entrenados en 100 mil iteraciones, como se visualiza en la Tabla 5 y en la Figura 27, donde se indica el porcentaje de CPU y memoria RAM.

Tabla 5: Rendimiento de la TPU VM v2-8

Modelo	Redimensión	CPU	RAM	Tiempo
A	100 %	5,01 % - 4,00 %	150,00 GB	1,60 h
B		3,54 % - 2,58 %	106,00 GB	1,65 h
C		5,45 % - 2,49 %	80,00 GB	1,50 h
A	50 %	5,64 % - 2,63 %	56,14 GB	1,25 h
B		3,91 % - 1,75 %	42,05 GB	1,20 h
C		5,15 % - 2,18 %	32,66 GB	1,20 h
A	25 %	8,54 % - 1,34 %	19,21 GB	1,02 h
B		3,56 % - 2,00 %	15,40 GB	1,01 h
C		2,57 % - 1,05 %	12,10 GB	0,98 h

Elaboración propia

Figura 27: Entrenamiento



Elaboración propia

La memoria RAM fue determinada por el tamaño de imágenes utilizadas durante el entrenamiento, considerando que entre más grandes sean las dimensiones, mayor será la cantidad de memoria utilizada, realizándose pruebas para diferentes redimensiones tales como: 100 %, 50 % y 25 %.

El análisis de estos datos permitió la creación de dos diferentes perfiles de VM para su uso posterior en TPU Nodo y TPU Nodo con GKE. Su construcción consideró el mínimo de CPU y RAM utilizados en el experimento anterior y el máximo de CPU registrado (Google, 2022a, 2022b). La configuración se muestra en la Tabla 4 donde se registra un máximo de 8 CPUs y un mínimo de un CPU utilizados por el SPRN en las TPUs VM. Se utilizaron VM con los perfiles n1-highmem-8 y e2-highmem-4 predeterminados por

el servicio Compute Engine del GPC, que priorizan la memoria RAM al ser el recurso de mayor consumo, que pertenecen a los tipos de VM N1 y E2 respectivamente (Google, 2022c).

4.4 TPU Nodo

Para evaluar el rendimiento de este diseño se usaron las TPUs Nodo, considerando los perfiles mencionados en el experimento anterior. Obteniendo que las redimensiones al 100 % superaban la memoria RAM, no pudieron continuar con el entrenamiento. El resultado de los experimentos restantes con las redimensiones al 50 % y 25 % se ilustran en la Tabla 6, mostrando el tiempo de ejecución y el consumo de recursos para 100 mil iteraciones, se observa que el tiempo se duplica y que el consumo de memoria RAM disminuye respecto al perfil seleccionado, el porcentaje de CPU fue mayor debido a que la capacidad de los perfiles N1 y E2 son mucho menores en comparación del Host del TPU VM, en la figura 31 se observa que el consumo de CPU no varía mucho.

4.5 TPU Nodo con GKE

Se experimentó con el perfil mínimo de la Tabla 4, para no exceder la cuota GCP y al igual que el diseño anterior, se evaluó el rendimiento de las VM utilizadas. El resultado fue muy similar al anterior experimento, los tiempos de los modelos se incrementaron ligeramente, con la gran diferencia que la tecnología GKE permitió crear varias réplicas, como se muestra en la Tabla 6, también se logra apreciar diferencias con el anterior resultado, sin embargo, el GKE fue capaz de escalar y orquestar los nodos VMs y TPUs instanciadas, maximizando el consumo de recursos en la nube.

Tabla 6: Rendimiento de las arquitecturas TPU VM, TPU Nodo y TPU Nodo con GKE

Diseño de TPU VM				
Modelo	**Perfil / Redimensión**	**CPU**	**RAM**	**Tiempo**
A	Host TPU VM 100 %	5,01 %	150,00 GB	1,60 h
B		3,54 %	106,00 GB	1,65 h
C		5,45 %	80,00 GB	1,50 h
A	Host TPU VM 50 %	5,64 %	56,14 GB	1,25 h
B		3,91 %	42,05 GB	1,20 h
C		5,15 %	32,66 GB	1,20 h
A	Host TPU VM 25 %	8,54 %	19,21 GB	1,02 h
B		3,56 %	15,40 GB	1,01 h
C		2,57 %	12,10 GB	0,98 h
Diseño TPU Nodo				
Modelo	**Perfil / Redimensión**	**CPU**	**RAM**	**Tiempo**
A	n1-high-mem-8 50 %	54,22 %	25,83 GB	2,60 h
B		53,08 %	19,02 GB	2,45 h
C		50,69 %	13,26 GB	2,50 h
A	n1-high-mem-8 25 %	53,27 %	7,10 GB	2,41 h
B		57,08 %	5,40 GB	2,31 h
C		46,50 %	3,76 GB	2,40 h
A	e2-high-mem-4 50 %	61,85 %	24,78 GB	2,63 h
B		62,51 %	19,45 GB	2,34 h
C		61,20 %	14,04 GB	2,45 h
A	e2-high-mem-4 25 %	57,20 %	7,20 GB	2,31 h
B		54,28 %	5,50 GB	1,98 h
C		52,90 %	3,75 GB	2,10 h
Diseño de TPU Nodo con GKE				
Modelo	**Perfil / Redimensión**	**CPU**	**RAM**	**Tiempo**
A	n1-high-mem-8 50 %	-	-	-
B		52,93 %	18,90 GB	2,31 h
C		53,23 %	13,20 GB	2,83 h
A	n1-high-mem-8 25 %	55,74 %	6,95 GB	2,65 h
B		54,22 %	5,45 GB	2,35 h
C		56,05 %	3,82 GB	2,00 h
A	e2-high-mem-4 25 %	62,56 %	7,25 GB	2,20 h
B		63,35 %	5,45 GB	2,08 h
C		60,14 %	3,76 GB	2,01 h

Elaboración propia

En la Tabla 7 podemos observar el tiempo y precios de la producción del SPRN con las diferentes arquitecturas con los perfiles y tipos de VM, los promedios del tiempo de producción de las arquitecturas TPU Nodo y TPU Nodo con GKE aumentan en más de un 100 % en comparación con la arquitectura TPU VM, al igual que los precios, siendo más económico la arquitectura TPU VM, con un mejor precio y mejor tiempo de producción.

Tabla 7: Promedio de porcentajes del tiempo y costos de los diseños TPU VM, TPU Nodo y TPU Nodo con GKE

Modelo	Redimensión	TIEMPO				
		TPU VM	N1 VM	E2 VM	N1 GKE	E2 GKE
A	50 %	1,25 h	2,60 h	2,63 h	-	-
B		1,20 h	2,45 h	2,34 h	2,31 h	-
C		1,20 h	2,50 h	2,45 h	2,83 h	-
A	25 %	1,02 h	2,41 h	2,31 h	2,65 h	2,20 h
B		1,01 h	2,31 h	1,98 h	2,35 h	2,08 h
C		0,98 h	2,40 h	2,10 h	2,00 h	2,01 h
Precio Promedio x Modelo con TRC		0,00$	1,06$	0,55$	1,03$	0,49$
Precio Promedio + TPU x Modelo		5,00$	12,00$	11,48$	11,60$	10,27$
Promedio de producción de modelos con TRC en 30 días		648,65	294,48	312,82	296,54	343,40
Precio promedio total		0$	312,14$	172,04$	305,43$	168,26$
Promedio		1,11 h	2,45 h	2,30 h	2,42 h	2,10 h
Promedio en %		100,00 %	220,27 %	207,35 %	218,73 %	188,88 %

Elaboración propia

4.6 Evaluación de calidad de imagen

Se realizaron 09 pruebas de calidad de imagen PSNR y SSIM para 100,000 iteraciones, los resultados son mostrados en la Tabla 8. La diferencia entre calidad por experimento es casi invariable, sin embargo, la calidad de imagen producida por las TPU VM con el 100 % de resolución son superiores al resto de las otras comparaciones, cabe resaltar que la métrica PSNR entre más alta sea es mejor al igual que la métrica SSIM que

entre más se acerque a 1 es mejor, los resultados de la Tabla 8 muestran valores similares al del artículo de Mildenhall *et al*. (2020b).

Tabla 8: Tabla de evaluación PSNR y SSIM

	1 eval		2 eval		3 eval	
A	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**
100 %	22,2859	0,8356	28,4324	0,8987	25,8827	0,9105
50 %	21,1761	0,8372	27,7312	0,9048	24,2001	0,9125
25 %	21,8137	0,8289	28,052	0,8886	25,1922	0,9041
B	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**
100 %	27,1873	0,8889	31,9832	0,8862	31,8018	0,8783
50 %	27,4282	0,9326	32,141	0,9335	32,4197	0,9243
25 %	27,2346	0,9266	32,3026	0,9398	32,3383	0,9275
C	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**
100 %	27,5944	0,7317	27,8228	0,795	26,1733	0,839
50 %	28,2142	0,7373	28,4637	0,865	25,981	0,8345
25 %	28,3682	0,7354	28,475	0,8461	26,4488	0,8585

Elaboración propia

4.7 Renderizado de los modelos

En esta sección se incluirán tres grupos de imágenes renderizadas por el SPRN de los modelos entrenados, estas se muestran en las siguientes Figuras 28, 29 y 30, estas imágenes son hiperrealistas e inferidas por la red neuronal desde puntos de vista de cámara no antes vistos.

4.8 Consideraciones Finales

Las Figuras 31, 32 y 33 hacen referencia gráfica a las Tablas 5 y 6, se puede apreciar que la arquitectura TPU VM cuenta con un mayor consumo de memoria RAM que decae respecto a la redimensión del grupo de imágenes, y una reducción en el tiempo de producción, sin embargo, el uso de memoria RAM también se reduce respecto a la capacidad del perfil VM utilizado, entre menos capacidad tenga una VM, menor también

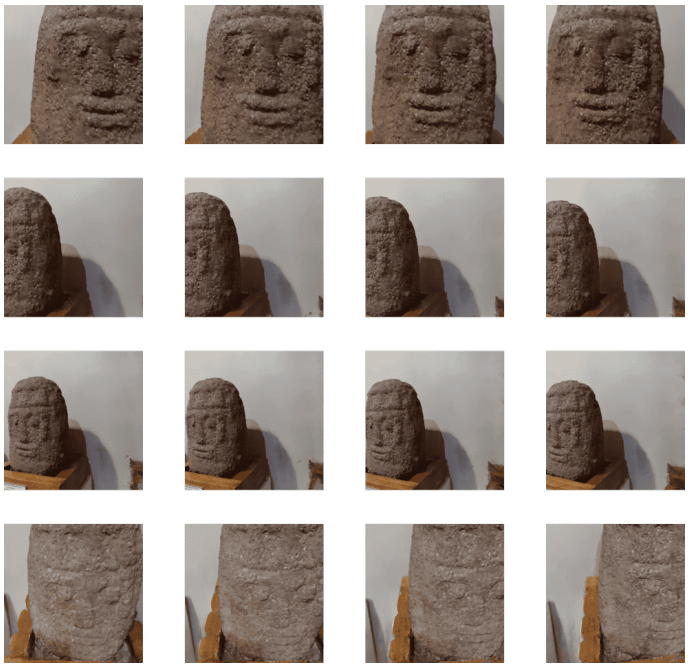
Figura 28: Plancha Renderizada con el SPRN

Elaboración propia

Figura 29: Lienzo Renderizado con el SPRN

Elaboración propia

Figura 30: Monolito Renderizado con el SPRN



Elaboración propia

será su consumo. El consumo de CPU varia ligeramente por el tipo de perfil y modelos utilizados, sin embargo, no existe una relación respecto a su rendimiento. El tiempo si está estrechamente relacionado con los modelos, sus redimensiones y capacidad de la VM.

Las tres arquitecturas presentadas son funcionales gracias a estos recursos, esto nos permite evaluar el rendimiento de un proyecto en VM de alta capacidad así como el entrenamiento de modelos en recursos TPU en un menor tiempo, el prototipo logra un alto rendimiento usando la arquitectura TPU VM, las características del Host permite lograr un mejor tiempo y rendimiento que el resto de las arquitecturas, teniendo una producción de 1960 modelos en 8 días con 10 instancias TPU, como se muestra en la Tabla 9, sin embargo, requiere una configuración manual, esta puede ser reemplazada por un script para la consola del GCP pero no se podría tener el control de las instancias por si algo falla o si se requieren cantidades como 100 nodos, para ello se utilizó GKE, el único problema es que la tecnología GKE no permite instanciar nodo y pods dentro de los Host de la TPU,

por lo cual se deben instanciar las TPU Nodo enlazadas a VMs.

La escalabilidad de la tecnología GKE puede llegar a utilizar un número mayor de TPU nodos, donde se compara la producción considerando la cantidad de TPUs brindados por TRC y el crédito otorgado por GCP al proyecto, que permiten agregar nuevas herramientas al SPRN como las mencionadas en el trabajo de Dellaert y Yen-Chen (2020), se realizó la comparación de producción de TPU VM en los tiempos máximos disponibles para el diseño GKE, también se consideró el límite de uso de TPUs, que son 30 días para cambiar el estimado del ”tiempo e2” de 1250 h a 720 h.

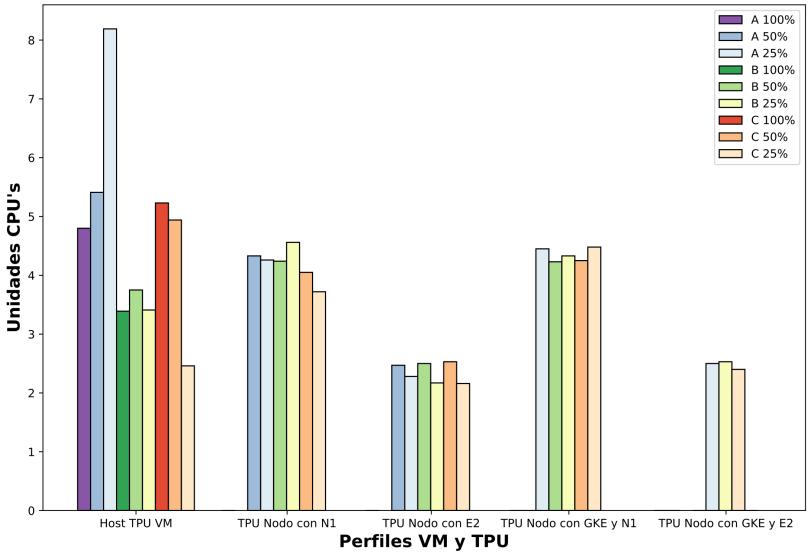
El diseño GKE produce una cantidad menor de modelos, reduciendo el número hasta en un 50,95 % y 56,58 % en el diseño TPU VM de los perfiles e2-high-mem-4 y n1-high-mem-8 respectivamente, por tal razón se estima que el diseño GKE es escalable, pues es capaz de usar en su totalidad la cantidad de TPUs disponibles, considerando el costo de acuerdo a la cantidad de Nodos utilizados, con una producción de 1026,55 modelos en 21,35h dando un estimado de 1440 modelos por día, como se muestra en la Tabla 9.

Tabla 9: Comparación TPU VM - diseño GKE

	1 Nodo	**5 Nodos**	**10 Nodos**	**100 Nodos**
Costo de e2	0,24$	0,80$	1,50$	14,05$
Costo de n1	0,44$	1,46$	2,81$	27,25$
Tiempo e2	720,00 h	375,00 h	200,00 h	21,35 h
Tiempo n1	681,81 h	205,47 h	106,76 h	11,01 h
Modelos e2	346,15	901,44	961,53	1026,55
Modelos n1	290,13	437,19	454,31	468,47
TPU VM en 720h	705,60	1837,50	1960,00	-
TPU VM en 681h	671,25	1006,80	1046,24	-
e2 vs TPU %	50,95 %	50,95 %	50,95 %	-
n1 vs TPU %	56,58 %	56,58 %	56,58 %	-

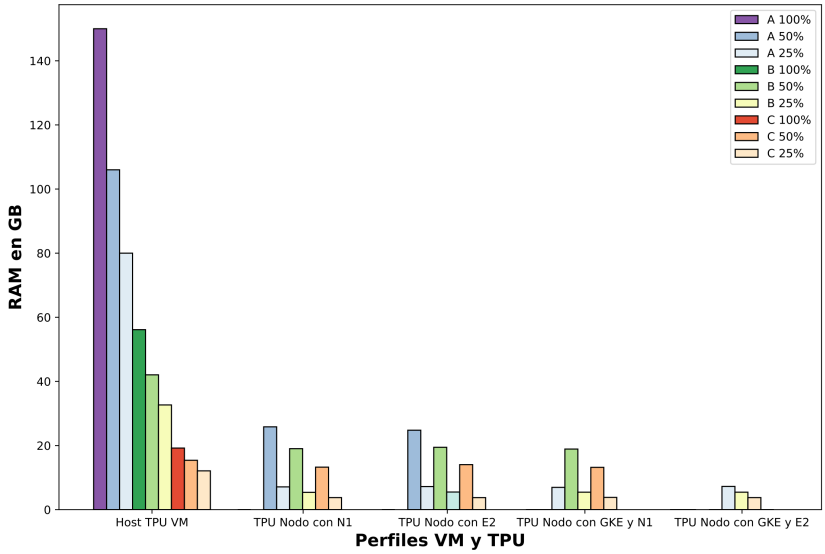
Elaboración propia

Figura 31: Rendimiento del CPU



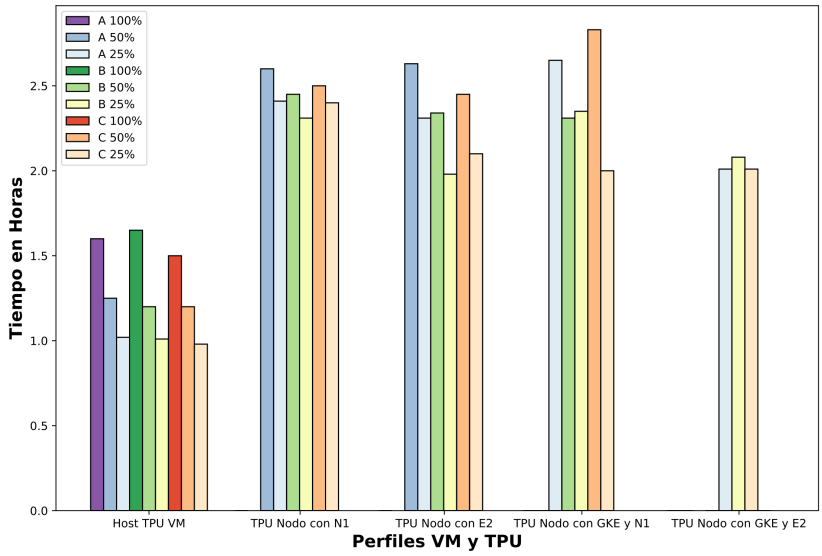
Elaboración propia

Figura 32: Consumo de la RAM



Elaboración propia

Figura 33: Tiempo de entrenamiento



Elaboración propia

4.9 Discusiones

Los resultados de la integración y adaptación de JAXNERF con el SPRN ayudaron a validar el trabajo de Deng, Barron y Srinivasan (2020) publicado en Github, así mismo se logró comprobar que las TPUs mejoran significativamente el tiempo de entrenamiento respecto al trabajo original de Mildenhall *et al*. (2020b), SPRN logra mejorar la cantidad de producción utilizando GKE.

La metodología de desarrollo evolutivo presentado por Sommerville (2011) para el desarrollo del SPRN fue efectiva y adecuada para establecer los perfiles VM, el incremento en cada fase del desarrollo permitió realizar las pruebas de rendimiento en cada arquitectura que forma parte de una mejora, siguiendo las directrices de Basil y Turner (1975).

El uso de recursos bajo demanda como los TPU y servicios en la nube como el GCP son una oportunidad para la realización de proyectos de alta escalabilidad como

se muestró en esta tesis y el proyecto presentado por Golubovic y Rocha (2021), que valida el uso de TPU para proyectos de ML e IA, haciendo accesible el uso de tecnologías como NeRF en centros de investigación como el CERN e instituciones culturales como los museos.

Esta investigación supone un aporte al campo de renderización, reconstrucción 3D y cultural, brindando tres arquitecturas funcionales utilizando el servicio GKE con TPU con recursos de uso libre para la investigación, produciendo modelos hiperrealistas de patrimonios culturales, asimismo dentro del área de ingeniería Computacional y Sistemas, que ofrece una introducción al uso de recursos TPU con GKE para Aprendizaje Automático, entrenamiento de Redes Neuronales y otros campos de Inteligencia Artificial.

V. CONCLUSIONES

Los resultados muestran que el prototipo es funcional y capaz de usar las TPUs de forma escalable en perfiles mínimos con un costo menor de CPU y memoria RAM usando el servicio de GKE. La adaptación e integración de JAXNERF en el SPRN es compatible con todas las arquitecturas y es escalable con la arquitectura TPU Nodo con GKE, el uso de la API/REST provee un servicio de Renderización Neural a través de los puertos expuestos y es capaz de sincronizar el SPRN con las TPU.

El rendimiento de la SPRN en la TPU VM fue fundamental para establecer los perfiles VM mínimos para las arquitecturas TPU Nodo y TPU Nodo con GKE, monitoreando el rendimiento y uso de recursos computacionales como el CPU y RAM, en la cual se prioriza elevar la capacidad de la RAM respecto a la capacidad mínima de CPUs, reduciendo significativamente los recursos en comparación con el Host del TPU VM.

La evaluación del rendimiento del SPRN en las VM con las Arquitecturas TPU VM, TPU Nodo y TPU con GKE, permitieron determinar la capacidad de producción de cada una, siendo las TPU VM con mayor capacidad de producción pero con menor escalabilidad, la TPU Nodo cuenta con una menor capacidad de producción y con un control medio de escalabilidad y finalmente la TPU Nodo con GKE, cuentan con una capacidad menor de producción y con un mayor control de escalabilidad.

La reducción de recursos CPU y RAM para las arquitecturas TPU Nodo y TPU Nodo con GKE amplía significativamente el tiempo y capacidad de producción de modelos digitales neurales, incrementando el tiempo hasta en un 120.27 %, gastando un promedio de hasta 1,06$ por modelo del crédito de 300$ que otorga el free trial de GCP, lo que implica un límite de uso de crédito disponible para la producción de modelos.

TPU VM produce modelos a costo nulo pues no utiliza crédito del GCP, sin embargo, a pesar de contar con 100 TPU interrumpibles, estas deben configurarse de forma manual, a diferencia del diseño GKE que es capaz de restablecer el servicio de forma automática. Si se desea realizar una renderización neural a baja escala, entre 600 y 3000 modelos por mes, el diseño TPU VM es la opción más viable.

Por otro lado si se requiere una renderización mayor, en paralelo, se puede optar por el diseño GKE que con 100 nodos es capaz de producir 1026 modelos en 21 horas utilizando al máximo el crédito free trial del GCP.

5.1 Trabajos futuros

- La aplicación e integración de diferentes implementaciones de NeRF tiene el suficiente potencial para revolucionar la digitalización de patrimonios culturales, ampliando el espectro de digitalización como danzas, flora, fauna, estructuras arquitectónicas, etc.

- La aplicación de esta tecnología abre nuevos caminos para la creación de contenido digital, estos modelos neurales pueden almacenar escenarios completos de 3D en un espacio reducido de memoria, sin embargo su tiempo de entrenamiento es relativamente costoso aun utilizando recursos TPU y servicios como el GKE, mejorar el rendimiento del entrenamiento es clave para poder introducir de forma sustentable estos modelos a medios de comunicación.

VI. RECOMENDACIONES

- Para el uso de arquitecturas con GKE, se recomienda ampliar la implementación utilizando Kubeflow para tener un mejor control del proceso de escalamiento, sin embargo, se debe realizar una configuración apropiada para integrar JAXNERF con las TPU.

- Para el despliegue con GKE se recomienda utilizar la mitad de la capacidad de la cuota del GCP y de los TPU interrumpibles, para evitar el consumo excesivo de recursos o la superación de la cuota.

- Para la evaluación de rendimiento se recomienda utilizar los servicios de Monitoring del GCP, sin embargo si se requiere un monitoreo de la RAM puede utilizar librerías para la consulta de procesos y subprocesos

- Se recomienda replicar este experimento con fines académicos y culturales, con el fin de promover una digitalización sustentable y responsable.

- Finalmente se recomienda que si se desea replicar los experimentos, la captura de imágenes sea en un espacio controlado, sin una excesiva variación de luz y con un mismo dispositivo de forma secuencial.

VII. REVISIÓN BIBLIOGRÁFICA

Apablaza, F. P. (2021). Transformación digital y museos sin fronteras. Una evaluación de siete museos de la ciudad de Lima en tiempos de pandemia. *Revista de Investigaciones de la Universidad Le Cordon Bleu*, *8*(1), 104-119. https://doi.org/10.36955/RIULCB.2021V8N1.010

Avidan, S., & Shashua, A. (1997). Novel view synthesis in tensor space. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1034-1040. https://doi.org/10.1109/CVPR.1997.609457

Basil, V. R., & Turner, A. J. (1975). Iterative Enhancement: A Practical Technique for Software Development. *IEEE Transactions on Software Engineering*, *SE-1*(4), 390-396. https://doi.org/10.1109/TSE.1975.6312870

Batchelor, D., Schnabel, M. A., & Dudding, M. (2021). Smart Heritage: Defining the Discourse. *Heritage 2021, Vol. 4, Pages 1005-1015*, *4*, 1005-1015. https://doi.org/10.3390/HERITAGE4020055

Bisong, E. (2019). An overview of google cloud platform services. *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, 7-10.

Burns, B., Beda, J., & Hightower, K. (2018). *Kubernetes*. Dpunkt Heidelberg, Germany.

Canales, F. H., Alvarado, E. L., & Pineda, E. B. (1994). Metodologia de la investigacion. Manual para el desarrollo de personal de salud. *Metodología de la investigación*, 232.

Caro, J. L., Luque Gil, A. M., & Zayas Fernández, B. (2014). Aplicaciones tecnológicas para la promoción de los recursos turísticos culturales. *Tecnologías de la información para nuevas formas de ver el territorio*, 938-946. http://hdl.handle.net/10045/46827

Champion, E., & Rahaman, H. (2019). 3D Digital Heritage Models as Sustainable Scholarly Resources. *Sustainability 2019, Vol. 11, Page 2425*, *11*(8), 2425. https://doi.org/10.3390/SU11082425

COLMAP. (2016). COLMAP — COLMAP 3.7 documentation. Consultado el 18 de octubre de 2021, desde https://colmap.github.io/index.html

Copeland, R. (2008). *Essential sqlalchemy*. .º'Reilly Media, Inc."

Cox, C., Sun, D., Tarn, E., Singh, A., Kelkar, R., & Goodwin, D. (2020). Serverless inferencing on Kubernetes. https://doi.org/10.48550/arxiv.2007.07366

Crowley, J. L. (2012). Intelligent Systems: Reasoning and Recognition. *training*, *1000*, 1.

Dellaert, F. (2020). NeRF Explosion 2020. *https://dellaert.github.io/NeRF/*.

Dellaert, F., & Yen-Chen, L. (2020). Neural Volume Rendering: NeRF And Beyond. https://arxiv.org/abs/2101.05204v2

Deng, B., Barron, J. T., & Srinivasan, P. (2020). JaxNeRF: an efficient JAX implementation of NeRF. *https://github.com/google-research/google-research/tree/master/jaxnerf*.

Deng, B., Barron, J. T., & Srinivasan, P. P. (2020). *JaxNeRF: an efficient JAX implementation of NeRF* (Ver. 0.0). https://github.com/google-research/google-research/tree/master/jaxnerf

Dkube. (2021). Create an Enterprise-Ready MLOps Platform Using Kubeflow. Consultado el 6 de agosto de 2022, desde https://www.dkube.io/blog/2021-04-07/

Eslami, A. (2018). Generative Query Networks - YouTube. Consultado el 6 de agosto de 2022, desde https://www.youtube.com/watch?v=RBJFngN33Qo

Flores, L. A. (2015). *AVANCE DE INVESTIGACIÓN MAPEO FOTOGRAMÉTRICO EN ARQUEOLOGÍA : CUENCA NORTE DEL TITICACA , PERÚ Photogrammetric Mapping in Archaeology : Experiences from the Ramis Project , Northern Titicaca Basin , Peru* (Tesis doctoral). Universidad Nacional Federico Villarreal.

Gao, C., Saraf, A., Kopf, J., & Huang, J. B. (2021). Dynamic View Synthesis from Dynamic Monocular Video. *Proceedings of the IEEE International Conference on Computer Vision*, 5692-5701. https://doi.org/10.48550/arxiv.2105.06468

Garbin, S. J., Kowalski, M., Johnson, M., Shotton, J., & Valentin, J. (2021). FastNeRF: High-Fidelity Neural Rendering at 200FPS. https://arxiv.org/abs/2103.10380v2

George, J., & Saha, A. (2022). End-to-end Machine Learning using Kubeflow. *ACM International Conference Proceeding Series*, 336-338. https://doi.org/10.1145/3493700.3493768

Golubovic, D., & Rocha, R. (2021). Training and Serving ML workloads with Kubeflow at CERN. *EPJ Web of Conferences*, *251*, 02067. https://doi.org/10.1051/EPJCONF/202125102067

Gonzalez-Diaz, R., Stelldinger, P., & Latecki, L. J. (2020). Digitization. *Computer Vision*, 1-8. https://doi.org/10.1007/978-3-030-03243-2_645-1

Google. (2022a). Cloud TPU documentation | Google Cloud. Consultado el 13 de abril de 2022, desde https://cloud.google.com/tpu/docs/

Google. (2022b). Pricing | Cloud TPU | Google Cloud. https://cloud.google.com/tpu/pricing

Google. (2022c). Pricing | Compute Engine: Virtual Machines (VMs) | Google Cloud. Consultado el 17 de abril de 2022, desde https://cloud.google.com/compute/all-pricing

Google. (2022d). System Architecture | Cloud TPU | Google Cloud. Consultado el 13 de abril de 2022, desde https://cloud.google.com/tpu/docs/system-architecture-tpu-vm

Gundi, N. D., Shabanian, T., Basu, P., Pandey, P., Roy, S., Chakraborty, K., & Zhang, Z. (2020). EFFORT: Enhancing Energy Efficiency and Error Resilience of a Near-Threshold Tensor Processing Unit. *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, *2020-January*, 241-246. https://doi.org/10.1109/ASP-DAC47756.2020.9045479

He, Y., Ma, Y. H., & Zhang, X. R. (2017). Digital heritage"theory and innovative practice. *International Archives of the Photogrammetry, Remote Sensing and Spatial Infor-*

mation Sciences - ISPRS Archives, *42*(2W5), 335-342. https://doi.org/10.5194/ISPRS-ARCHIVES-XLII-2-W5-335-2017

Herrera, A. M., Cuadros-Vargas, A. J., & Pedrini, H. (2019). Improving semantic segmentation of 3D medical images on 3D convolutional neural networks. *Proceedings - 2019 45th Latin American Computing Conference, CLEI 2019*. https://doi.org/10.1109/CLEI47609.2019.235102

Horé, A., & Ziou, D. (2010). Image quality metrics: PSNR vs. SSIM. *Proceedings - International Conference on Pattern Recognition*, 2366-2369. https://doi.org/10.1109/ICPR.2010.579

Ifrah, S. (2021). Get Started with Google Cloud Platform (GCP). *Getting Started with Containers in Google Cloud Platform*, 1-37. https://doi.org/10.1007/978-1-4842-6470-6_1

Inc., A. (2022). conda-docs/miniconda.rst at master · conda/conda-docs. https://github.com/conda/conda-docs/blob/master/docs/source/miniconda.rst

Jouppi, N., Young, C., Patil, N., & Patterson, D. (2018). Motivation for and Evaluation of the First Tensor Processing Unit. *IEEE Micro*, *38*(3), 10-19. https://doi.org/10.1109/MM.2018.032271057

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P. L., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., ... Yoon, D. H. (2017). In-datacenter performance analysis of a tensor processing unit. *Proceedings - International Symposium on Computer Architecture*, *Part F128643*, 1-12. https://doi.org/10.1145/3079856.3080246

Kanso, A., Palencia, E., Patra, K., Shan, J., Chao, M., Wei, X., Cai, T., Chen, K., & Qiao, S. (2021). Designing a kubernetes operator for machine learning applications. *WoC 2021 - Proceedings of the 2021 7th International Workshop on Container Technologies and Container Clouds*, 7-12. https://doi.org/10.1145/3493649.3493654

Kohli, A. P. S., Sitzmann, V., & Wetzstein, G. (2020). Semantic Implicit Neural Scene Representations with Semi-Supervised Training. *Proceedings - 2020 International Conference on 3D Vision, 3DV 2020*, 423-433. https://doi.org/10.1109/3DV50981.2020.00052

Lambers, K., Eisenbeiss, H., Sauerbier, M., Kupferschmidt, D., Gaisecker, T., Sotoodeh, S., & Hanusch, T. (2007). Combining photogrammetry and laser scanning for the recording and modelling of the Late Intermediate Period site of Pinchango Alto, Palpa, Peru. *Journal of Archaeological Science*, *34*(10), 1702-1712. https://doi.org/10.1016/J.JAS.2006.12.008

Li, T., Slavcheva, M., Zollhoefer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Newcombe, R., & Lv, Z. (2021). Neural 3D Video Synthesis from Multi-view Video. https://doi.org/10.48550/arxiv.2103.02597

Li, Z., Niklaus, S., Snavely, N., & Wang, O. (2020). Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. http://arxiv.org/abs/2011.13084

Liu, L., Gu, J., Lin, K. Z., Chua, T.-S., & Theobalt, C. (2020). Neural sparse voxel fields. *Advances in Neural Information Processing Systems (NeurIPS)*, *33*.

Manovich, L. (2021). Database as Symbolic Form. *Museums in a Digital Age*, 79-86. https://doi.org/10.4324/9780203716083-15/DATABASE-SYMBOLIC-FORM-LEV-MANOVICH

Martin-Brualla, R., Radwan, N., Sajjadi, M. S. M., Barron, J. T., Dosovitskiy, A., & Duckworth, D. (2020). NeRF in the wild: Neural radiance fields for unconstrained photo collections. *arXiv*. https://arxiv.org/abs/2008.02268v3

Mildenhall, B., Srinivasan, P. P., Cayon, R. O., Kalantari, N. K., Ramamoorthi, R., Ng, R., & Kar, A. (2019). Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *CoRR*, *abs/1905.00889*. http://arxiv.org/abs/1905.00889

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020a). *bmild/nerf: Code release for NeRF (Neural Radiance Fields)*. Consultado el 6 de agosto de 2022, desde https://github.com/bmild/nerf

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020b). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12346 LNCS*, 405-421. https://doi.org/10.1007/978-3-030-58452-8_24

Norrie, T., Patil, N., Yoon, D. H., Kurian, G., Li, S., Laudon, J., Young, C., Jouppi, N., & Patterson, D. (2021). The Design Process for Google's Training Chips: TPUv2 and TPUv3. *IEEE Micro*, *41*(2), 56-63. https://doi.org/10.1109/MM.2021.3058217

Nvidia. (2016). Build and run Docker containers leveraging NVIDIA GPUs v2.11.0. Consultado el 6 de agosto de 2022, desde https://github.com/NVIDIA/nvidia-docker

Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D., Seitz, S., & Martin-Brualla, R. (2020). Deformable Neural Radiance Fields. *https://arxiv.org/abs/2011.12948*.

Parry, R. (2010). Museums in a digital age. *1*, 478. https://books.google.com/books/about/Museums_in_a_Digital_Age.html?hl=es%5C&id=-SBmGtwekr4C

Paul R. Wolf, P., Bon A. Dewitt, P., & Benjamin E. Wilkinson, P. (2014). Elements of Photogrammetry with Applications in GIS, Fourth Edition. https://www.accessengineeringlibrary.com/content/book/9780071761123%20https://www.accessengineeringlibrary.com/content/book/9780071761123.abstract

Pavlidis, G., Koutsoudis, A., Arnaoutoglou, F., Tsioukas, V., & Chamzas, C. (2007). Methods for 3D digitization of Cultural Heritage. *Journal of Cultural Heritage*, *8*(1), 93-98. https://doi.org/10.1016/J.CULHER.2006.10.007

Peng, S., Zhang, Y., Xu, Y., Wang, Q., Shuai, Q., Bao, H., & Zhou, X. (2020). Neural Body: Implicit Neural Representations with Structured Latent Codes for Novel View Synthesis of Dynamic Humans. http://arxiv.org/abs/2012.15838

Peng, S., Zhang, Y., Xu, Y., Wang, Q., Shuai, Q., Bao, H., & Zhou, X. (2021). Neural Body: Implicit Neural Representations with Structured Latent Codes for Novel View Synthesis of Dynamic Humans. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 9050-9059. https://doi.org/10.1109/CVPR46437.2021.00894

Pineda, F., Ayma, V., & Beltran, C. (2020). A GENERATIVE ADVERSARIAL NETWORK APPROACH FOR SUPER-RESOLUTION OF SENTINEL-2 SATELLITE IMAGES. https://doi.org/10.5194/isprs-archives-XLIII-B1-2020-9-2020

Pumarola, A., Corona, E., Pons-Moll, G., & Moreno-Noguer, F. (2020). D-NeRF: Neural Radiance Fields for Dynamic Scenes. *https://arxiv.org/abs/2011.13961*.

Rebain, D., Jiang, W., Yazdani, S., Li, K., Yi, K. M., & Tagliasacchi, A. (2020a). DeRF: Decomposed Radiance Fields. https://arxiv.org/abs/2011.12490v1

Rebain, D., Jiang, W., Yazdani, S., Li, K., Yi, K. M., & Tagliasacchi, A. (2020b). DeRF: Decomposed Radiance Fields. *https://arxiv.org/abs/2011.12490*.

RN, S., & J, M. (1971). Mental rotation of three-dimensional objects. *Science (New York, N.Y.)*, *171*(3972), 701-703. https://doi.org/10.1126/SCIENCE.171.3972.701

Sánchez Carlessi, H. H., Reyes Romero, C., & Mejía Sáenz, K. (2018). *Manual de términos en investigación científica, tecnológica y humanística*. http://repositorio.urp.edu.pe/bitstream/handle/URP/1480/libro-manual-de-terminos-en-investigacion.pdf?sequence=1%5C&isAllowed=y

Sara, U., Akter, M., Uddin, M. S., Sara, U., Akter, M., & Uddin, M. S. (2019). Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study. *Journal of Computer and Communications*, *7*, 8-18. https://doi.org/10.4236/JCC.2019.73002

Schoenberger, J. L. (2022). *COLMAP - Tutorial*. Consultado el 6 de agosto de 2022, desde https://colmap.github.io/tutorial.html

Sommerville, I. (2011). Software engineering, 9th edition. En L. M. Cruz Castillo (Ed.), *Software engineering, 9th edition* (9th, p. 792). Pearson Education, Inc.

Terras, M. (2015). Cultural Heritage Information: Artefacts and Digitization Technologies, 25. http://www.bodley.ox.ac.uk/ilej/

Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin-Brualla, R., Simon, T., Saragih, J., Nießner, M., Pandey, R., Fanello, S., Wetzstein, G., Zhu, J. Y., Theobalt, C., Agrawala, M., Shechtman, E., Goldman, D. B., & Zollhöfer, M. (2020). State of the Art on Neural Rendering. *Computer Graphics Forum*, *39*(2), 701-727. https://doi.org/10.1111/cgf.14022

Ticona Quispe, D. Y. (2021). *Universidad Nacional Del Altiplano* (Tesis doctoral). Universidad Nacional del Altiplano.

Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., & Theobalt, C. (2020). Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Deforming Scene from Monocular Video. *https://arxiv.org/abs/2012.12247*.

UNESCO. (2003). Charter on the Preservation of Digital Heritage: UNESCO. Consultado el 26 de abril de 2021, desde http://portal.unesco.org/en/ev.php-URL_ID=17721%5C&URL_DO=DO_TOPIC%5C&URL_SECTION=201.html

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.

Wu, C., & Song, M. (2020). An Automatic Artificial Intelligence Training Platform Based on Kubernetes. *Proceedings of the 2020 2nd International Conference on Big Data Engineering and Technology*. https://doi.org/10.1145/3378904

Wudenhe, A., & Tseng, H. W. (2021). TPUPoint: Automatic Characterization of Hardware-Accelerated Machine-Learning Behavior for Cloud Computing. *Proceedings - 2021 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2021*, 254-264. https://doi.org/10.1109/ISPASS51385.2021.00048

Xian, W., Huang, J.-B., Kopf, J., & Kim, C. (2020). Space-time Neural Irradiance Fields for Free-Viewpoint Video. *https://arxiv.org/abs/2011.12950*.

Yuan, W., Lv, Z., Schmidt, T., & Lovegrove, S. (2021). STaR: Self-supervised Tracking and Reconstruction of Rigid Objects in Motion with Neural Rendering. *arXiv preprint arXiv:2101.01602*.

Zhang, K., Riegler, G., Snavely, N., & Koltun, V. (2020). NERF++: Analyzing and Improving Neural Radiance Fields. *https://arxiv.org/abs/2010.07492*.

Zhou, T., Tulsiani, S., Sun, W., Malik, J., & Efros, A. A. (2016). View Synthesis by Appearance Flow. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9908 LNCS*, 286-301. https://doi.org/10.1007/978-3-319-46493-0_18