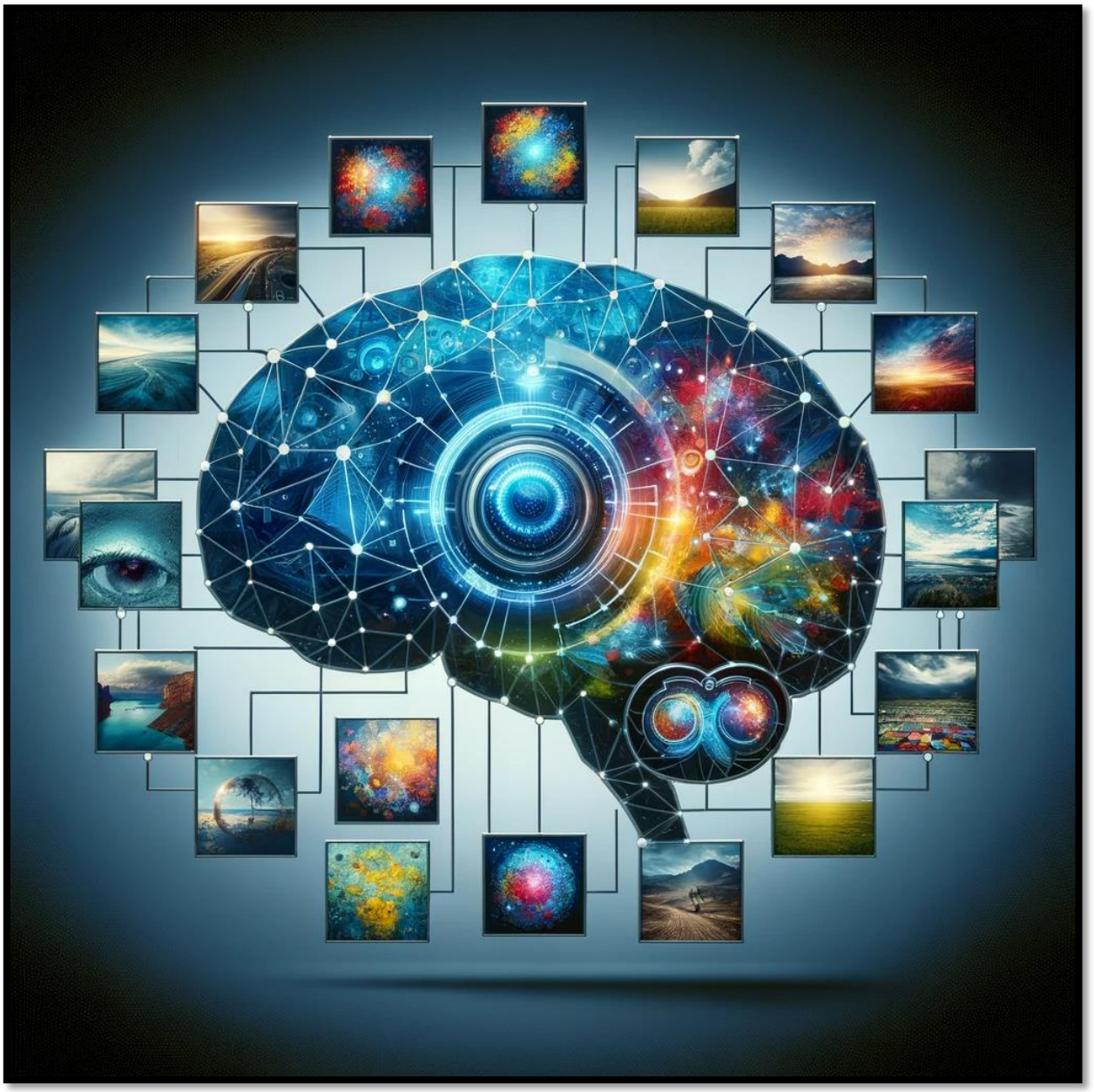
Sistema de Reconocimiento Facial para Predicción Género

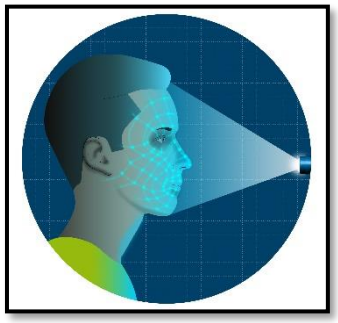


Nicolás Martín Díaz
Ian Meléndez Delgado

Índice

1. Introducción

En la era actual, la necesidad de soluciones avanzadas de reconocimiento facial ha adquirido una relevancia crucial. Este proyecto se sitúa en el contexto de una sociedad cada vez más digital, donde la identificación precisa de individuos desempeña un papel vital en numerosos ámbitos. Desde la seguridad y la detección de fraudes hasta la personalización de servicios, la capacidad de predecir la edad, etnia y género a partir de imágenes faciales se ha convertido en un requisito clave. La justificación de este proyecto radica en abordar estas demandas, proporcionando una herramienta eficiente y precisa basada en técnicas de machine learning utilizando Keras. A través de este enfoque, se busca contribuir a la mejora de sistemas de identificación, ofreciendo aplicaciones prácticas y beneficios a diversos sectores.



El proyecto se centra en la creación de un Sistema de Reconocimiento Facial mediante técnicas de machine learning empleando la biblioteca Keras. Su objetivo es predecir la edad, etnia y género de una persona a partir de imágenes faciales, con aplicaciones potenciales en seguridad, publicidad personalizada y estudios demográficos. Para lograrlo, se utilizará Visual Studio conectado a un servidor remoto mediante SSH, aprovechando la tarjeta gráfica A6000 para mejorar la eficiencia del entrenamiento del modelo.

Objetivos y usos del modelo

- Implementar un Modelo Preciso: Desarrollar un modelo de machine learning en Keras que pueda realizar predicciones precisas de edad, etnia y género a partir de imágenes faciales, mejorando la eficiencia y la confiabilidad de los sistemas de reconocimiento facial.

- Interfaz de Usuario Intuitiva: Diseñar e implementar una interfaz de usuario fácil de usar, que permita a los usuarios interactuar de manera eficiente con el sistema, facilitando la carga de imágenes y visualización de resultados.

- Utilización del Modelo en Escenarios Comerciales: La adaptabilidad y robustez del modelo de reconocimiento facial desarrollado con Keras permiten su implementación efectiva en diversos escenarios comerciales. Su capacidad para predecir con precisión la edad, etnia y género de individuos presenta oportunidades valiosas para la personalización de servicios, análisis demográfico y toma de decisiones estratégicas.

- Personalización de Servicios: El modelo puede ser empleado en entornos comerciales para personalizar servicios, adaptándolos a las características demográficas específicas de los clientes. Desde campañas de marketing dirigidas hasta recomendaciones de productos, el sistema puede mejorar la experiencia del usuario al proporcionar servicios más relevantes y personalizados.

- Seguridad y Acceso Autorizado: En el ámbito de la seguridad, la capacidad del

modelo para identificar de manera precisa características faciales brinda aplicaciones significativas. Puede ser utilizado para sistemas de control de acceso, autenticación segura y vigilancia, contribuyendo a la protección de instalaciones comerciales y garantizando la integridad del entorno.

- Análisis Demográfico para toma de Decisiones: La predicción precisa de edad, etnia y género puede ser valiosa para la toma de decisiones estratégicas. Desde la planificación de inventarios hasta la segmentación de clientes, el modelo puede proporcionar información demográfica útil que respalde decisiones empresariales informadas.

- Potencial de Comercialización del Modelo: El modelo de reconocimiento facial desarrollado con Keras posee un potencial significativo para la comercialización en diversas industrias y aplicaciones. Su capacidad para predecir con precisión la edad, etnia y género abre oportunidades estratégicas que pueden ser aprovechadas en diferentes sectores comerciales.

- Sistemas de Seguridad Avanzados: La implementación del modelo en sistemas de seguridad avanzados, como cámaras de vigilancia inteligentes y sistemas de control de acceso, ofrece una solución eficiente para la autenticación segura y la identificación precisa de individuos. Este enfoque innovador puede ser atractivo para empresas y organizaciones que buscan fortalecer sus medidas de seguridad.

- Desarrollo de Aplicaciones Personalizadas: La capacidad del modelo para personalizar servicios basados en la identificación de características faciales lo convierte en una herramienta valiosa para el desarrollo de aplicaciones personalizadas. Desde aplicaciones móviles hasta plataformas web, el modelo puede ser integrado para ofrecer experiencias de usuario altamente personalizadas.
- Industria del Retail y Marketing: En el sector minorista, el modelo puede ser utilizado para analizar el perfil demográfico de los clientes, permitiendo estrategias de marketing más efectivas y la personalización de ofertas. La capacidad de adaptar las estrategias comerciales a las preferencias de los clientes puede generar un impacto positivo en la fidelización y las ventas.

- Licenciamiento del Modelo a Terceros: El modelo puede ser licenciado a terceros interesados en integrar capacidades de reconocimiento facial en sus productos o servicios. Este enfoque ofrece oportunidades de ingresos a través de acuerdos de licenciamiento y colaboraciones estratégicas.

- Entorno de Desarrollo Optimizado: Configurar Visual Studio para el desarrollo del proyecto y establecer una conexión eficiente a un servidor remoto mediante SSH, asegurando un entorno de desarrollo robusto y colaborativo.

- Aprovechar la Potencia de la Tarjeta Gráfica A600: Utilizar la capacidad de procesamiento de la tarjeta gráfica A600 para acelerar el tiempo de entrenamiento del modelo, mejorando así su rendimiento y eficiencia.

Alcance y Limitaciones:

Este proyecto abarcará desde la conceptualización y diseño del modelo hasta la implementación de una interfaz de usuario funcional. Se priorizará la integración eficiente con Visual Studio y la explotación de la tarjeta gráfica A6000 para maximizar la eficiencia del sistema.

Se reconocen limitaciones inherentes, como la dependencia de la calidad de las imágenes de entrada y posibles sesgos en la predicción de etnias. Además, factores externos como condiciones ambientales y variabilidad en las imágenes pueden afectar la precisión del sistema. Estas limitaciones se documentarán y abordarán con transparencia para comprender las restricciones del sistema propuesto.

2. Descripción de técnicas de reconocimiento facial y sus librerías.

Librería OpenCV (cv2):

OpenCV es una biblioteca de código abierto ampliamente utilizada para realizar tareas de visión por computadora y procesamiento de imágenes en Python. Proporciona una amplia gama de funciones para cargar, procesar y manipular imágenes y videos. Sus características incluyen detección de bordes, detección de características, seguimiento de objetos y reconocimiento de formas.

Librería Face_recognition:

Desarrollada por Adam Geitgey, face_recognition es una biblioteca de Python que simplifica el proceso de reconocimiento facial. Utiliza la biblioteca dlib para extraer características faciales y realizar el reconocimiento. face_recognition ofrece una interfaz simple para el entrenamiento y reconocimiento de caras.

La combinación de OpenCV y face_recognition permite construir sistemas completos de reconocimiento facial en Python.

.

> *OpenCV se utiliza para tareas de preprocesamiento de imágenes, detección de rostros y manipulación de imágenes, mientras que face_recognition se encarga de identificar y reconocer caras en las imágenes.*

3. Herramientas y Tecnologías

Keras

Keras es una interfaz de alto nivel para la construcción y entrenamiento de modelos de redes neuronales, diseñada para ser modular, extensible y fácil de usar. A lo largo de los años, Keras se ha convertido en una de las bibliotecas de deep learning más populares debido a sus características y flexibilidad.

Aquí se destacan aspectos clave sobre el uso de Keras en el contexto del reconocimiento facial:

Keras proporciona una abstracción de nivel superior que facilita la creación de modelos de deep learning sin requerir un profundo conocimiento matemático. Esto permite a los desarrolladores centrarse en la arquitectura del modelo y los aspectos específicos del problema en lugar de detalles de implementación complejos.

Actúa como una interfaz de alto nivel y es compatible tanto con TensorFlow como con Theano, dos de las bibliotecas de cálculo numérico más utilizadas en el ámbito de deep learning. Esto brinda flexibilidad a los usuarios para elegir la biblioteca de backend según sus preferencias y requisitos.

La sintaxis simple y consistente de Keras facilita la definición de capas, la configuración de hiperparámetros y la compilación de modelos. La legibilidad del código permite a los desarrolladores experimentar con arquitecturas y ajustar parámetros de manera eficiente.

Ofrece dos modos principales para construir modelos: el modelo secuencial, que es lineal y apropiado para redes donde las capas se apilan una encima de la otra, y el modelo funcional, que permite construir modelos más complejos y no lineales mediante la conexión de capas de manera más flexible

Keras proporciona una amplia gama de capas predefinidas, como capas densas, convolucionales y recurrentes, así como funciones de activación. Esto simplifica la creación de modelos personalizados al ofrecer bloques de construcción comunes.

Se integra fácilmente con herramientas de visualización, como TensorBoard, que permite realizar un seguimiento del rendimiento del modelo, visualizar la arquitectura de la red y analizar métricas de entrenamiento.

El uso de Keras simplifica significativamente el proceso de desarrollo de modelos de deep learning, lo que lo convierte en una opción ideal para implementar sistemas de reconocimiento facial. Su versatilidad, facilidad de uso y compatibilidad con las principales bibliotecas de backend hacen de Keras una herramienta poderosa para la construcción y experimentación eficiente en proyectos de aprendizaje profundo.

Es compatible con la aceleración mediante el uso de unidades de procesamiento gráfico (GPU). Esto permite un entrenamiento más rápido de los modelos, especialmente útil al trabajar con grandes conjuntos de datos y arquitecturas complejas.

Especificaciones de la tarjeta gráfica A6000.

La tarjeta gráfica NVIDIA A6000, al ser parte de la serie profesional de GPUs, se utiliza comúnmente en entornos de trabajo que involucran grandes conjuntos de datos y tareas de procesamiento intensivo, como aquellas relacionadas con el aprendizaje profundo utilizando bibliotecas como Keras.

Aquí hay algunas razones clave por las cuales esta tarjeta gráfica puede ser beneficiosa para tales escenarios:

Paralelismo masivo: Las tarjetas gráficas modernas, incluida la A6000, están diseñadas con una gran cantidad de núcleos CUDA que permiten el procesamiento paralelo masivo. Esto es esencial para acelerar tareas que implican operaciones matriciales y cálculos intensivos, como los que se encuentran comúnmente en el aprendizaje profundo.

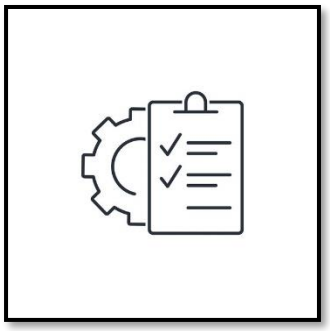
Memoria de video amplia: La A6000 generalmente viene con una cantidad significativa de memoria de video GDDR6 o GDDR6X. Esta memoria de alta velocidad es crucial para manejar grandes conjuntos de datos, modelos complejos y operaciones de entrenamiento que requieren un acceso rápido y eficiente a la memoria.

Tensor Cores para aceleración AI: La arquitectura Ampere en la que se basa la A6000 incluye Tensor Cores, que están diseñados específicamente para acelerar operaciones de inteligencia artificial (IA). Esto es beneficioso al utilizar bibliotecas de aprendizaje profundo como TensorFlow y Keras, que pueden aprovechar estas capacidades para mejorar el rendimiento.

Optimizaciones para Deep Learning Frameworks: La arquitectura y las características de las tarjetas gráficas NVIDIA están altamente optimizadas para trabajar con marcos de trabajo de aprendizaje profundo como TensorFlow y PyTorch. Keras, siendo una interfaz de alto nivel que se integra con TensorFlow, se beneficia de estas optimizaciones para ejecutar de manera más eficiente las tareas de aprendizaje profundo en la A6000.

Rendimiento de punto flotante: El rendimiento de punto flotante sólido de la A6000 es esencial para realizar operaciones numéricas intensivas, como las que se encuentran en las operaciones matriciales de las redes neuronales.

En resumen, la tarjeta gráfica A6000 es una elección popular para aquellos que trabajan con grandes conjuntos de datos y aplicaciones intensivas en cálculos, como el aprendizaje profundo con Keras. Su arquitectura potente y características específicas para IA hacen que sea una opción sólida para acelerar tareas de procesamiento de datos complejas.



> *48 gigabytes (GB) de memoria de GPU: La memoria GDDR6 ultrarrápida, escalable hasta 96 GB con NVLink, ofrece a los científicos de datos, ingenieros y profesionales creativos la gran cantidad de memoria necesaria para trabajar con conjuntos de datos y cargas de trabajo masivos como los de ciencia de datos y simulación.*

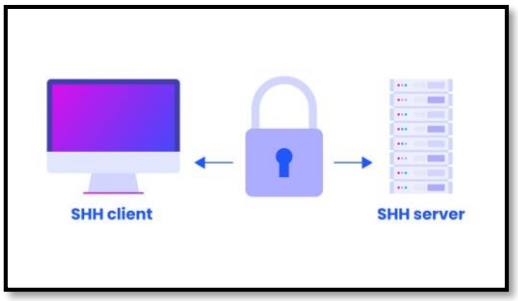
4. Configuración del Entorno

En este proyecto, utilizamos Visual Studio como nuestro entorno de desarrollo principal para la creación y desarrollo del modelo Keras. Visual Studio nos proporciona un conjunto de herramientas integradas que facilitan la escritura, depuración y optimización del código, permitiéndonos trabajar eficientemente en la implementación y entrenamiento del modelo de aprendizaje profundo utilizando la biblioteca Keras. La interfaz intuitiva y las funcionalidades de Visual Studio han sido fundamentales para agilizar el proceso de desarrollo y garantizar la calidad del código.

Además, en este proyecto, hemos configurado Visual Studio para conectarnos a un servidor virtual. Esta conexión nos permite aprovechar la potencia de la tarjeta gráfica A6000, que está instalada en el servidor remoto. Al utilizar una conexión SSH desde Visual Studio, podemos ejecutar y entrenar modelos de Keras en el servidor virtual, aprovechando la capacidad de procesamiento paralelo y la memoria de alta velocidad de la tarjeta A6000. Esta configuración nos permite manejar grandes conjuntos de datos y tareas intensivas en cómputo de manera eficiente, proporcionando un entorno de desarrollo más potente y escalable.

La conexión SSH en Visual Studio es un proceso que permite establecer una comunicación segura entre el entorno de desarrollo y un servidor remoto. Esto posibilita ejecutar y depurar código directamente en el servidor, aprovechando recursos como tarjetas gráficas potentes, como la A6000, que pueden estar instaladas en el servidor. La conexión SSH permite trabajar de manera eficiente en entornos distribuidos, donde la potencia de cómputo necesaria para tareas intensivas se encuentra en un servidor remoto.

La conexión SSH en Visual Studio es un proceso que permite establecer una comunicación segura entre el entorno de desarrollo y un servidor remoto. Esto posibilita ejecutar y depurar código directamente en el servidor, aprovechando recursos como tarjetas gráficas potentes, como la A6000, que pueden estar instaladas en el servidor. La conexión SSH permite trabajar de manera eficiente en entornos distribuidos, donde la potencia de cómputo necesaria para tareas intensivas se encuentra en un servidor remoto.



5. Conjunto de Datos

El conjunto de datos UTKFace es un conjunto de datos de rostros a gran escala con un amplio intervalo de edades (de 0 a 116 años). El conjunto de datos consta de más de 20.000 imágenes faciales con anotaciones de edad, sexo y origen étnico. Las imágenes presentan grandes variaciones en cuanto a pose, expresión facial, iluminación, oclusión, resolución, etc.

En este proyecto, empleamos un conjunto de datos que consta de 20,000 fotografías faciales proporcionadas generosamente por **https://susanqq.github.io/UTKFace/**. Este conjunto de datos se utiliza para diversas aplicaciones, como reconocimiento facial, clasificación por edad u otros proyectos relacionados con el análisis de imágenes faciales.

Análisis exploratorio de datos.

Descompensación en las Edades:

Uno de los aspectos que observamos durante nuestro análisis fue una descompensación significativa en las edades representadas en nuestro conjunto de datos. Al desglosar la distribución de las edades, notamos que ciertos grupos de edad estaban sobrerrepresentados en comparación con otros. Este desequilibrio en las edades puede influir en el rendimiento y la generalización de nuestro modelo de deep learning, ya que podría aprender sesgos asociados con las distribuciones desiguales de las edades en los datos de entrenamiento.

Agrupación de Edades en Categorías:

Dada la descompensación observada en las edades en nuestro conjunto de datos, hemos optado por agrupar las edades en categorías más amplias, a saber, joven, adulto y anciano. Esta estrategia nos permite simplificar la tarea del modelo al reducir la complejidad asociada con la predicción precisa de edades individuales, al tiempo que aborda la descompensación en la distribución de edades.

Reducción de la Descompensación:

Agrupar las edades en categorías más amplias nos permite mitigar la descompensación en la distribución de edades al agrupar edades menos representadas con aquellas que están más presentes en el conjunto de datos. Esto ayuda a equilibrar la distribución de edades y mejora la capacidad del modelo para generalizar a través de las diferentes categorías de edad.

Simplificación del Modelo:

Al reducir el número de clases objetivo de edades individuales a categorías más amplias, simplificamos la tarea del modelo de deep learning. Esto puede ayudar a mejorar la capacidad del modelo para aprender patrones relevantes en los datos y reducir el riesgo de sobreajuste, especialmente en conjuntos de datos limitados.

Limpieza de Datos: Se realizaron tareas de limpieza de datos para eliminar imágenes irrelevantes, duplicadas o de baja calidad que podrían afectar la calidad del conjunto de datos.

Normalización de Datos: Se normalizaron los datos para garantizar la consistencia en el formato, la resolución y otras características relevantes de las imágenes en todo el conjunto de datos.

División del Conjunto de Datos: Se dividió el conjunto de datos en conjuntos de entrenamiento, validación y prueba para su uso en el entrenamiento y evaluación del modelo de machine learning.

Balance de Clases: Se abordaron desequilibrios en la distribución de clases, como la agrupación de edades en categorías más amplias, para mejorar la capacidad del modelo para generalizar de manera equitativa a través de las diferentes clases.

6. Diseño del modelo Keras

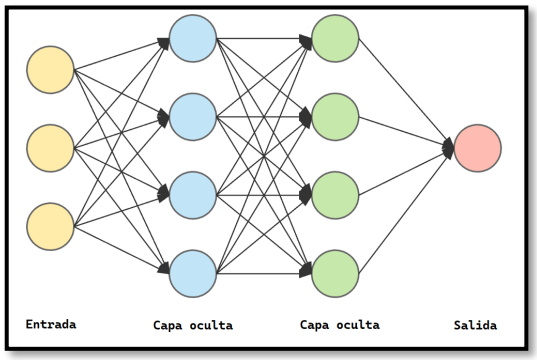
Este modelo de red neuronal convolucional (CNN) en Keras está diseñado para abordar tareas de clasificación binaria, particularmente en el contexto de procesamiento de imágenes. La arquitectura del modelo se compone de varias capas que se combinan para extraer características significativas de las imágenes y realizar predicciones precisas.

Capas y Parámetros Utilizados:

Capas Convolucionales (Conv2D): Las capas convolucionales son fundamentales en las CNN y se utilizan para convolucionar los datos de entrada, lo que permite la detección de características relevantes en las imágenes. En este modelo, hemos incluido tres capas convolucionales, cada una seguida de una activación ReLU para introducir no linealidades en los datos.

Capas de Agrupación (MaxPooling2D): Las capas de agrupación se utilizan para reducir la dimensionalidad de las características extraídas por las capas convolucionales, preservando al mismo tiempo la información más importante. Después de cada capa convolucional, hemos agregado capas de agrupación máxima (MaxPooling2D).

Capas Completamente Conectadas (Dense): Después de aplanar las características en un vector unidimensional, se han agregado capas densamente conectadas. La primera capa densa tiene 128 neuronas con activación ReLU y normalización por lotes (BatchNormalization), seguida de una capa de abandono (Dropout) para prevenir el sobreajuste. La capa de salida tiene una sola neurona con activación sigmoidal, adecuada para problemas de clasificación binaria.



Función de Pérdida y Optimizador:

Función de Pérdida (loss): La función de pérdida seleccionada es la entropía cruzada binaria (binary_crossentropy), ideal para problemas de clasificación binaria. Esta función mide la discrepancia entre las distribuciones de probabilidad predichas y reales.

Optimizador (optimizer): El optimizador Adam se ha elegido para ajustar los pesos del modelo durante el entrenamiento. Adam adapta automáticamente las tasas de aprendizaje durante el proceso de entrenamiento, lo que lo hace altamente eficiente y adecuado para una variedad de problemas de aprendizaje automático.

Justificación del Diseño:

La arquitectura del modelo ha sido seleccionada y diseñada cuidadosamente para maximizar el rendimiento y la generalización en la tarea de clasificación binaria. La combinación de capas convolucionales, capas de agrupación y capas completamente conectadas permite al modelo aprender características complejas y realizar predicciones precisas.

7. Entrenamiento del modelo y aplique de hiperparámetros

Aumentación de Datos con ImageDataGenerator:

En este paso, estás utilizando ImageDataGenerator de Keras para aumentar y preparar los datos antes de alimentarlos al modelo durante el entrenamiento y la validación. La aumentación de datos es una técnica clave en el aprendizaje profundo que ayuda a mejorar la generalización del modelo y a prevenir el sobreajuste.

ImageDataGenerator y Parámetros:

rescale: Normalizas los valores de píxeles de las imágenes dividiendo cada píxel por 255, lo que escala los valores a un rango entre 0 y 1.

width_shift_range y height_shift_range: Estableces un rango para desplazar horizontalmente y verticalmente aleatoriamente las imágenes durante el entrenamiento. Este parámetro introduce variación en la posición de los objetos en las imágenes, lo que ayuda al modelo a aprender invariantes espaciales.

horizontal_flip: Permite voltear horizontalmente aleatoriamente las imágenes durante el entrenamiento. Esto aumenta la diversidad del conjunto de datos al proporcionar al modelo ejemplos adicionales de la misma clase desde diferentes perspectivas.

Flujos de Datos:

train2 y test2: Estás creando flujos de datos (flow) utilizando los generadores datagen y test_datagen respectivamente. Estos flujos de datos contienen lotes de imágenes aumentadas y sus etiquetas correspondientes, y se utilizan para alimentar el modelo durante el entrenamiento y la validación.

Entrenamiento del Modelo:

history2: Al entrenar el modelo (genmodel.fit) utilizando los flujos de datos de entrenamiento y validación, estás almacenando el historial del entrenamiento para su posterior análisis y visualización.

epochs: Estableces el número de épocas de entrenamiento, es decir, el número de veces que el modelo verá todo el conjunto de datos de entrenamiento.

shuffle: Barajas aleatoriamente el conjunto de datos de entrenamiento en cada época para garantizar que el modelo no memorice el orden de los ejemplos.

callbacks: Utilizas una devolución de llamada (callbacks) de EarlyStopping para monitorear la pérdida en el conjunto de datos de validación y detener el entrenamiento si la pérdida no mejora después de un cierto número de épocas (patience=5).

Este proceso de aumentación de datos con ImageDataGenerator es una estrategia efectiva para mejorar la robustez y la generalización del modelo durante el entrenamiento. La combinación de aumentación de datos y técnicas de regularización como la devolución de llamada EarlyStopping ayuda a evitar el sobreajuste y a mejorar el rendimiento del modelo en datos no vistos.

8. Evaluación del Modelo

Detalles sobre el proceso de entrenamiento.

El método evaluate() en Keras se utiliza para calcular y evaluar el rendimiento de un modelo de aprendizaje automático sobre un conjunto de datos de prueba, que son datos que el modelo no ha visto durante el entrenamiento. Esta función proporciona una forma rápida y conveniente de obtener métricas de evaluación, como la pérdida y la precisión del modelo, en un conjunto de datos independiente.

Evaluación del Rendimiento del Modelo:

Pérdida (Loss): La pérdida es una medida de cuán bien el modelo está realizando la tarea que se le ha asignado. En el contexto de la clasificación, la pérdida suele ser la entropía cruzada (cross-entropy), que mide la discrepancia entre las distribuciones de probabilidad predichas por el modelo y las distribuciones reales de las etiquetas.

Precisión (Accuracy): La precisión es una métrica que indica la proporción de ejemplos clasificados correctamente por el modelo con respecto al total de ejemplos en el conjunto de datos de prueba.

Proceso de Evaluación:

Datos de Prueba: El método evaluate() toma como entrada los datos de prueba (X_test) y las etiquetas de prueba correspondientes (y_test). Estos datos no se utilizaron durante el entrenamiento del modelo y se utilizan exclusivamente para evaluar su rendimiento generalización.

Predicciones del Modelo: El modelo realiza predicciones sobre los datos de prueba utilizando las características proporcionadas por X_test.

Cálculo de la Pérdida y la Precisión: Una vez que se obtienen las predicciones del modelo, el método evaluate() calcula la pérdida y la precisión comparando las predicciones del modelo con las etiquetas reales proporcionadas en y_test.

Resultados de la Evaluación: El resultado del método evaluate() es una lista que contiene la pérdida y la precisión del modelo en el conjunto de datos de prueba.

Interpretación de los Resultados:

Una pérdida baja y una alta precisión indican que el modelo generaliza bien a datos no vistos y es capaz de realizar predicciones precisas en un entorno del mundo real.

Una pérdida alta y una baja precisión sugieren que el modelo puede estar sobreajustando los datos de entrenamiento y no generaliza bien a datos no vistos.

Conclusión:

El método evaluate() en Keras es una herramienta esencial para evaluar el rendimiento de un modelo de aprendizaje automático en datos independientes. Proporciona métricas cuantitativas que ayudan a los desarrolladores y científicos de datos a comprender la eficacia y la generalización del modelo en el mundo real.

Evaluación del Modelo de Aprendizaje Automático

El siguiente extracto representa una evaluación realizada en un modelo de aprendizaje automático. Esta evaluación proporciona información crucial sobre el desempeño y la eficacia del modelo en una tarea específica.

Modelo 1/131: Este indicador sugiere que se están evaluando múltiples modelos, y el modelo actual es el primero de un total de 131 modelos.

Progreso y Tiempo Estimado de Llegada (ETA): La barra de progreso muestra el avance de la evaluación, indicando que aún queda un 100% de los datos por evaluar con un ETA estimado de 10 segundos para la finalización de la evaluación.

Pérdida (loss) y Precisión (accuracy): La pérdida (loss) es una métrica que representa cómo de efectivo es el modelo para hacer predicciones. En este caso, la pérdida inicial es de 0.6248, lo que sugiere que el modelo puede estar cometiendo errores en sus predicciones. La precisión (accuracy) indica qué proporción de las predicciones del modelo son correctas, y en este punto, la precisión es del 68.75%.

Número de Lotes de Datos: El modelo se evalúa en un total de 131 lotes de datos, lo que implica que se están utilizando múltiples fragmentos de información para la evaluación del modelo.

Tiempo de Evaluación por Paso: Cada paso de evaluación toma en promedio 8 segundos y 57 milisegundos. Este tiempo puede variar dependiendo del tamaño y complejidad de los datos, así como de la capacidad de procesamiento del sistema.

Resultados Finales: Tras completar la evaluación de todos los lotes de datos, la pérdida final del modelo es de 0.5752 y la precisión final es del 73.91%. Estos resultados indican una mejora respecto a la fase inicial de evaluación, lo que sugiere que el modelo pudo ajustarse y mejorar su desempeño durante el proceso de evaluación.

Este análisis proporciona una visión detallada del proceso de evaluación de un modelo de aprendizaje automático, ofreciendo información valiosa sobre su desempeño y capacidad predictiva.

9. Implementación de la Aplicación