Vision Transformers sobre EuroSAT-RGB

Figura 1: Imágen ilustrativa del conjunto de datos. 🔗

Bruno Masoller

Mauro Aguirregaray

Juan Cruz Ferreyra

Resumen

Este trabajo se desarrolló en el marco de la asignatura Visión por Computadora III de la Maestría en Inteligencia Artificial de la Universidad de Buenos Aires, cuyo enfoque se centra en modelos modernos de visión por computadora basados en transformers. El objetivo del proyecto fue evaluar el desempeño de tres arquitecturas representativas —*ViT-Base, Swin Transformer y Convolutional Vision Transformer*— en una tarea de clasificación multiclase sobre el conjunto de datos satelitales *EuroSAT-RGB*, empleando técnicas de aprendizaje por transferencia. Además, se incluyó un modelo *YOLOv11-cls* como línea base para establecer comparaciones. El informe documenta el diseño experimental, los principales desafíos técnicos enfrentados durante la implementación y el análisis comparativo de los resultados obtenidos.

Introducción

En este trabajo se aborda una tarea de clasificación de imágenes satelitales mediante modelos de visión por computadora basados en arquitecturas modernas. El objetivo principal es comparar el desempeño de distintas variantes de *Vision Transformer* (ViT) con al menos un modelo clásico basado en redes convolucionales profundas. Esta comparación permite, por un lado, poner en práctica técnicas de aprendizaje por transferencia sobre modelos pre entrenados y, por otro, analizar las capacidades reales de las arquitecturas transformer en el dominio seleccionado.

La elección de este enfoque responde al interés por explorar herramientas actuales en un contexto concreto y delimitado, donde sea posible evaluar tanto el proceso de adaptación como la eficacia predictiva de los modelos. En particular, se busca determinar si las ventajas atribuidas a los transformers en el tratamiento de imágenes naturales también se manifiestan en escenarios menos explorados, como el análisis de imágenes satelitales. Esta inquietud está motivada por estudios recientes con conclusiones divergentes: mientras que algunos trabajos reportan mejoras significativas [ref], otros señalan limitaciones frente a datos fuera de distribución o entornos con bajo volumen de ejemplos [ref].

Cabe destacar que el presente estudio tiene un carácter exploratorio y fue desarrollado en el marco del curso Visión por Computadora III de la Maestría en Inteligencia Artificial de la Universidad de Buenos Aires. Aunque se toman como referencia trabajos científicos actuales, el enfoque está orientado principalmente a la aplicación práctica de contenidos teóricos abordados durante la cursada.

Descripción del Proyecto

Versionado de código

Se utilizó GitHub como plataforma de control de versiones. Todo el trabajo se encuentra disponible en el repositorio brunomaso1/vision-transformer, pudiendo ser posible reproducir los resultados obtenidos en este trabajo. El repositorio incluye el código fuente, los scripts de entrenamiento y evaluación, así como los notebooks utilizados para el análisis exploratorio de datos y la visualización de resultados.

Estructura

La estructura de este trabajo se organiza siguiendo las recomendaciones de la herramienta "cookiecutter-data-science", que propone una estructura modular y coherente para proyectos de ciencia de datos. Puede obtenerse más información en el "readme.txt" del proyecto.

No fue necesaria ninguna adaptación con respecto a la estructura original que genera esta herramienta. Sin embargo, no se utilizó la herramienta `make` sugerida por los autores, ya que se implementó un flujo de trabajo utilizando Prefect y otras herramientas de orquestación de tareas. Por lo tanto, el archivo "Makefile" no refleja las tareas que se ejecutan en este proyecto, sino que se ha dejado como referencia para futuras implementaciones.

Herramientas

Documentación

Para la documentación del proyecto se utilizó *MkDocs*, una herramienta de documentación estática que permite generar sitios web a partir de archivos Markdown. La documentación se encuentra en el directorio "docs" y se puede acceder de forma online en https://brunomaso1.github.io/vision-transformer. Esta documentación tiene estructura de "wiki" y brinda enlaces a los distintos notebooks utilizados en el proyecto, así como a la información general relevante.

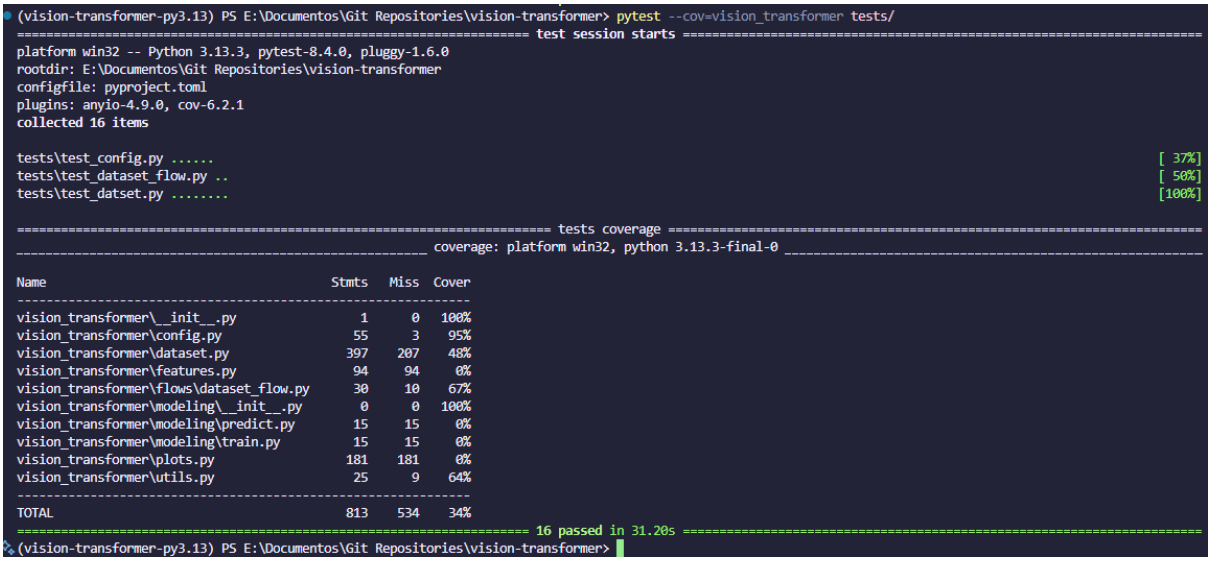
La wiki fue desplegada utilizando GitHub Pages, lo que permite que la documentación sea accesible públicamente y fácil de navegar.

Tests

Usualmente los proyectos de ciencia de datos no incluyen tests unitarios, ya que el enfoque está más orientado a la exploración y análisis de datos. Sin embargo, en este proyecto se implementaron algunos tests básicos para verificar el correcto funcionamiento de las funciones principales del código. Estos tests se encuentran en el directorio “tests” y se ejecutan utilizando “pytest”. Se puede obtener más información en el siguiente enlace de la wiki.

Cabe destacar, que existe una baja cobertura de tests, ya que no se implementaron tests para todas las funciones del código. Esto se debe a que el enfoque principal del proyecto fue la exploración y análisis de datos, y no tanto la implementación de pruebas exhaustivas. Sin embargo, se espera que en futuras iteraciones del proyecto se puedan agregar más tests para mejorar la cobertura y garantizar el correcto funcionamiento del código.

Actualmente, la cobertura de tests se puede observar en la siguiente imagen:



Orquestación de tareas

Para la orquestación de tareas se utilizó Prefect, una herramienta que permite definir flujos de trabajo y ejecutar tareas de manera programada. Los flujos de trabajo se encuentran en el directorio “flows” y se pueden ejecutar utilizando el comando “prefect run flow”. Estos flujos de trabajo permiten automatizar tareas como la descarga de datos, el entrenamiento de modelos y la generación de informes. En la wiki se puede encontrar más información sobre los flujos de trabajo implementados.

MLflow

Finalmente, se utilizó “MLFlow” para el seguimiento de experimentos y la gestión de modelos. Esta herramienta permite registrar métricas, parámetros y artefactos de los experimentos realizados, así como gestionar los modelos entrenados. Los experimentos se pueden visualizar en la interfaz web de MLFlow, lo que facilita el análisis y comparación de los resultados obtenidos.

En el siguiente enlace a la wiki existe más información y capturas del panel de MLFlow.

Infraestructura

Como infraestructura, se utilizó un entorno virtual creado con Poetry y los modelos se entrenaron tanto en una máquina local con una GPU NVIDIA RTX 4080 como en Google Colab. El modelo que más tiempo de entrenamiento requirió fue el SwinV2-Large, que se entrenó durante aproximadamente 9 horas en la GPU local. Se puede obtener más información sobre los tiempos de entrenamiento de cada modelo en sus respectivos notebooks de entrenamiento, los cuales se pueden acceder desde la wiki en el siguiente enlace: Notebooks de entrenamiento.

Dataset

Para este estudio se empleó la versión RGB del conjunto de datos *EuroSAT*, que contiene 27.000 imágenes satelitales georeferenciadas capturadas por el satélite Sentinel-2. Las imágenes, todas de 64×64 píxeles y en el espectro visible, están clasificadas en 10 categorías que reflejan distintos tipos de uso del suelo, desde áreas agrícolas hasta zonas urbanas. Esta homogeneidad en el tamaño de los datos simplifica considerablemente las tareas de carga y preprocesamiento. Además, al limitarse a las bandas RGB, se favorece la compatibilidad con modelos pre entrenados sobre imágenes naturales. La figura 2 muestra una selección de imágenes —una por clase— que ilustra la diversidad visual presente en el conjunto.

Figura 2: Imagen ejemplo por clase. 🔗

El análisis exploratorio inicial reveló una distribución balanceada entre clases y una separación visual razonable entre muchas de ellas. Sin embargo, también se observaron similitudes relevantes en algunos casos. Por ejemplo, las clases “Annual Crop” y “Permanent Crop” presentan patrones repetitivos difíciles de distinguir a simple vista. Esta ambigüedad podría representar un desafío para los modelos, especialmente en las primeras etapas del entrenamiento. En contraste, otras clases como “Highway” o “River” muestran morfologías más singulares y fácilmente reconocibles. La figura 3 resume gráficamente la cantidad de instancias por clase, tanto en el conjunto de entrenamiento como en el de evaluación.

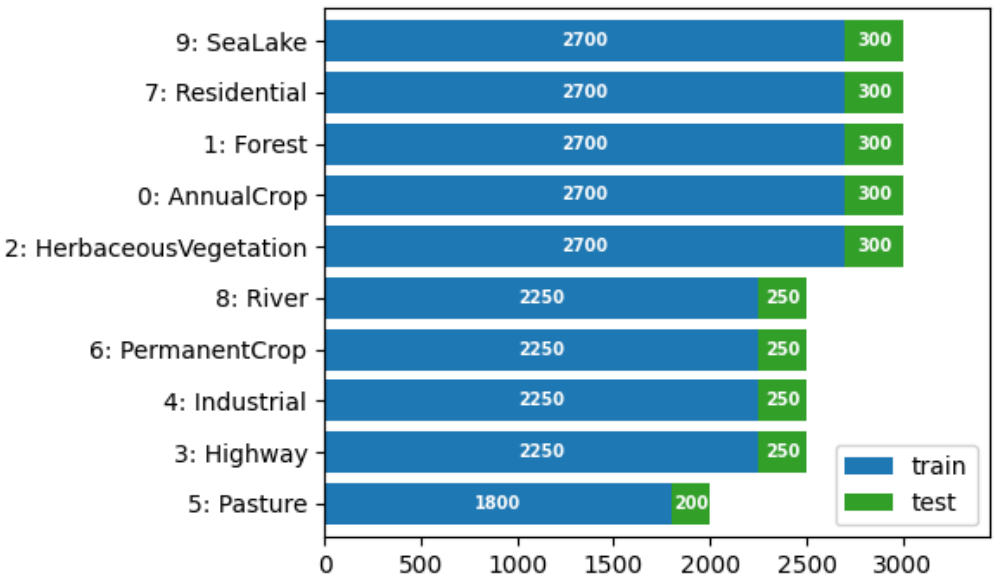


Figura 3: Cantidad de instancias por clase. 🔗

Modelos

1. Vision Transformer base

El modelo *Vision Transformer* (ViT), presentado por Dosovitskiy et al. en 2021 [ref], marcó un punto de inflexión en el campo de la visión por computadora al introducir con éxito los mecanismos de atención propios de los transformers al procesamiento de imágenes. A diferencia de las redes convolucionales, que construyen representaciones jerárquicas mediante filtros locales, ViT divide la imagen de entrada en parches fijos (generalmente de 16×16 píxeles), los proyecta a un espacio de embeddings y los trata como una secuencia, de manera análoga a las palabras en una oración. Esta formulación permite aplicar directamente arquitecturas desarrolladas originalmente para procesamiento de lenguaje natural, como el *Transformer-Encoder* de Vaswani et al. [ref], al dominio visual.

La arquitectura *ViT-Base*, representada en la figura 4, sigue esta lógica con una estructura simple pero potente. Cada imagen es tokenizada en una secuencia plana de embeddings, a la que se añade un token especial de clasificación y una codificación posicional fija. Esta secuencia es luego procesada por múltiples bloques de atención y capas feed-forward, manteniendo el mismo tamaño a lo largo de toda la red. Finalmente, la predicción se realiza a partir del estado final del token de clasificación. Si bien este diseño carece de inductivos espaciales como la localidad o la jerarquía presentes en las CNN, el ViT ha demostrado un desempeño competitivo en múltiples benchmarks al ser entrenado con grandes volúmenes de datos, lo que motivó su adopción como punto de partida en numerosos trabajos posteriores.

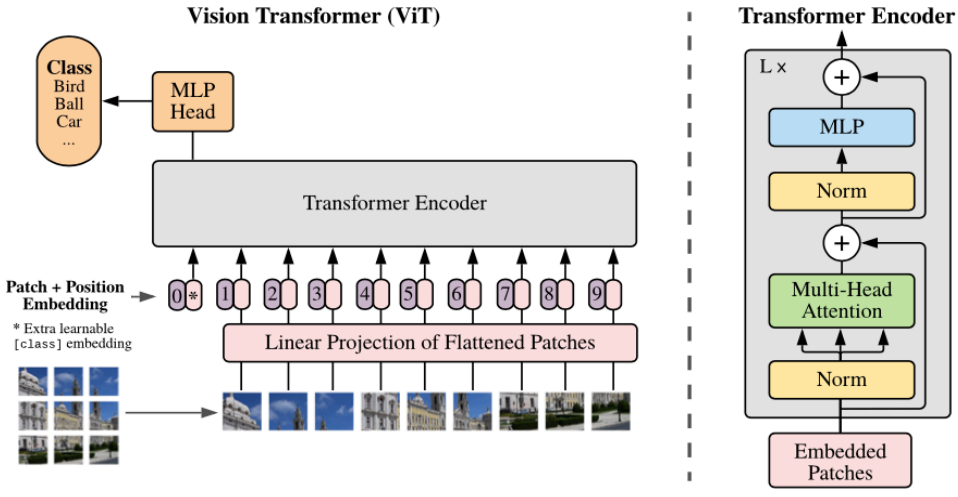


Figura 4: Diagrama de arquitectura ViT. 🔗

2. Swin Transformer v2

El *Swin Transformer*, propuesto por Liu et al. en 2021 [ref], es un modelo jerárquico de visión por computadora que introduce un mecanismo de atención localizado mediante ventanas deslizantes. Esta arquitectura permite capturar eficientemente tanto características locales como globales, adaptándose a distintas escalas espaciales sin perder estructura. Su diseño, basado en aplicar transformaciones dentro de regiones delimitadas de la imagen, ofrece una mejora significativa en eficiencia computacional frente a los *Vision Transformer* tradicionales, lo que lo convierte en una opción sólida para tareas como clasificación y detección de objetos. Desde su publicación, ha sido adoptado ampliamente en diversos dominios de la visión artificial.

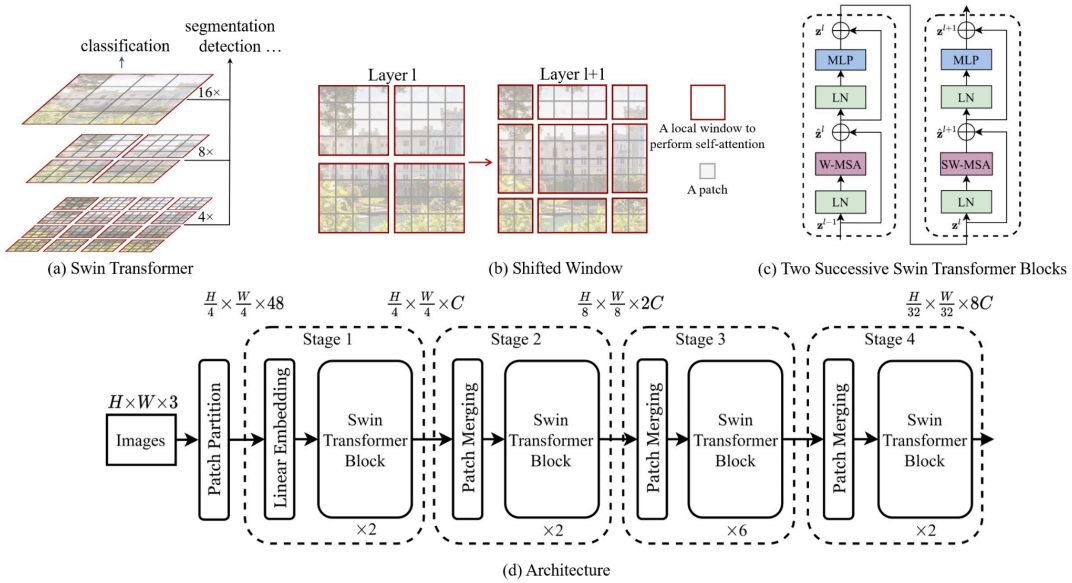


Figura 5: Diagrama de arquitectura Swin Transformer. 🔗

La segunda versión del modelo, publicada en 2022 por Liu et al. [ref], introduce varias mejoras significativas respecto a la original. Entre ellas se destaca la adopción de una configuración *res-post-norm*, que reemplaza el esquema *prenorm* utilizado previamente, así como la incorporación de un mecanismo de atención cosenoidal escalada en lugar del tradicional *dot product*. Además, se abandona el enfoque parametrizado anterior para implementar una representación basada en posicionamiento relativo continuo y espaciado logarítmicamente. Estas modificaciones no solo facilitan la escalabilidad del modelo en términos de capacidad, sino que también mejoran su capacidad de generalización al trabajar con distintas resoluciones de ventanas. Un resumen visual de estos cambios se presenta en la figura 6.

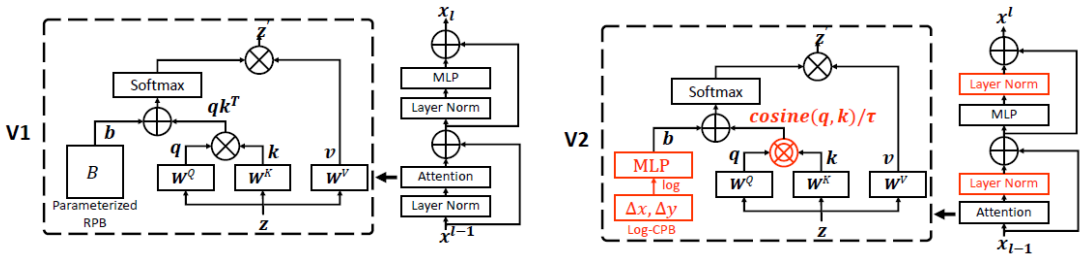


Figura 6: Cambios introducidos en la arquitectura de Swin Transformer V2. 🔗

3. Convolutional Vision Transformers

El modelo *Convolutional Vision Transformer* (CvT), desarrollado por el equipo de Microsoft Research [ref], representa una arquitectura híbrida que combina elementos característicos de los Vision Transformer (ViT) y de las redes convolucionales (CNN). Su diseño busca mejorar el desempeño y la eficiencia de los ViT tradicionales al incorporar inductivos espaciales propios de las CNN, como el sesgo local y la equivariancia traslacional. De este modo, CvT conserva la capacidad de atención global y escalabilidad de los transformers, pero con una mayor estructura jerárquica y menor costo computacional.

Estas mejoras se implementan a través de dos cambios fundamentales respecto a los ViT estándar. En primer lugar, se reemplaza el tradicional patch embedding por una etapa denominada *Convolutional Token Embedding*, que introduce convoluciones en la generación de los tokens de entrada. En segundo lugar, las matrices que producen las representaciones de consulta, clave y valor (*Q, K, V*) no se obtienen mediante capas lineales, sino mediante bloques convolucionales en lo que se denomina *Convolutional Projection for Attention*. Un diagrama de la arquitectura de este modelo se observa en la figura 7.

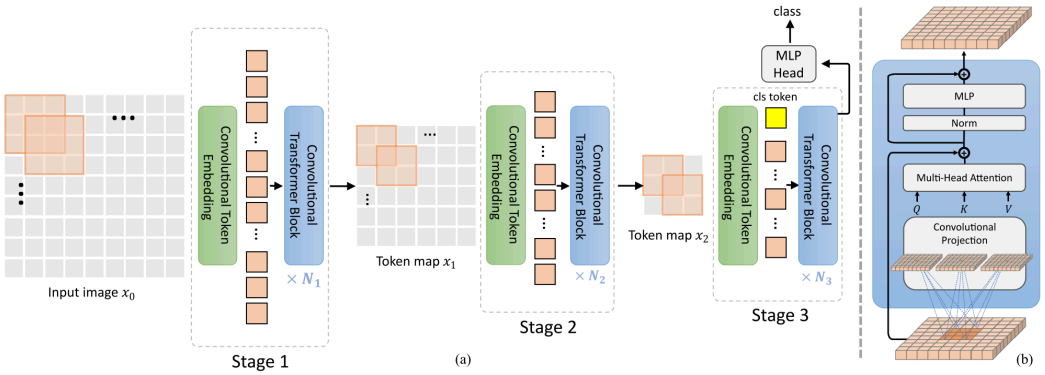


Figura 6: Arquitectura del Convolutional Vision Transformer. 🔗

4. You Only Look Once

Como línea base para la comparación de resultados, se incluyó el modelo YOLOv11m-cls, una variante adaptada de la arquitectura YOLOv11 para tareas de clasificación de imágenes. Si bien YOLO ha sido históricamente asociado a la detección de objetos, su versión para clasificación permite evaluar su desempeño en escenarios más simples, conservando ciertas ventajas de eficiencia y rapidez características del diseño original.

Entrenamiento

Para el entrenamiento de las tres arquitecturas basadas en transformers —ViT, Swin Transformer y CvT— se optó por utilizar la biblioteca *Transformers* desarrollada por Hugging Face. Esta elección se fundamenta en las ventajas que ofrece dicha librería para la experimentación con modelos pre entrenados en visión por computadora. En primer lugar, brinda acceso directo a una amplia variedad de arquitecturas a través de una API unificada, que permite descargar configuraciones y pesos desde el *Model Hub* con apenas unas líneas de código. Esta disponibilidad resulta especialmente útil para comparar modelos sin necesidad de implementar cada uno desde cero.

Además, la librería proporciona una clase de alto nivel denominada *Trainer*, que encapsula el ciclo de entrenamiento sobre *PyTorch* y permite lograr un equilibrio adecuado entre flexibilidad y facilidad de uso. A través de esta interfaz, es posible definir parámetros de entrenamiento, métricas de evaluación y estrategias de validación sin reescribir explícitamente los bucles internos. Otro aspecto clave es la integración nativa con la librería *datasets*, también de *Hugging Face*, que facilita la carga, transformación y partición de datos. Esta compatibilidad permite pasar el conjunto de entrenamiento al *Trainer* directamente, resolviendo por detrás tareas como la generación de *dataloaders* o la aplicación de estrategias de *batching*, lo cual agiliza el desarrollo sin perder control sobre los detalles relevantes del proceso.

Con el objetivo de mejorar la capacidad de generalización de los modelos entrenados mediante aprendizaje por transferencia, se implementaron técnicas de aumentación de datos en línea durante el entrenamiento. Para ello se utilizaron las transformaciones provistas por la librería *torchvision.transforms*, aprovechando su integración directa con *PyTorch*. Las modificaciones aplicadas fueron deliberadamente moderadas, ya que el

objetivo no era simular condiciones extremas sino introducir variabilidad visual suficiente como para reforzar la robustez del modelo ante leves cambios de orientación, iluminación o disposición espacial. En particular, se incluyeron transformaciones como rotación aleatoria, espejado horizontal y vertical (random flip), y ajuste de brillo, contraste y saturación mediante color jitter.

Durante el entrenamiento de los modelos basados en transformers no se congelaron capas del backbone, ya que se consideró que la diferencia entre los datos utilizados para el pre entrenamiento (como ImageNet) y el dominio satelital de EuroSAT era lo suficientemente significativa como para justificar una actualización completa de los pesos. Además, la cantidad moderada de imágenes disponibles —junto con el uso de técnicas de aumentación— permitía suponer que el modelo contaría con suficiente información para aprender sin riesgo inmediato de sobreajuste. El tamaño del batch fue variable en función de las restricciones del entorno de cómputo, aunque para la mayoría de los experimentos se estableció en 64. Se utilizó una tasa de aprendizaje del orden de 1e-5, junto con una fase de warm-up del 10 % del total de pasos, con el fin de estabilizar la adaptación de los pesos en las primeras épocas. Adicionalmente, se implementó un criterio de early stopping con una paciencia de dos épocas consecutivas sin mejora, criterio razonable dadas las limitaciones prácticas de un trabajo académico. Por último, se evaluaron múltiples variantes de cada arquitectura: en el caso de Swin Transformer, se probaron las versiones Tiny, Base y Large, mientras que para CvT se emplearon las versiones CvT-13 y CvT-21, que difieren principalmente en la cantidad de parámetros entrenables.

En cuanto al modelo YOLOv11m-cls, se utilizó la API provista por Ultralytics, que permite descargar, configurar y entrenar modelos de clasificación a partir de imágenes organizadas en carpetas por clase. Se optó por mantener los hiperparámetros por defecto ofrecidos por la herramienta, sin congelar el backbone, permitiendo así que todo el modelo se ajustara al dominio específico de EuroSAT-RGB. Dado que este modelo actuó como línea base en el análisis comparativo, no se realizaron experimentos adicionales con variantes arquitectónicas ni ajustes finos del entrenamiento.

Resultados

La evaluación cuantitativa se realizó sobre el conjunto de test, utilizando métricas estándar para clasificación multiclase: accuracy, precision, recall y f1-score. De esta forma, se obtiene una visión más completa del comportamiento del modelo, más allá de una única medida agregada. Las métricas se calcularon de manera global, considerando el promedio ponderado por clase, y se presentan en las tablas que siguen.

Si bien no se reportan tiempos de inferencia debido a la ejecución en distintos entornos de hardware, sí se registraron tiempos de inferencia dentro de cada familia de modelos, lo que permite identificar tendencias relativas. En el caso de Swin Transformer, se observó un aumento considerable del tiempo de inferencia al incrementar la escala del modelo: la versión Large duplicó el tiempo requerido por la versión Tiny para procesar el mismo conjunto. En cambio, entre las versiones de CvT, el tiempo de inferencia resultó sorprendentemente estable, aun cuando CvT-21 presenta una cantidad de parámetros significativamente mayor que CvT-13, lo que sugiere una mayor eficiencia relativa en su diseño interno.

Modelo	Accuracy	F1 score	Precision	Recall	Epocas
Vit-base	0.987	0.987	0.987	0.98	10
SwinV2-Tiny	0.984	0.984	0.984	0.984	3
SwinV2-Base	0.987	0.987	0.987	0.987	7
SwinV2-Large	0.993	0.993	0.993	0.993	10
CvT-13	0.979	0.979	0.979	0.979	5
CvT-21	0.976	0.976	0.977	0.976	4
YOLOv11-cls	0.981	0.982	0.982	0.981	14

La figura 7 resume visualmente el desempeño de los modelos evaluados, mostrando para cada uno de ellos las cuatro métricas consideradas: accuracy, precision, recall y f1-score. Como puede observarse, el modelo SwinV2-Large obtuvo los mejores resultados generales, superando el 99 % en todas las métricas, incluso a pesar de haber alcanzado el límite de 10 épocas de entrenamiento antes de activarse el criterio de early stopping, lo cual

sugiere que aún existía margen potencial de mejora. En segundo lugar se ubicó el Swin-Base métricas en torno al 98.7 %, confirmando su capacidad para generalizar correctamente sobre el conjunto de evaluación. Por otro lado, si bien CvT fue la familia de modelos con menor rendimiento dentro de los evaluados, alcanzó valores cercanos al 98 %, lo que sugiere que todas las arquitecturas consideradas son suficientemente expresivas para resolver de manera eficaz el problema planteado, y que las clases del dataset presentan una diferenciación considerable.

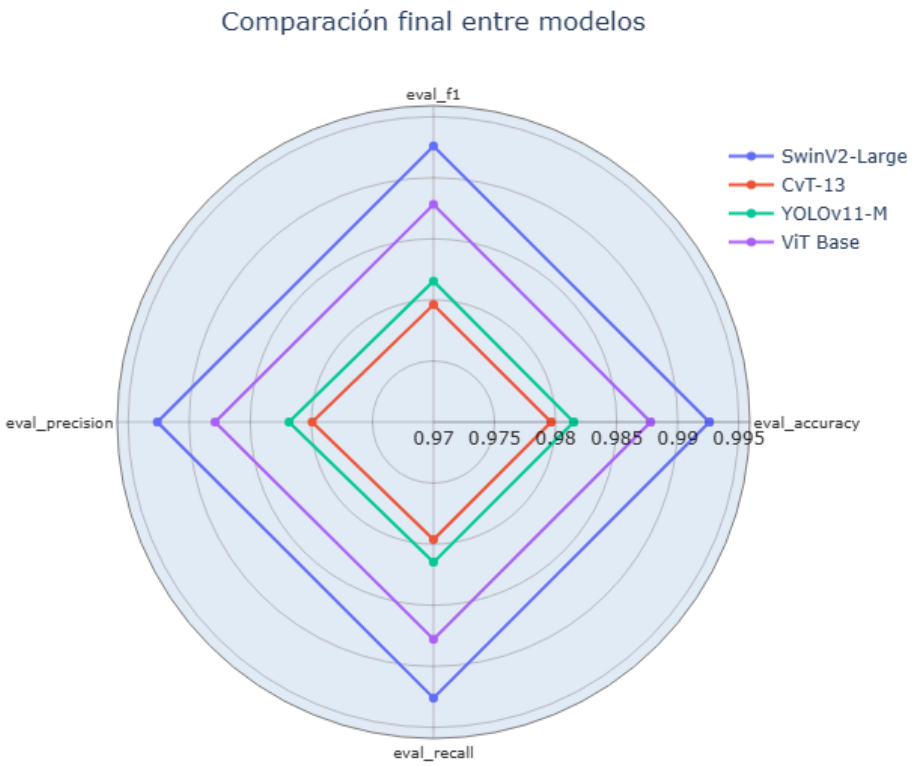


Figura 7: Ploteo radar de las métricas obtenidas por los modelos. 🔗

Los resultados detallados por modelo, así como las métricas de entrenamiento registradas por época, se presentan en el anexo de este informe.

Conclusiones

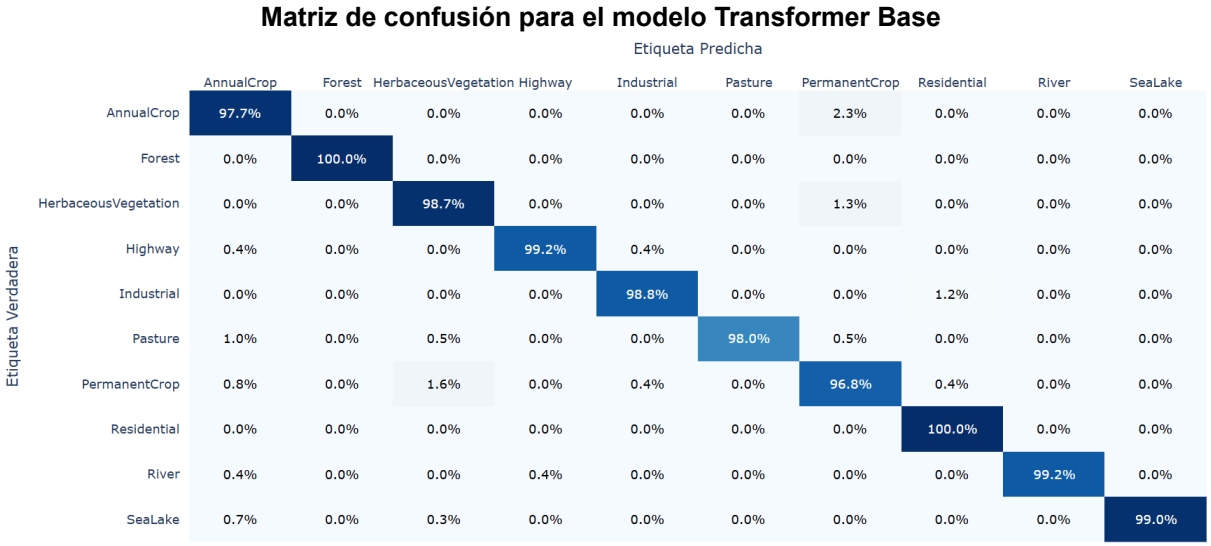
El trabajo permitió entrenar y evaluar distintos modelos Vision Transformer (ViT) utilizando el framework de Hugging Face, logrando un pipeline completo de clasificación de imágenes satelitales con resultados muy buenos. Todos los modelos alcanzaron métricas altas, sin diferencias significativas salvo un leve aumento del error al distinguir entre las clases Permanent Crop y Annual Crop, donde se pronuncia un mayor error en los modelos basados en CNN. Si bien inicialmente se buscaba comprobar si ViT supera a las CNN, los resultados obtenidos no permiten validar ni refutar dicha hipótesis, ya que el desempeño general fue alto en todos los casos. Como próximos pasos, se propone evaluar la robustez de los modelos frente a imágenes fuera del dataset original, lo cual permitiría explorar diferencias más claras entre arquitecturas.

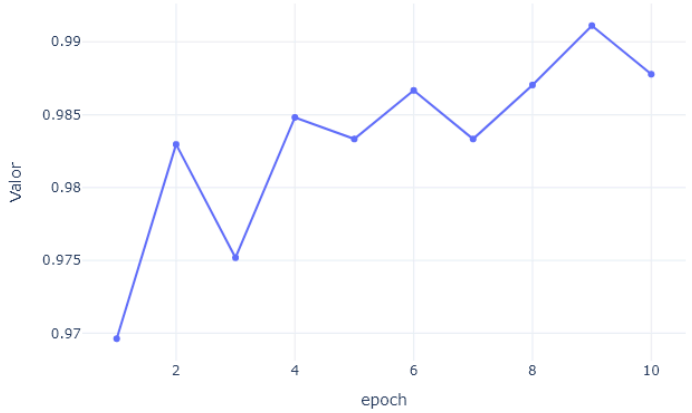
Anexo

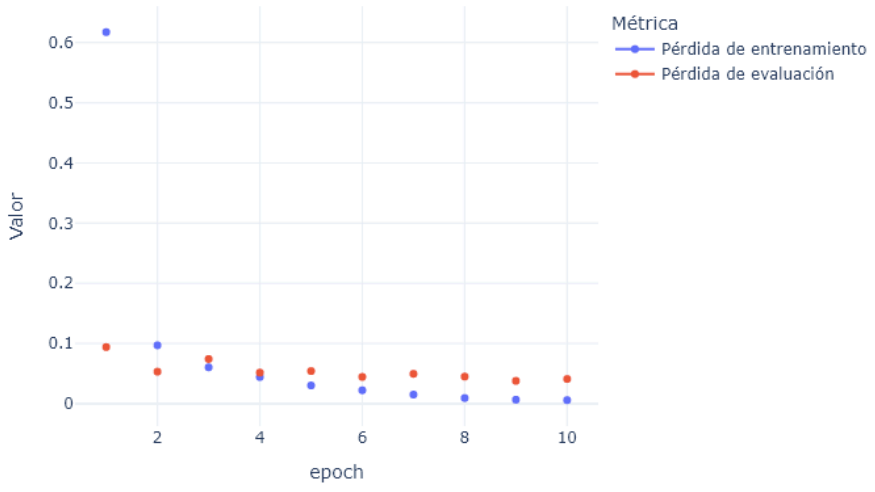
En las páginas siguientes se muestran los resultados pormenorizados del entrenamiento de cada uno de los modelos utilizados en este trabajo.

Cada uno de ellos cuenta con la evolución de las métricas durante el entrenamiento, así como las métricas calculadas sobre el set de evaluación de los modelos finales.

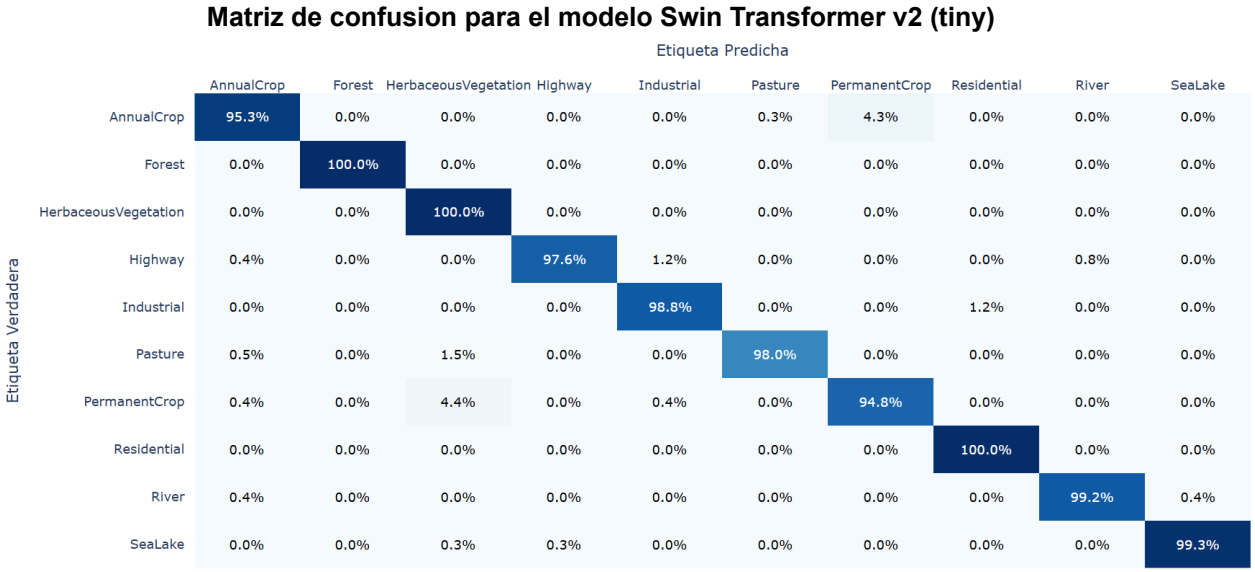
1. Vision Transformer base

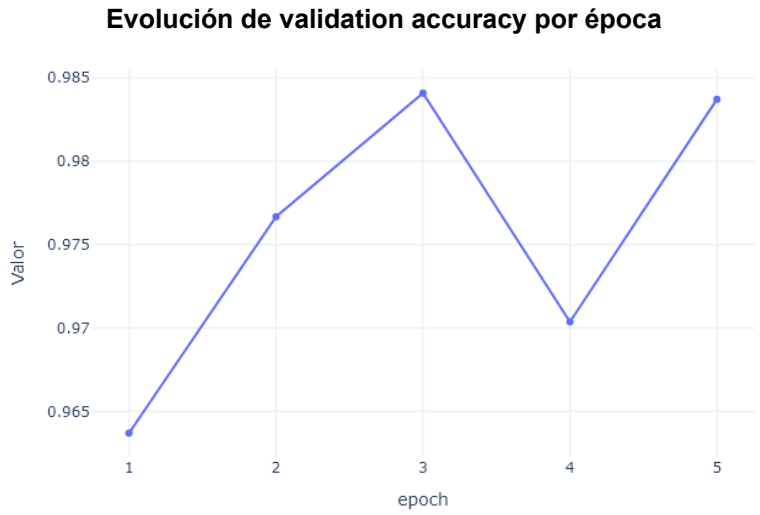






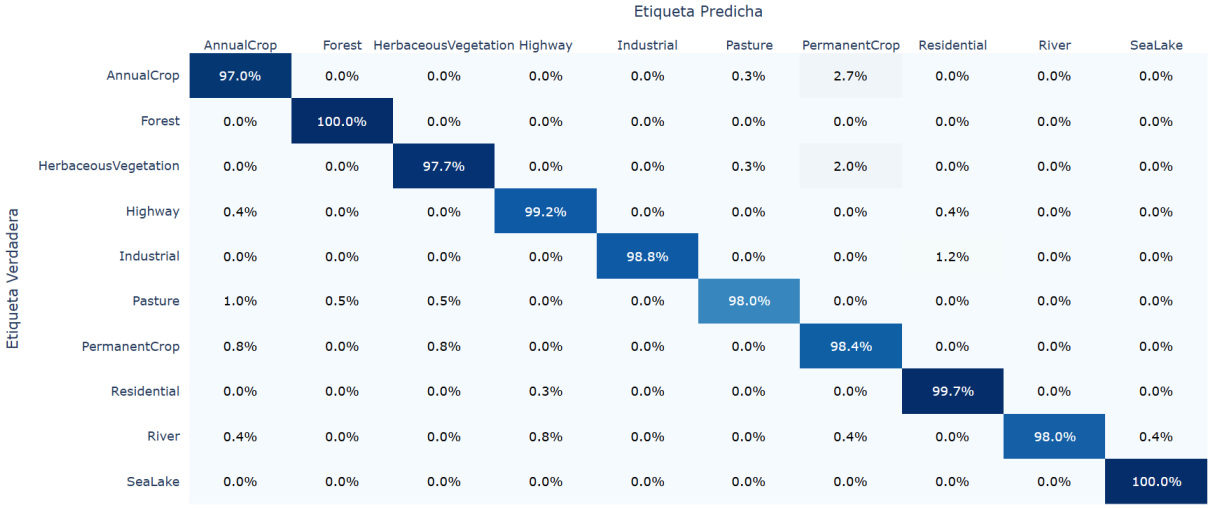
2. Swin Transformer v2 (tiny)



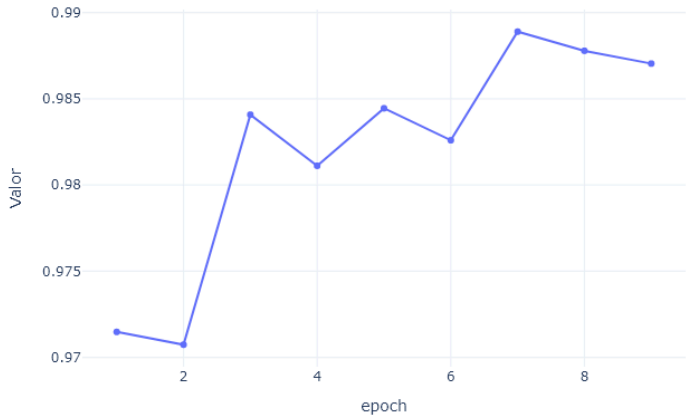


2. Swin Transformer v2 (base)

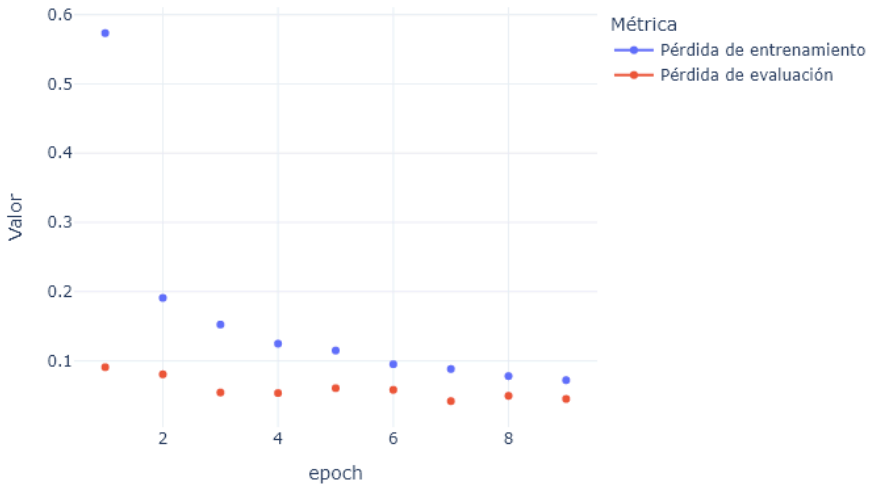
Matriz de confusion para el modelo Swin Transformer v2 (base)



Evolucion de validation accuracy por época

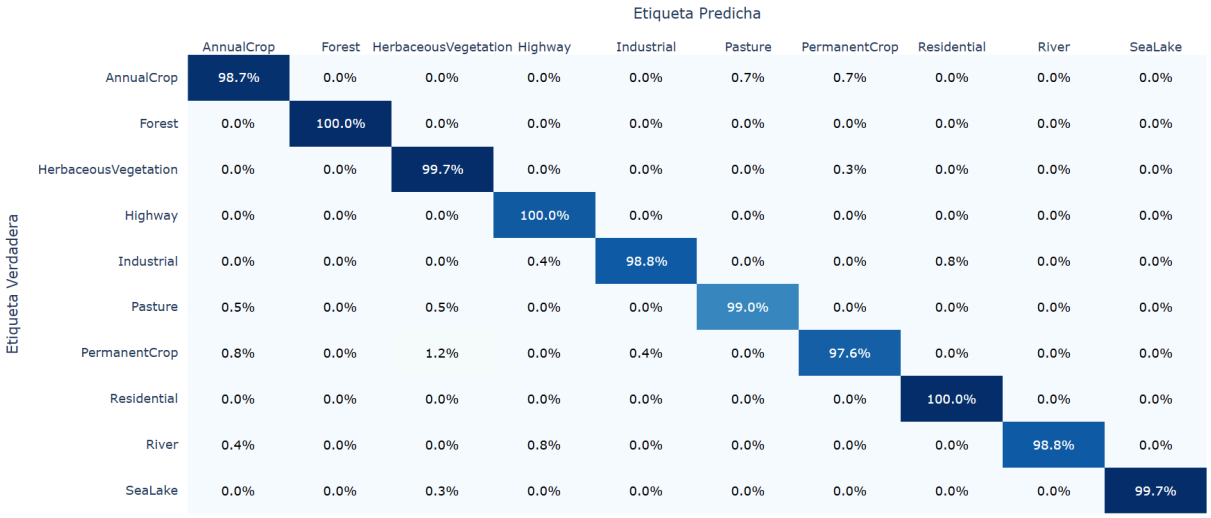


Evolucion de la funcion de perdida por época

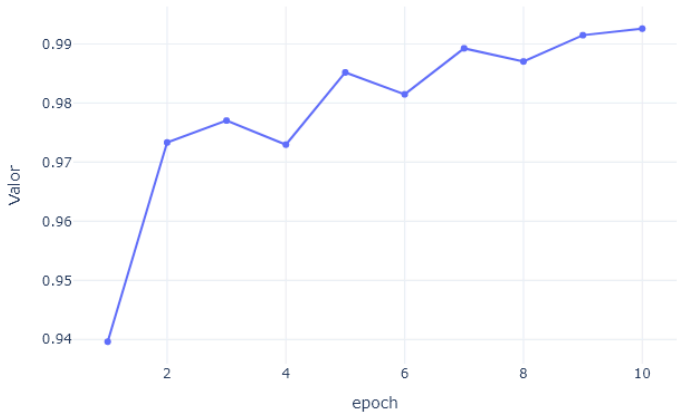


2. Swin Transformer v2 (large)

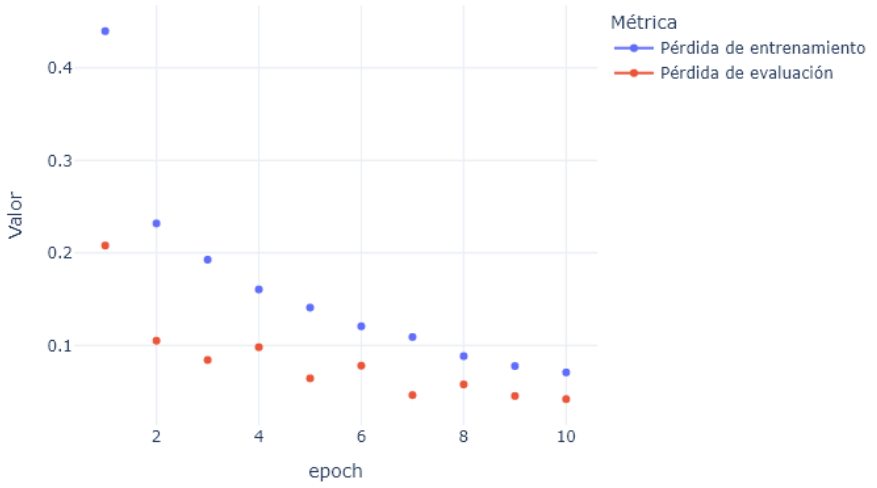
Matriz de confusion para el modelo Swin Transformer v2 (large)



Evolucion de validation accuracy por época

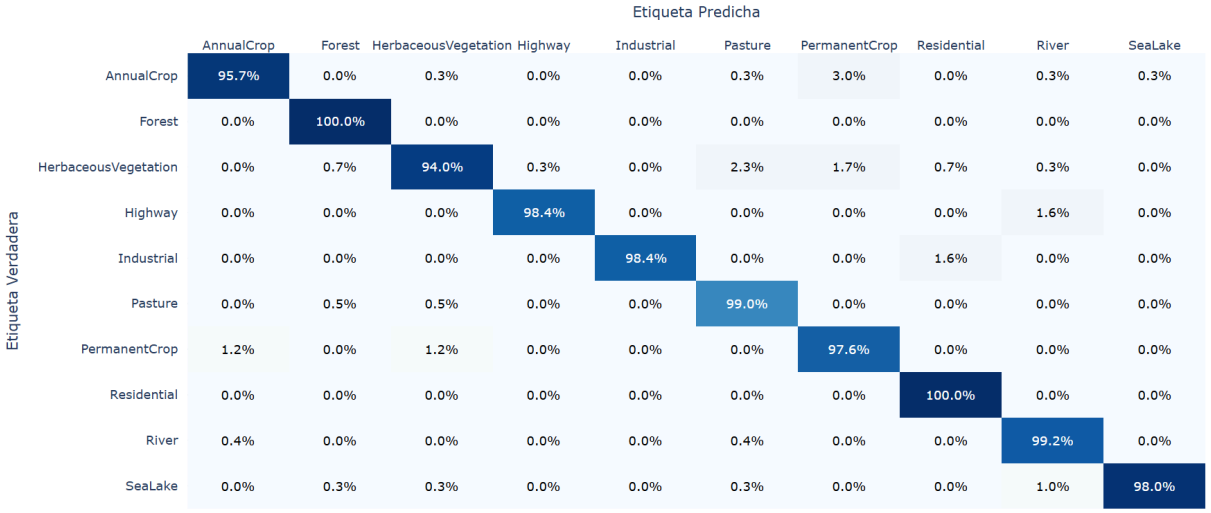


Evolucion de la funcion de perdida por época

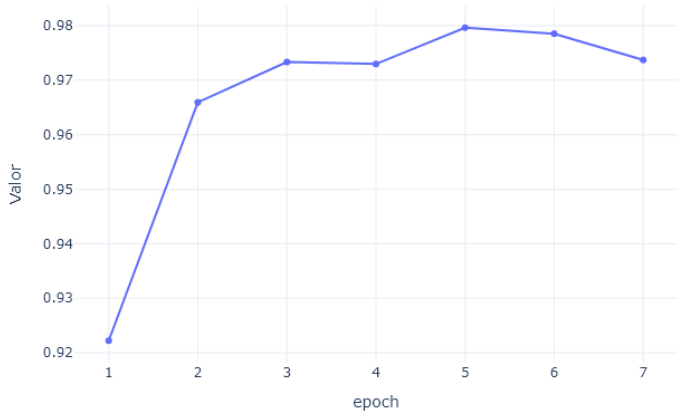


3. Convolutional Vision Transformers (cvt-13)

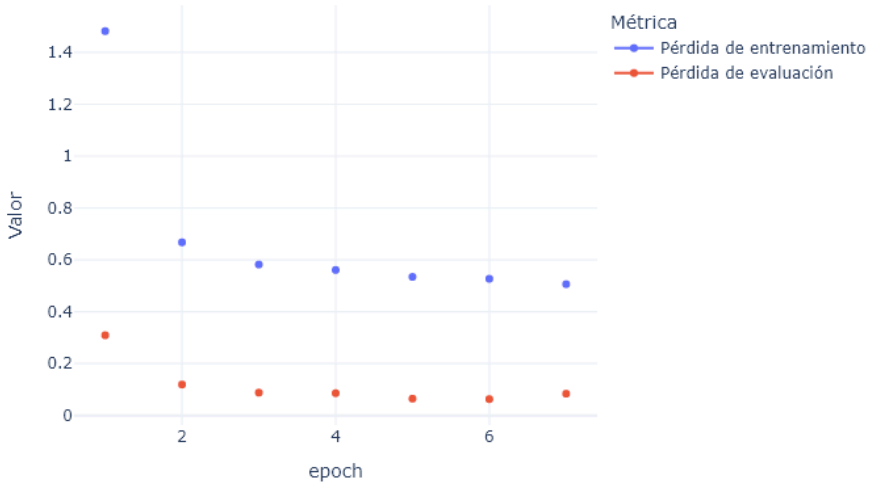
Matriz de confusion para el modelo Convolutional Vision Transformers (cvt-13)



Evolucion de validation accuracy por época

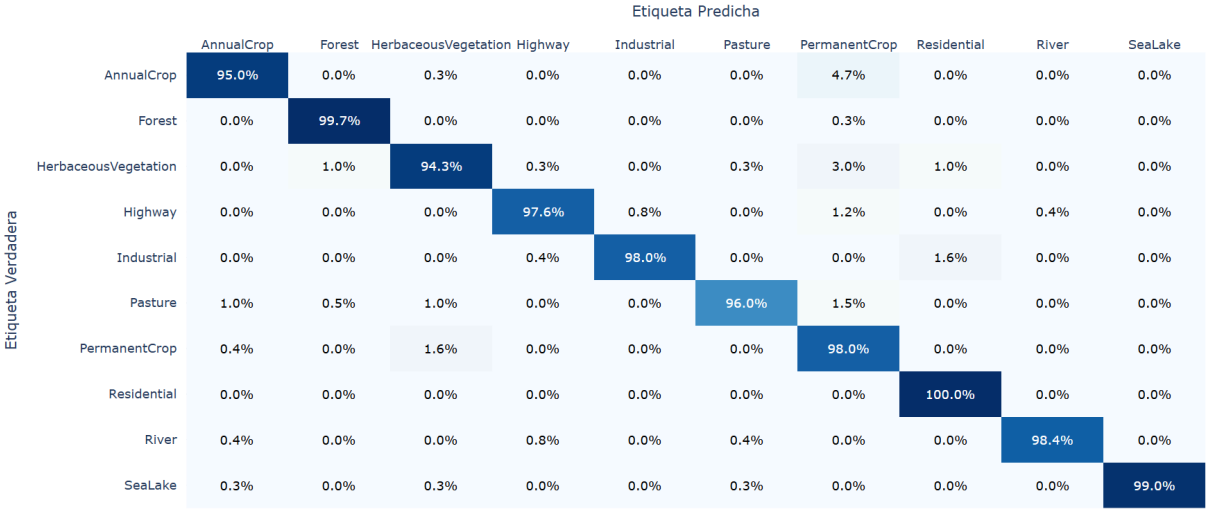


Evolucion de la funcion de perdida por época

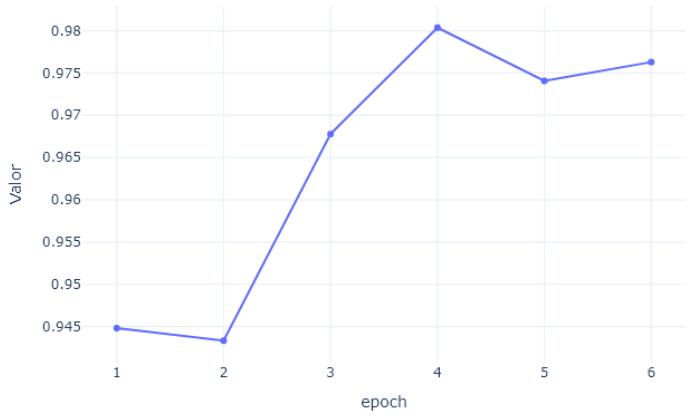


3. Convolutional Vision Transformers (cvt-21)

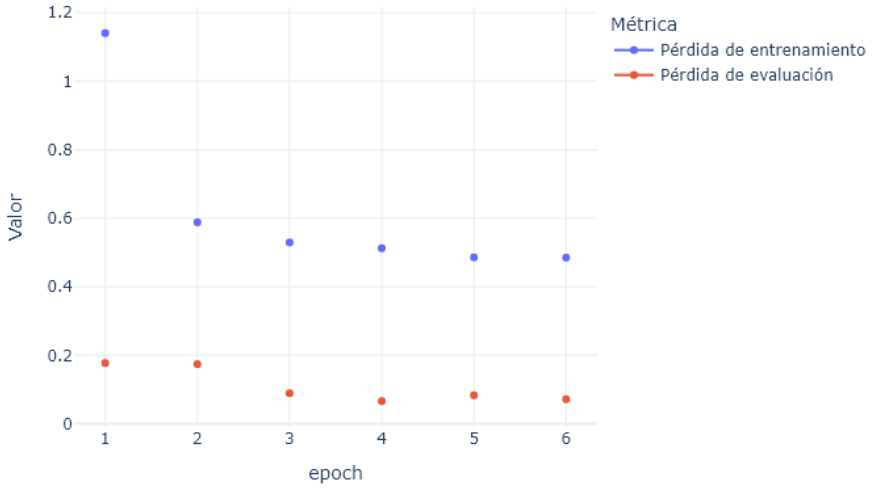
Matriz de confusion para el modelo Convolutional Vision Transformers (cvt-21)



Evolucion de validation accuracy por época

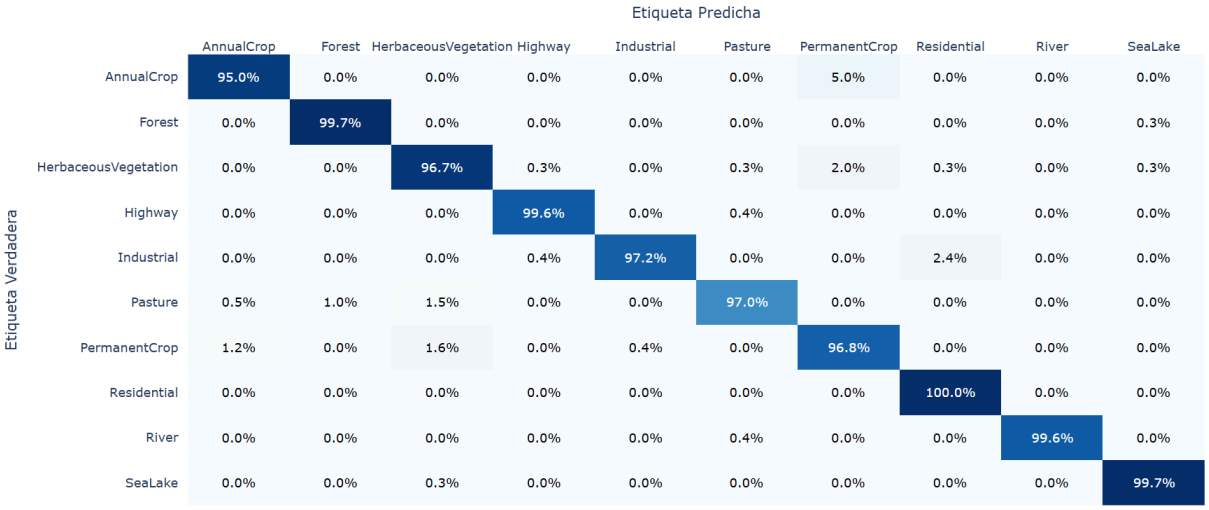


Evolucion de la funcion de perdida por época



4. You Only Look Once (YOLOv11-cls)

Matriz de confusion para el modelo You Only Look Once (YOLOv11-cls)



Evolucion de validation accuracy por época