Instituto Tecnológico de Estudios Superiores de Monterrey

TC5035 Proyecto Integrador

Semana 8 - Actividad 1
Avance 6 – Conclusiones Clave

Equipo #43

Ronald David Zuniga Sánchez A01686240

09/Junio/2024

Introducción

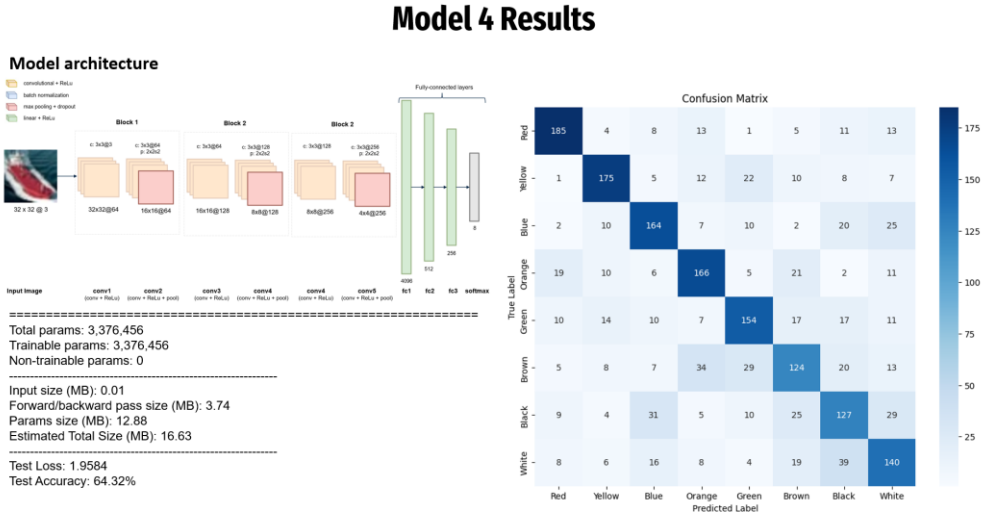
El presente reporte resume los resultados del proyecto **Dominant Color Detection Model**, con el objetivo de evaluar los resultados obtenidos en la etapa de investigación, la comparación del mejor modelo observado y su viabilidad para la implementación en entornos de producción. El primer objetivo es determinar si el modelo actual es implementable o si necesitamos retroceder a fases anteriores, como el modelado o la preparación de datos, para asegurar su efectividad. También se propondrá un entorno de producción adecuado que garantice la confiabilidad, escalabilidad y eficiencia del modelo para su uso en aplicaciones en tiempo real.

Análisis del Modelo Seleccionado

En la siguiente tabla se resumen los resultados del entrenamiento y evaluación comparativo de los modelos propuestos:

Criteria	Model 1: Base Twin Network Model	Model 2: Single Network Model	Model 3: Updated Twin Network Model	Model 4: VGG-16 Based Single Network	Model 5: Transfer Learning VGG16
Architecture Summary	Complex multi-layer network with dual paths	Simpler, more compact network	Enhanced version of Model 1 with additional dropout and batch normalization layers	Based on VGG-16, deep with multiple convolutions and fully connected layers	Fixed Feature Extractor Model through transfer learning on VGG16 ImageNet
Total Parameters	22,067,784	1,275,720	9,660,040	3,376,456	134,293,320
Validation Loss (final)	2.122	1.822	1.266	1.899	1.9565
Test Loss (avg)	2.327	1.840	1.343	1.958	2.0398
Validation Accuracy (max)	66.25%	61.82%	66.77%	64.95%	65.62%
Test Accuracy (avg)	63.18%	59.27%	64.48%	64.32%	65.16%
Best Model Size (MB)	86.29	4.98	37.75	13.19	524.59
CPU Inference Time (ms)	9.215	2.644	3.881	3.444	75.46
CUDA Inference Time (ms)	8.08	1.3	3.067	2.515	147.96
Max Memory Usage (CUDA Mem)	23.98 Mb	7.00 Mb	20.00 Mb	28.00 Mb	1.6 Gb

Dadas las métricas de rendimiento y las consideraciones de recursos, el **Modelo 3** se selecciona como el mejor modelo considerando que ofrece el mayor equilibrio entre recursos y rendimiento. Proporciona una precisión del 64,48% junto con demandas de recursos razonables, lo que lo hace adecuado para aplicaciones que requieren un buen equilibrio entre rendimiento y eficiencia.

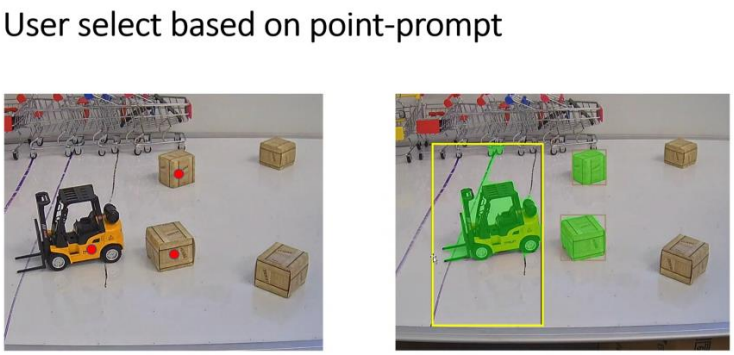


¿El rendimiento del modelo es lo suficientemente bueno para su implementación en producción?

Aún no. Considerando que no cumple con todos los parámetros clave de rendimiento solicitado por el sponsor/negocio, se observa que aún hay oportunidad de mejora en el modelo para cumplir con los requerimientos establecidos para implementar el modelo en borde (en una cámara de seguridad):

1. Accuracy >= 90% (No cumple)
    a. Modelo 3 cuenta con un Test Accuracy del 64.48%.
2. Inference time <= 10 ms (Cumple)
    a. Modelo 3 cuenta con un tiempo de inferencia promedio de:
        i. 3.44 ms en CPU
        ii. 2.515 en GPU
3. Parameter count <= 1M (after pruning and quantization) (No cumple)
    a. Modelo 3 cuenta con 3M de parámetros previo a etapas de optimización.

Sin embargo, el modelo muestra potencial para ser utilizado como un elemento de UX en el proceso de entrenamiento de modelos para la selección de color dominante de objetos de manera asistida. Ejemplo “User select based on point-prompt”:



¿Existe margen para mejorar aún más el rendimiento?

Sí bien se cumple con 1 de los 3 requerimientos mínimos para considerar el modelo listo para implementación en borde, existe margen de mejora en el rendimiento de precisión (accuracy) y exploración de técnicas de optimziación de modelos. Considerando las estrategias descritas en la siguiente sección (Acciones a Futuro), se propone un plan para mejorar el rendimiento del modelo.

¿Cuáles serían las recomendaciones clave para poder implementar la solución?

- Considerando que el objetivo es crear un modelo que pueda generalizar sobre diversos casos de uso y posiciones de captura de imagen (por medio de implementación en borde en cámaras de seguridad) se recomienda optimizar al máximo la precisión de predicción de color dominante y colores complementarios.
- Adicionalmente, se recomienda aprovechar el caso de uso de inferencias y entrenamiento dentro de plataforma Flex AI, para la creación de un dataset variado y enfocado a los casos de uso de los usuarios, lo cual permitirá futuros ejercicios de re-entrenamiento del modelo para la mejora de su rendimiento.

¿Qué tareas / procedimientos son accionables para las partes interesadas (stakeholders)?

Considerando que las partes interesadas para este proyecto involucran a los equipos de producto y R&D de Hanwha Vision, las acciones directas de los stakeholders corresponden a supervisión y guía continua en el desarrollo de mejora del rendimiento del modelo. Adicionalmente, considerando la experiencia en optimización de modelos para ejecución en borde, también se asigna las acciones directas de optimización del modelo final para ejecución en los dispositivos.

Acciones a Futuro

Se proponen 3 estrategias para mejorar el rendimiento en la continuación del desarrollo de la investigación a futuro:

1. **Mejora del Dataset.**
   - Mejora en categorización: El conjunto de datos utilizado se produjo utilizando una técnica mixta de creación de etiquetas utilizando OpenClip y validando el color dominante con Kmeans clustering. Sin embargo, aún así se observan algunos datos ambiguos o mal categorizados que pueden estar impactando en el rendimiento del modelo y su capacidad de generalizar.
   - Data Augmentation: Mejorar el conjunto de datos con más variaciones como diferentes iluminaciones, ángulos y cambios de fondo podría ayudar a mejorar la robustez y la generalización del modelo.
   - Datasets alternativos: El dataset utilizado es el de CIFAR10 con imágenes de 32x32 pixeles. El tamaño pequeño de las imágenes puede estar influyendo en el rendimiento del modelo considerando la información limitada para la generación y abstracción de características en las capas convolucionales. La exploración de modelos alternativos como la reevaluación del uso del dataset COCO 2017 pueden apoyar a mejorar la calidad de información para entrenamiento.

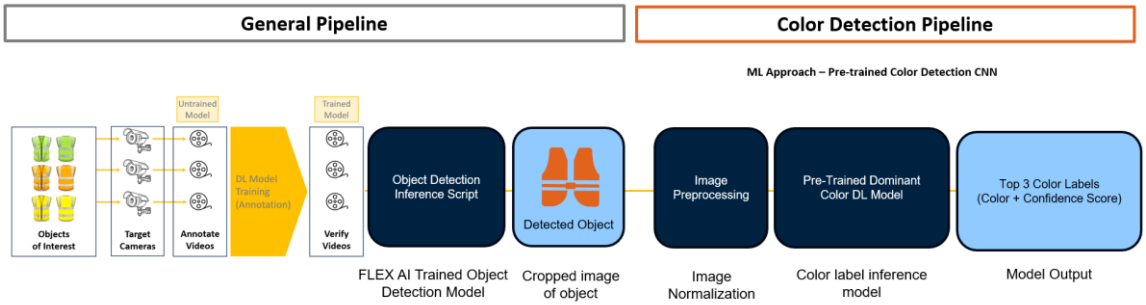
2. **Mejora del Modelo.**
   - Optimización de Hiperparámetros: Ajustar la tasa de aprendizaje, el tamaño del lote, el número de épocas y otros hiperparámetros mediante métodos como la búsqueda en cuadrícula o la optimización bayesiana podría llevar a mejores resultados.
   - Modificaciones Arquitectónicas Avanzadas: Incorporar mecanismos de atención o conexiones residuales podría ayudar al modelo a enfocarse mejor en características relevantes y mejorar la eficiencia del aprendizaje.

3. **Arquitecturas Alternativas**
   - Técnicas de Ensemble: Combinar las predicciones de múltiples modelos (incluidos aquellos que no fueron elegidos como el mejor modelo) a menudo puede mejorar el rendimiento predictivo sobre cualquier modelo individual.
   - Modelos de Segmentación en lugar de arquitecturas de clasificación como YOLO para la exploración de comportamiento en el aprendizaje y discernimiento de características de color.

Plataforma de Implementación

La plataforma de implementación para este proyecto será Amazon Web Services (AWS) considerando lo siguiente:

El modelo de **Detección de Color Dominante** será parte del pipeline del modelo final que la plataforma Flex AI entrena para la detección de objetos personalizados.



La plataforma de Flex AI opera actualmente sobre servicios de AWS. Específicamente utilizando SageMaker para la administración de flujos de entrenamiento, validación y complicación de modelos en nube.

Descripción de pipeline actual en AWS:

1. Flex AI UI: La interfaz de usuario donde el usuario inicia el proceso de entrenamiento del modelo.

2. Custom Event Bus & EventBridge Rules: Estos componentes actúan como mediadores y distribuidores de eventos, permitiendo la comunicación entre diferentes servicios y disparando acciones basadas en eventos específicos.

3. Start Training Lambda: Una función Lambda que se activa para iniciar el proceso de entrenamiento. Puede estar configurada para preparar y verificar los datos antes de enviar la tarea de entrenamiento.

4. Repository & Private Elastic Container Registry: Almacena el código y las imágenes del contenedor que se utilizarán para el entrenamiento. Aquí se aloja el modelo YOLO v4 PyTorch.

5. SageMaker: Servicio de AWS utilizado para entrenar modelos de machine learning. En este paso, SageMaker ejecuta el contenedor de entrenamiento en una instancia con GPU para procesar el entrenamiento del modelo.

6. Save Trained Model: Una vez completado el entrenamiento, el modelo entrenado se guarda en un registro de modelos o un sistema de almacenamiento.

7. Compile Lambda: Función Lambda que se invoca para iniciar la compilación del modelo entrenado, preparándolo para la implementación en producción o en dispositivos específicos.

8. SageMaker Neo: Servicio utilizado para optimizar modelos para una variedad de plataformas de hardware, asegurando que el modelo sea eficiente en términos de tiempo de inferencia y consumo de recursos.

9. Save Compiled Model: El modelo compilado y optimizado se guarda para su posterior uso o despliegue.

Este flujo de trabajo está altamente automatizado, utilizando servicios serverless de AWS como Lambda y SageMaker, lo que facilita la escalabilidad, la gestión y el mantenimiento del sistema. Además, la arquitectura aprovecha los eventos y las reglas de AWS para automatizar el flujo de trabajo en respuesta a diferentes triggers, lo que aumenta la eficiencia y reduce la necesidad de intervención manual.

Fuentes:

1. Arquitectura interna de Flex AI en AWS **(Confidencial)**
2. Servicio de machine learning - Amazon SageMaker (2024). Retrieved from https://aws.amazon.com/es/sagemaker/