Assignment brief A.B.

PORTADA

Nombre Alumno / DNI	Guillermo De Brabanter Quiroga – 09107267ª
Título del Programa	Cybersecurity & Hacking
Nº Unidad y Título	UNIT 25 – Applied Machine Learning
Año académico	2025-2026
Profesor de la unidad	Rabindranath Andujar
Título del Assignment	AB FINAL
Día de emisión	13/01/2026
Día de entrega	20/01/2026
Nombre IV y fecha	
Declaración del estudiante	Certifico que la presentación del assignment es completamente mi propio trabajo y entiendo completamente las consecuencias del plagio. Entiendo que hacer una declaración falsa es una forma de mala práctica. Fecha: 16/01/2026 Firma del alumno: 

Plagio

El plagio es una forma particular de hacer trampa. El plagio debe evitarse a toda costa y los alumnos que infrinjan las reglas, aunque sea inocentemente, pueden ser sancionados. Es su responsabilidad asegurarse de comprender las prácticas de referencia correctas. Como alumno de nivel universitario, se espera que utilice las referencias adecuadas en todo momento y mantenga notas cuidadosamente detalladas de todas sus fuentes de materiales para el material que ha utilizado en su trabajo, incluido cualquier material descargado de Internet. Consulte al profesor de la unidad correspondiente o al tutor del curso si necesita más consejos.

MSMK University

Contenido

Introducción

El uso del *Machine Learning* y *Deep Learning* ha aumentado en los últimos años ya que es capaz de, por ejemplo, clasificar automáticamente imágenes. Por ello, se ha convertido en una herramienta fundamental en la seguridad, salud y controles de accesos.

Durante la pandemia, el uso de mascarillas obligo a los sistemas de detección facial, generar un sistema que verifica rápidamente y que tenga fiabilidad. Por ello, este proyecto lo he orientado un poco hacia eso. El objetivo de este proyecto es detectar si dicha persona lleva mascarilla o no a partir de una imagen.

Por lo tanto, he generado un modelo de *Machine Learning,* abordando todo el proceso para obtener unos resultados buenos, pero principalmente entender el proceso, analizar los errores y la justificar decisiones técnicas.

Definición del problema

Contexto del problema

La detección de las mascarillas mediante estas herramientas, pueden aplicarse en muchos tipos de sistemas. Automatizar este proceso va a permitir reducir la intervención humana y mejorar la eficiencia de los sistemas.

El problema que he planteado es el siguiente: "¿Cómo de posible y fácil es detectar si una persona lleva mascarilla o no a través de un modelo?"

Objetivo

El objetivo principal de este proyecto es desarrollar un sistema que sea capaz de:

- Clasifica imágenes faciales en dos: **WithMask** y **WithoutMask**
- Evaluar el rendimiento del modelo entrenado
- Analizar los errores del modelo y comprender sus limitaciones
- Exportar el modelo para utilizarlo en un entorno web
- Integrar el modelo en una aplicación web con **Next.js**
- Desplegar la aplicación en producción mediante **Vercel**

Usuario final y valor de la predicción

El usuario final puede ser cualquier persona que necesite verificar el uso de mascarilla a partir de imágenes. El valor de este proyecto consiste en:

- La automatización del proceso de detección
- La reducción de latencia al ser usado por el cliente
- La mejora de la privacidad, ya que las imágenes no se envían a un servidor

Hipótesis

Antes de desarrollar el modelo se pensó lo siguiente:

- El uso de *Transfer Learning* permitiría un buen rendimiento con un dataset de tamaño moderado
- Un modelo ligero como MobileNetV2 sería suficiente para captar los patrones visuales relevantes
- El modelo podría presentar dificultades con imágenes borrosas, oclusiones o mascarillas poco contrastadas

Fundamentación teórica

Clasificación de imágenes y redes neuronales

La clasificación de las imágenes se basa habitualmente en redes neuronales convolucionales. Estas redes están diseñadas para procesar datos con estructura espacial, y son capaces de aprender automáticamente características de los píxeles de entrada (imágenes).

Transfer Learning

Consiste en reutilizar el modelo entrenado, adaptándolo a un nuevo problema, permitiendo reducir el tiempo de entrenamiento y la cantidad de datos.

Relación con los conceptos trabajados

El entrenamiento del modelo pone en práctica conceptos como:

- **Pesos y bias**: parámetros ajustables
- **Función de pérdida**: medida del error entre la predicción y la realidad
- **Optimización**: ajustando parámetros mediante algoritmos
- **Regularización**: técnicas como “Early Stopping”

Ingeniería y análisis de datos

Descripción del dataset

El dataset está compuesto por imágenes faciales etiquetadas con: WithMask y WithoutMask. Estas imágenes se diferencian por la iluminación, claridad, ángulo de rostro, tipo de mascarilla, etc…

Esta diversidad resulta relevante ya que permite al modelo entrenarse con condiciones variables.

Exploración inicial de los datos

Antes de iniciar con el entrenamiento, se exploraron los datos con el objetivo de:

- Verificar la correcta asignación de etiquetas
- Evaluar la calidad general de las imágenes
- Identificar posibles dificultades

Conseguí observar como he comentado en sus diferencias, imágenes desenfocadas, resoluciones bajas, oclusiones parciales. Prediciendo errores y dificultades que el modelo puede experimentar.

Procesamiento de los datos

Todas las imágenes fueron transformadas para adaptarse a los requisitos:

- Redimensionado a 224x244
- Conversión a RGB
- Normalización de valores de píxel mediante la función "preprocess_input" del modelo

De esta forma, todos los datos son coherentes.

Data augmentation

Con el objetivo de mejorar la capacidad de generalizar al modelo y reducir su sobreajuste, se aplicó esta técnica durante el entrenamiento **únicamente**.

- Rotaciones aleatorias
- Volteo horizontal
- Pequeños desplazamientos y escalados

Entrenamiento

El entrenamiento tuvo dos fases:

Entrenamiento inicial

Entrené únicamente la cabeza de clasificación añadida, manteniendo congeladas las capas del modelo base. Así adaptamos rápidamente el clasificador al problema sin alterar características generales ya aprendidas.

Para controlarlo, implementamos técnicas como:

- *Early Stopping*, para detener el entrenamiento cuando la validación dejaba de mejorar
- *ReduceLROnPlateau,* para ajustar dinámicamente la tasa de aprendizaje

Fine-tuning

Posteriormente, se realizó un fine-tuning, descongelando parcialmente las últimas capas y así como bien dice el nombre, conseguimos refinar lo aprendido y mejorar el rendimiento del modelo sin dar evidencias de sobreajuste.

Resultados y análisis de errores

Métricas cuantitativas

El modelo obtuvo una precisión cercana al 99%, lo que indica un claro rendimiento alto clasificando.
Además de la precisión, analizamos curvas de pérdida y precisión durante el entrenamiento y la validación, observándose una convergencia estable y sin divergencias significativas entre ambos conjuntos.

Esto demuestra que es capaz de clasificar sin haber tenido dicho dato/imagen previamente.

Análisis de errores

A pesar del elevado rendimiento, se identificaron casos donde el modelo no clasificó correctamente. Dio error por los siguientes patrones:

- **Imágenes borrosas**
- **Rostros parcialmente ocluidos**
- **Objetos distintos tapando la boca y nariz**

Estos casos sugieren que el modelo tiende a determinar que una oclusión significa uso de mascarilla.

Predicciones de baja confianza

Se analizaron también predicciones, incluso cuando la clasificación fue correcta. De esta forma se identificaron imágenes límite, significando que el modelo mostraba que no estaba seguro de su respuesta.

Este estudio es relevante, ya que proporciona información de los límites del modelo y de su fiabilidad en un entorno real.

Conclusión de resultados

Las hipótesis iniciales fueron cumplidas. Conseguimos obtener el rendimiento elevado con un dataset de tamaño considerable.

Eso sí, el análisis de errores muestra las limitaciones contra el dataset, mostrando la importancia de complementar con métricas cuantitativas.

Arquitectura y despliegue

Arquitectura general

La arquitectura del sistema sigue un enfoque de inferencia en el lado del cliente, donde el modelo que hemos entrenado se ejecuta directamente en el navegador del usuario.

El flujo es de la siguiente forma:
Usuario -> Aplicación Web -> Modelo -> Predicción -> Usuario

El diseño elimina la necesidad de un backend para realizar estas ejecuciones.

Integración del modelo en aplicación web

El modelo entrenado fue exportado a TensorFlow.js y almacenado en el directorio "public/model" para que la aplicación web lo ejecute a petición del usuario.

La imagen subida se preprocesa en el navegador siguiendo los mismos pasos que en el entrenamiento.

Inferencia en el navegador

La inferencia es completamente realizada en el navegador de forma que:

- Reduce la latencia
- Mejora la privacidad ya que no se almacena nada
- Elimina costes asociados de backend si lo hubiese

Pero bueno, si que es verdad, que de esta forma, el rendimiento depende del hardware del usuario y el navegador, aunque no debería haber problema.

Despliegue en producción

La aplicación web se desplegó en producción mediante Vercel, que proporciona varios frameworks diferentes, en este caso Next.js.

El despliegue permite que la aplicación sea accesible públicamente desde cualquier navegador, mediante la siguiente URL: **https://ab-applied-machine-learning.vercel.app**

Esto demuestra que un modelo puede ir desde el entorno de entrenamiento hasta una situación real.

Logs de predicción

Se incluye un sistema básico de logging en cliente como prueba de concepto, registrando las predicciones realizadas durante la sesión y mostrando dichos eventos en la interfaz.

Esto permite visualizar el comportamiento del modelo en tiempo real y sirve como base para una futura persistencia en backend orientada a reentrenamiento.

Conclusiones

Este proyecto demuestra que el uso de *Transfer Learning* junto a modelos preentrenados como MobileNetV2, permite alcanzar un gran rendimiento con muchos tipos de datasets, incluso de tamaño moderado.

La precisión del 99% que alcanzamos, valida la arquitectura y estrategia de entrenamiento utilizada. Gracias a las métricas y mayoritariamente al análisis de errores, hemos aprendido las limitaciones del modelo y determinamos las situaciones reales donde el modelo puede escasear.

Bibliografía

TensorFlow *Transfer learning & fine-tuning (Keras)*
https://www.tensorflow.org/tutorials/images/transfer_learning

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018)
MobileNetV2: Inverted Residuals and Linear Bottlenecks
https://arxiv.org/abs/1801.04381

Keras Documentation *EarlyStopping Callback*
https://keras.io/api/callbacks/early_stopping/

Keras Documentation *ReduceLROnPlateau Callback*
https://keras.io/api/callbacks/reduce_lr_on_plateau/

TensorFlow *Save and load models (Keras)*
https://www.tensorflow.org/guide/keras/save_and_serialize

TensorFlow *The SavedModel format*
https://www.tensorflow.org/guide/saved_model

TensorFlow.js *Import a TensorFlow SavedModel into TensorFlow.js*
https://www.tensorflow.org/js/tutorials/conversion/import_saved_model

TensorFlow.js Documentation *Save and load models in TensorFlow.js*
https://www.tensorflow.org/js/guide/save_load