Título del Proyecto: Reconocimiento Inteligente de Imágenes (RIIM)

Objetivo General

Desarrollar una aplicación web simple y educativa que permita a los estudiantes implementar y comprender los principales conceptos de Inteligencia Artificial (IA), Machine Learning (ML), Deep Learning (DL) y las técnicas relacionadas. El proyecto será una introducción práctica y sencilla al mundo de la IA, utilizando herramientas modernas para procesar imágenes y extraer información.

Idea Principal

El sistema permitirá a los usuarios cargar imágenes y analizar aspectos como:

1. Clasificación (e.g., ¿es un perro o un gato?).
2. Generación automática de etiquetas relacionadas con el contenido de la imagen.
3. Detección de anomalías en las imágenes (e.g., encontrar patrones inusuales).

Este proyecto incorpora varias técnicas y conceptos importantes de IA y ML de manera modular, asegurando que cada tema pueda ser explorado por los estudiantes sin complejidades excesivas.

Componentes y Objetivos Específicos

Cada componente del proyecto está diseñado para introducir y aplicar conceptos clave**:**

1. **Frontend:**
    - Proporcionar una interfaz web que permita a los usuarios cargar imágenes y ver los resultados.
    - Mostrar cómo una red neuronal clasifica imágenes y genera etiquetas.
2. **Backend y Machine Learning:**
    - Implementar un modelo de red neuronal convolucional (CNN) para procesar imágenes.
    - Usar técnicas como aumento de datos, validación cruzada y poda de redes para optimizar el modelo.
    - Experimentar con modelos preentrenados y aplicar ajuste fino (fine tuning).
3. **Optimización y Despliegue:**
    - Convertir modelos a formatos optimizados (ONNX, TensorFlow Lite).
    - Implementar inferencia tanto en la nube como en dispositivos locales (edge computing).
4. **Educación en Técnicas Avanzadas:**
    - Introducir conceptos como overfitting, generalización, etiquetado de datos y cuantificación.

- Comparar y contrastar enfoques supervisados, no supervisados y de refuerzo.

5. **Integración Continua y Servicios Cloud:**
   - Configurar un sistema automatizado para probar y desplegar el código.
   - Mostrar cómo usar plataformas cloud para entrenar y desplegar modelos.

Flujo del Proyecto

1. **Carga de Imágenes:**
   - El usuario sube una imagen a través de la interfaz web.
2. **Procesamiento en el Backend:**
   - El modelo clasifica la imagen (e.g., identifica si es un perro o gato).
   - Se genera un análisis adicional, como detección de anomalías o generación de etiquetas.
3. **Visualización de Resultados:**
   - El frontend muestra los resultados del análisis de manera interactiva.
   - Los usuarios pueden explorar cómo el modelo llegó a su conclusión.
4. **Aprendizaje Interactivo:**
   - Entrenamiento del modelo.
   - Optimización y despliegue.
   - Implementación del frontend y conexión al backend.

Componentes del Proyecto

1. **Frontend:**
   - **Lenguajes y Frameworks:** HTML, CSS, JavaScript, Bootstrap para la interfaz de usuario.
   - **Funcionalidad:**
     - Subir imágenes.
     - Mostrar resultados del análisis (etiquetas, anomalías detectadas, etc.).
     - Visualización de redes neuronales (interfaz gráfica para mostrar cómo funciona la red).
2. **Backend:**
   - **Lenguaje y Framework:** Python con Flask o Django.
   - **Servicios:** Modelos entrenados usando TensorFlow o PyTorch exportados a ONNX para inferencia rápida.

3. **Modelo de Machine Learning:**
    - **Red Neuronal Convolucional (CNN):**
        - Para clasificación de imágenes (e.g., determinar si una imagen muestra un perro o un gato).
    - **Técnicas Usadas:**
        - **Cuantificación:** Reducir el tamaño del modelo para una inferencia más rápida.
        - **Poda de Redes:** Simplificar la red eliminando conexiones redundantes.
        - **Fine Tuning:** Ajuste del modelo pre-entrenado para las necesidades del proyecto.
        - **Aumento de Datos:** Generar imágenes adicionales rotadas, espejadas, etc., para mejorar la generalización.
        - **Validación Cruzada:** Para verificar la precisión del modelo.
4. **Servicios Cloud y Edge Computing:**
    - **Cloud:** Usar Google Cloud AI o AWS para el entrenamiento del modelo.
    - **Edge:** Implementar la inferencia del modelo directamente en el navegador con TensorFlow.js.
5. **Evaluación del Modelo:**
    - Detectar overfitting con gráficos de pérdida y precisión.
    - Mostrar cómo el modelo mejora con el etiquetado manual de datos cargados por los usuarios.

Plan de Implementación

1. **Estructura del Proyecto**
    - Configuración del entorno (Python, Flask/Django, TensorFlow/PyTorch).
    - Diseño de la interfaz web básica.
2. **Backend y API**
    - Crear una API que acepte imágenes y devuelva resultados de clasificación.
    - Integrar ONNX para optimización del modelo.
3. **Modelos de Machine Learning**
    - Entrenar una CNN simple con un dataset pequeño.
    - Aplicar aumento de datos, validación cruzada y cuantificación.
4. **Integración y Pruebas**
    - Integrar frontend con backend.
    - Implementar aprendizaje supervisado y no supervisado en el análisis de datos.

5. **Semana 5: Documentación y Evaluación**
    - Crear una guía para que los alumnos puedan entender y replicar el proyecto.
    - Realizar pruebas finales y evaluar el modelo.

Tareas

- Implementar el frontend básico y conectarlo con la API.
- Entrenar modelos simples y aplicar técnicas como poda de redes y fine tuning.
- Realizar la cuantificación y desplegar el modelo en un entorno de inferencia cloud/edge.
- Evaluar el modelo detectando overfitting y ajustando parámetros.
- Ampliar la funcionalidad para incorporar aprendizaje no supervisado y por refuerzo

Que necesitas para resolverlo:

1. Cuantificación

Qué es: Cuantificación es un proceso de optimización para reducir el tamaño de los modelos de Machine Learning y hacer que su inferencia sea más rápida. Esto se logra representando los pesos del modelo con números de menor precisión (e.g., convertir pesos en flotantes de 32 bits a 8 bits).

Qué necesitas:

- **Bibliotecas:** TensorFlow o PyTorch tienen funciones integradas para cuantificación.
- **Modelo:** Necesitas un modelo previamente entrenado (e.g., una CNN).
- **Herramientas:** TensorFlow Lite para cuantificación en dispositivos móviles.

Pasos:

1. Entrena el modelo con precisión completa.
2. Usa las funciones de TensorFlow o PyTorch para convertirlo a un modelo cuantificado.

---

2. Poda de Redes

Qué es: La poda de redes (network pruning) es una técnica que elimina conexiones neuronales innecesarias para hacer el modelo más pequeño y rápido sin comprometer mucho su precisión.

Qué necesitas:

- **Modelo entrenado:** Una red neuronal preentrenada.
- **Herramientas:** Librerías como TensorFlow Model Optimization Toolkit.

Pasos:

1. Identifica las conexiones menos importantes (aquellas con pesos muy pequeños).
2. Elimina estas conexiones y reentrena el modelo para recuperar precisión.

---

3. ONNX (Open Neural Network Exchange)

Qué es: ONNX es un formato estándar que permite transferir modelos de Machine Learning entre diferentes frameworks (e.g., convertir un modelo entrenado en PyTorch para usarlo en TensorFlow o un sistema de inferencia personalizado).

Qué necesitas:

- **Modelo:** Creado en frameworks como TensorFlow, PyTorch o Scikit-learn.
- **Conversor:** Librerías como onnx y onnxruntime.

Pasos:

1. Exporta tu modelo al formato ONNX usando PyTorch (torch.onnx.export) o TensorFlow.

2. Usa ONNX Runtime para realizar inferencias rápidas con el modelo exportado.

---

4. Integración Continua
Qué es: Es una práctica de desarrollo donde el código se prueba y despliega automáticamente cada vez que alguien realiza un cambio en el repositorio.
Qué necesitas:

- **Plataforma CI/CD:** GitHub Actions, GitLab CI, o Jenkins.
- **Código:** Un repositorio con tu proyecto.
- **Pruebas:** Tests automáticos para garantizar que el modelo funciona correctamente.

Pasos:

1. Configura un archivo de flujo de trabajo (.yaml) en GitHub Actions.
2. Escribe pruebas para verificar el correcto funcionamiento del código.
3. Configura despliegues automáticos tras pasar las pruebas.

---

5. Servicios Cloud Machine Learning
Qué es: Son servicios como Google Cloud AI, AWS SageMaker, o Azure ML que facilitan el entrenamiento, despliegue y escalado de modelos de Machine Learning.
Qué necesitas:

- **Cuenta en la nube:** Google, AWS, o Azure.
- **Modelo o datos:** Para entrenar o desplegar.
- **Configuración:** Credenciales y configuraciones de acceso a los servicios.

Pasos:

1. Sube tus datos y modelo al servicio cloud.
2. Entrena o despliega el modelo utilizando las herramientas que ofrece la plataforma.

---

6. Edge Computing
Qué es: Inferencia de modelos directamente en dispositivos locales (como móviles o IoT) en lugar de depender de un servidor.
Qué necesitas:

- **Modelo optimizado:** Usualmente con TensorFlow Lite o PyTorch Mobile.
- **Dispositivo:** Un móvil o dispositivo IoT con capacidad para ejecutar modelos.

Pasos:

1. Optimiza el modelo con TensorFlow Lite o PyTorch Mobile.
2. Integra el modelo en una aplicación móvil.

---

7. Deep Learning (CNN, LSTM, GAN, Auto codificadores)
Qué es:

- **CNN:** Redes neuronales convolucionales, ideales para imágenes.
- **LSTM:** Redes de memoria a largo plazo, útiles para datos secuenciales como texto.
- **GAN:** Redes Generativas Adversarias, para crear contenido como imágenes.
- **Auto codificadores:** Redes que aprenden representaciones comprimidas de datos.

Qué necesitas:

- **Frameworks:** TensorFlow o PyTorch.
- **Conocimientos básicos:** Entender el tipo de datos que procesan.

Pasos:

1. Elige el modelo que se ajuste a tu problema.
2. Entrénalo con datos relevantes.

---

8. Fine Tuning
Qué es: Usar un modelo preentrenado y ajustarlo a tu problema con datos específicos.
Qué necesitas:

- **Modelo preentrenado:** Como ResNet, VGG, etc.
- **Dataset:** Datos específicos para ajustar el modelo.

Pasos:

1. Carga el modelo preentrenado.
2. Reentrena solo las últimas capas con tu dataset.

---

9. Etiquetado de Datos
Qué es: Asignar etiquetas a datos para entrenar modelos supervisados.
Qué necesitas:

- **Dataset sin etiquetas.**
- **Herramientas:** Aplicaciones como LabelImg (para imágenes).

Pasos:

1. Usa una herramienta para etiquetar los datos.
2. Guarda las etiquetas en un formato legible (JSON, CSV).

---

10. Aumento de Datos
Qué es: Generar más datos a partir de los existentes aplicando transformaciones como rotación, zoom, etc.
Qué necesitas:

- **Dataset inicial.**

- **Librerías:** Keras o PyTorch para aumento.

Pasos:

1. Define las transformaciones que deseas.
2. Aplica estas transformaciones al dataset.

---

11. Generalización y Overfitting

Qué es:

- **Generalización:** Capacidad del modelo para funcionar con datos nuevos.
- **Overfitting:** Cuando el modelo se ajusta demasiado a los datos de entrenamiento y falla con nuevos datos.

Qué necesitas:

- **Dataset:** Conjunto de entrenamiento y prueba.
- **Técnicas:** Regularización (e.g., dropout).

Pasos:

1. Divide los datos en entrenamiento y prueba.
2. Ajusta la arquitectura y parámetros del modelo para evitar overfitting.

---

12. Validación Cruzada

Qué es: Una técnica para evaluar modelos dividiendo los datos en varios subconjuntos.

Qué necesitas:

- **Dataset:** Con suficientes datos para dividirlos.
- **Librerías:** Scikit-learn ofrece funciones para validación cruzada.

Pasos:

1. Usa cross_val_score de Scikit-learn para realizar validación.

---

13. Aprendizaje Supervisado, No Supervisado y por Refuerzo

Qué es:

- **Supervisado:** Modelo entrenado con datos etiquetados.
- **No supervisado:** Identifica patrones en datos no etiquetados.
- **Por refuerzo:** Aprende a través de recompensas y castigos.

Qué necesitas:

- **Dataset:** Etiquetado para supervisado o no etiquetado para no supervisado.
- **Frameworks:** TensorFlow o PyTorch.

Pasos:

1. Selecciona el enfoque según el tipo de datos.
2. Entrena el modelo con el algoritmo correspondiente.

---

14. Redes Neuronales y Backpropagation

Qué es:

- **Redes Neuronales:** Modelos que simulan el cerebro humano.
- **Backpropagation:** Algoritmo para ajustar los pesos en redes neuronales.

Qué necesitas:

- **Librerías:** TensorFlow, PyTorch.
- **Modelo básico:** Una red neuronal simple.

Pasos:

1. Diseña una red neuronal.
2. Usa backpropagation para entrenarla.

---

15. Herramientas Generales Necesarias

- **Python (última versión).**
- **TensorFlow, PyTorch, Keras.**
- **Flask/Django para backend.**
- **GitHub para control de versiones.**
- **Google Colab o Jupyter Notebooks para entrenar modelos.**