Trabajo Práctico Final

Redes Neuronales aplicadas a Visión

Introducción

En el trabajo práctico final deberán abordar una problemática mediante el desarrollo de modelos de aprendizaje profundo basados en conjuntos de datos reales.

El TPF se realizará de manera **individual o en grupos de hasta dos estudiantes**.

La cátedra provee una lista de problemas a abordar de los cuales podrán elegir.

De manera excepcional, la cátedra permitirá que aborden un tema que no sea uno de los propuestos por la cátedra, el cual requerirá la aprobación del Profesor a cargo.

El TPF tendrá una nota numérica del 0 al 10. Está nota se basará en la evaluación que hagan los docentes del **Informe**, **Código Fuente** y la presentación de un **Poster**.

La fecha de presentación de pósters en “el cubo”:
Viernes 13 de Diciembre

La fecha de entrega del informe:

Lunes 16 de Diciembre

Cómputo en la Nube

Para desarrollar los modelos cada estudiante tendrá acceso a 1 cupón de “Google Cloud” mediante el cual podrán utilizar tiempo de cómputo en la nube.

Es responsabilidad del estudiante utilizar el tiempo de cómputo correctamente, el cual debe ser utilizado exclusivamente para el desarrollo del trabajo.

En el Campus Virtual podrán encontrar slides informativas y una guía detallada sobre cómo utilizar “Google Cloud” para la creación de una Máquina Virtual con GPUs.

Se recomienda **utilizar Google Colab** (gratuito) **para desarrollar** los modelos, prototipar y **realizar pruebas pequeñas**, para luego llevar los modelos a los servidores de Google Cloud.

Informe y Entregable

Deberán presentar un informe de **no más de 6 páginas** exponiendo las técnicas y decisiones tomadas a lo largo del trabajo.

El informe debe tener al menos 5 secciones, **introducción**, **método, conjuntos de datos**, **resultados** y **discusión**. Deberán explicar los métodos utilizados detallando arquitecturas de redes neuronales utilizadas, así como también dar detalles del conjunto de datos sobre el que trabajaron. El nivel de detalle debe ser acorde con la extensión máxima del informe considerando que deben exponer imágenes de los resultados obtenidos.

Pueden utilizar implementaciones existentes liberadas por investigadores del área. En esos casos, deberán explicar las fuentes, sus detalles y particularidades. En todos los casos deberán proponer al menos una modificación, mejora o tipo de uso ideado por ustedes.

El entregable debe contener:

- **Informe**.
- **Código fuente** utilizado para resolver el problema
- **Enlaces o referencias a los *datasets* utilizados**
- **Póster**

Póster

La diagramación del Poster queda a entera discreción de los estudiantes, pero deberá tener el “Título” y “Autores” del mismo en la parte superior. Se le dará un puntaje al Póster en base a la claridad de la presentación gráfica y oral, y del dominio que demuestren los estudiantes sobre los temas de su TPF.

El póster debe ser impreso en **tamaño A0 con orientación vertical**, y la impresión es responsabilidad de los estudiantes. Se recomienda buscar el servicio de impresión con anticipación para evitar contratiempos.

- Pueden crear su póster utilizando herramientas como LaTeX, PowerPoint, Keynote, Canva, Illustrator, entre otras.
- Se recomienda usar un template para facilitar el diseño.
- El póster debe ser de **tamaño A0 con orientación vertical.**

1. Planificación de Póster

- **Mensaje principal**: Antes de empezar el diseño, definan cuál es el mensaje clave que quieren transmitir. ¿Qué es lo más importante que quieren que la audiencia recuerde?
- **Audiencia**: Consideren a quién va dirigido el póster para ajustar el nivel de detalle y la terminología.

2. Estructura del Póster

La diagramación del Poster queda a entera discreción de los estudiantes, pero deberá tener el “Título” y “Autores” del mismo en la parte superior.

- **Título**
  - Debe ser claro, conciso y reflejar el contenido del trabajo.
  - Es importante que el título represente la extensión del trabajo, más allá de la tarea básica, evitando títulos genéricos como “Estimación de Precios de Alquiler en AMBA” y enfocándose en las particularidades de su extensión para diferenciarse.
  - Utilicen un tamaño de fuente grande para que se pueda leer desde lejos.
- **Autores**
  - Incluyan sus nombres, la institución a la que pertenecen y sus emails. Es recomendable ponerlo justo debajo del título.

Las diferentes secciones no tienen que llamarse igual que en el informe ni seguir la misma estructura. Estas son sugerencias generales, que pueden adaptar libremente según las necesidades y características de su trabajo:

- **Metodología / arquitecturas de red**
    - Describan brevemente el enfoque y los métodos utilizados en su investigación.
- **Resultados**
    - Presenten sus hallazgos de manera clara y compacta.
    - Utilicen gráficos, tablas y/o imágenes para facilitar la comprensión.
- **Conclusiones**
    - Resalten los puntos más importantes y las implicaciones de sus resultados.
- **Trabajo a Futuro**
    - Indiquen posibles líneas de investigación o próximos pasos.
    - Si tuvieran otros 6 meses para trabajar en esto, ¿qué harían primero?
- **Referencias**

3. Diseño Visual

- **Colores y Tipografía**
    - Utilicen una paleta de colores coherente y tipografías legibles.
    - Eviten combinaciones de colores que dificulten la lectura.
    - Eviten usar más de 3 colores diferentes. Tengan en cuenta que demasiados colores hacen que el diseño se vea poco profesional, mientras que no tener color lo convierte en algo monótono.
    - El fondo nunca debería distraer del contenido en sí, así que eviten cualquier diseño que sea demasiado cargado.
- **Tamaños de Letras Sugeridos**
    - **Título:** 72-100 pt
    - **Autores:** 40-70 pt
    - **Títulos de Secciones (Ejemplo: Resumen):** 36-48 pt
    - **Texto Principal:** 24-36 pt
    - **Leyendas de Gráficos/Imágenes:** 18-24 pt
- **Espacio en Blanco**
    - No sobrecarguen el póster.
    - El espacio en blanco ayuda a que la información sea más accesible y menos abrumadora.
- **Gráficos e Imágenes**
    - Asegúrense de que sean de alta calidad y relevantes.
    - Incluyan leyendas para aclarar cualquier gráfico o imagen.
- **Estilo de Redacción**
    - Utilicen oraciones cortas y concisas.
    - Utilicen viñetas para facilitar la lectura y evitar párrafos extensos.
    - El póster debe ser fácil de leer y captar la atención de manera efectiva.

4. Impresión

- Imprimir en **tamaño A0 (841 x 1189 mm) en orientación vertical**.
- Planifiquen la impresión con suficiente antelación para evitar contratiempos.
- La sede del Centro de Copiado “La Copia” del Campus de UdeSA **no realiza impresiones de pósters**. Como alternativa, les sugerimos imprimir el póster en la sede principal del Centro de Copiado “La Copia” en Ciudad Universitaria (ccclacopia@gmail.com), donde estará listo en aproximadamente 2 horas. El horario de atención es de lunes a viernes de 08:00 a 21:00 hs y los sábados de 08:00 a 12:00 hs. Esto solo es una recomendación, pueden optar por imprimir el póster en el lugar que consideren más conveniente.

5. Presentación Oral

- **Preparación**: Practiquen una breve presentación oral (~ 5 min) que resuma los puntos clave de su póster.
- **Interacción**: Estén preparados para responder preguntas y discutir su trabajo con los asistentes de la sesión.

1. Clasificación de Imágenes

La investigación académica tiende a enfocarse en aspectos técnicos novedosos y generalmente muestra resultados en conjuntos de datos de referencia estándar para permitir comparaciones justas con trabajos anteriores.
En el mundo real, los despliegues de visión por computadora a menudo modifican ambas suposiciones: tienden a utilizar ideas simples y comprobadas en lugar de modelos complejos de última generación y suelen utilizar conjuntos de datos personalizados adaptados al problema en cuestión. En muchas aplicaciones prácticas de visión por computadora, la tarea de construir y curar conjuntos de datos para entrenamiento y evaluación toma mucho más tiempo y esfuerzo que el proceso de construcción de los modelos.
De todas las ideas que hemos discutido en clase, la más útil en la práctica es la clasificación de imágenes mediante aprendizaje por transferencia desde un modelo pre-entrenado. Si en el futuro aplicas visión por computadora en tu trabajo, esta es probablemente la técnica que utilizarás. En este proyecto construirás un sistema de clasificación de imágenes tal como lo harías en una situación real.

Deberás recolectar tu propio conjunto de datos de clasificación de imágenes (puede ser un *dataset* preexistente de la web), dividirlo en particiones (entrenamiento / validación / prueba) y entrenar modelos de clasificación.
Se les pide proponer una arquitectura convolucional diseñada por ustedes y comparar los resultados con un modelo obtenido mediante aprendizaje por transferencia desde redes preentrenadas en ImageNet.

Deberás explorar varias opciones y parámetros, y analizar el rendimiento de tus redes entrenadas. Además, existen técnicas avanzadas de visualización, interpretación y exploración de redes convolucionales, estas técnicas se las conoce como de "**Model Understanding**", siendo el método **Grad-CAM** (Gradient-weighted Class Activation Mapping) uno de los más conocidos.
Para este trabajo se les pide, entonces, desarrollar uno o más de estos métodos de interpretación y mostrar resultados sobre alguno de los modelos de clasificación obtenidos.

Links de información general sobre **Grad-CAM** y otros métodos de interpretación:

- En este link podrán encontrar una introducción al método **Grad-CAM**
- La librería **Captum** para PyTorch está dedicada específicamente a esto: https://pytorch.org/tutorials/beginner/introyt/captumyt.html

Artículos recomendados

Además de lo visto en clase, estos son dos de los primeros artículos sobre aprendizaje por transferencia desde redes convolucionales profundas:

- Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014. [arXiv]
- Razavian et al, “CNN Features off-the-shelf: an Astounding Baseline for Recognition”, CVPR Workshops 2014. [arXiv]

Al analizar un modelo de clasificación entrenado, un método de atribución intenta determinar qué porción de una imagen de entrada fue responsable de la decisión del modelo. Algunos artículos en esta área que podés leer son:

- Simonyan et al, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014. [arXiv]
- Selvaraju et al, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization”, IJCV 2019. [arXiv]
- Fong et al, “Understanding Deep Networks via Extremal Perturbations and Smooth Masks”, ICCV 2019. [arXiv]

Crear un Conjunto de Datos de Clasificación de Imágenes

La primera tarea en este proyecto es recolectar tu propio conjunto de datos de clasificación de imágenes para una tarea de clasificación que elijas.
Primero deberías definir un conjunto de dos o más categorías en las que quieras clasificar las imágenes. Estas categorías pueden ser amplias (p. ej., perro/gato/pez, hotdog/no-hotdog) o más específicas (por ejemplo, perro siberiano/perro salchicha). ¡Ponete creativo con las categorías que elijas! Decidí si tus categorías son mutuamente excluyentes (una imagen tendrá exactamente una etiqueta correcta) o no excluyentes (una imagen puede tener cero o más etiquetas correctas). Las categorías mutuamente excluyentes son más simples de manejar y analizar, por lo que te sugerimos que intentes idear un conjunto de categorías mutuamente excluyentes.
Una vez que hayas decidido un conjunto de categorías, necesitarás construir un conjunto de datos recolectando y etiquetando imágenes. Los conjuntos de datos más grandes suelen dar mejores resultados, pero tu conjunto de datos debería tener al menos 100 imágenes por categoría. Usá la creatividad en la fuente de tus imágenes: podés tomar fotos vos mismo, buscar imágenes en internet o pensar en otra fuente de imágenes. Sin embargo, tus imágenes no deberían provenir de un conjunto de datos de visión por computadora existente; queremos que pases por el proceso de construir tu propio conjunto de datos.

Pueden explorar la librería **Selenium** para “scrapear” imágenes de la web, algunos links útiles: How to Scrape Images from Google Using Python Selenium:

End-to-End ML, A simple Selenium image scrape from an interactive Google Image Search

Luego de recolectar y etiquetar tu conjunto de datos, deberías dividirlo en conjuntos de entrenamiento, validación y prueba. Una regla general para dividir conjuntos de datos es asignar el 70% de tus datos a entrenamiento, 10% a validación y 20% a prueba.
Esperamos que el proceso de construir y etiquetar el conjunto de datos constituya una gran parte del trabajo para este proyecto.

Tu informe debería incluir una sección que describa tu tarea de clasificación y tu conjunto de datos. Explicá qué tarea de clasificación estás tratando de resolver, por qué podría ser útil y cuáles son las categorías. Describí tu proceso de recolección y etiquetado de imágenes, y discutí cualquier desafío que enfrentaste al construir tu conjunto de datos. Finalmente, mostrá ejemplos de tu conjunto de datos para dar una idea de los tipos de imágenes que contiene.

Entrenar un Clasificador de Imágenes

Una vez que hayas recolectado un conjunto de datos, es hora de entrenar varios modelos en este conjunto. Pueden proponer una arquitectura de red neuronal diseñada por ustedes mismos, pero no deben cuidar el tiempo que le dedican a esto, ¡el transfer learning es la norma no la excepción!

Deberán utilizar torchvision, que proporciona una amplia variedad de modelos comunes. Te recomendamos comparar múltiples arquitecturas, pero como primer paso te sugerimos usar ResNet18, ResNet50 o un modelo pequeño de RegNet (p. ej., RegNetY-400MF).
Podés considerar tres maneras diferentes de entrenar un clasificador:

- Entrenar desde cero: Inicializar el modelo aleatoriamente y entrenarlo desde cero en tu conjunto de entrenamiento.
- Extracción de características: Inicializar el modelo usando pesos pre entrenados en ImageNet; extraer características del modelo y entrenar un modelo lineal sobre las características extraídas. Esto puede implementarse de varias maneras: podés usar PyTorch para agregar una nueva capa al modelo, congelar todas menos la última capa y entrenar en PyTorch; también podés extraer características del modelo y guardarlas en algún formato, luego entrenar un clasificador lineal con otro paquete como scikit-learn.
- Ajuste fino (Fine-Tuning): Inicializar el modelo usando pesos pre entrenados en ImageNet; y ajustar una porción (o todo) el modelo en tu conjunto de entrenamiento. Esto debería funcionar al menos tan bien como la extracción de características.

Es posible que necesites experimentar con diferentes optimizadores, tasas de aprendizaje, regularización L2 y estrategias de aumento de datos.
Deberán entrenar al menos 6 modelos diferentes y reportar su precisión tanto en los conjuntos de entrenamiento como de validación.
Deberán mostrar las curvas de entrenamiento para tus modelos, graficando la pérdida de entrenamiento por iteración, así como la precisión del modelo en los conjuntos de entrenamiento y validación en cada época (pasada por el conjunto de entrenamiento).
Podés intentar usar aumento en tiempo de prueba o ensambles de modelos para mejorar el rendimiento de clasificación de tus modelos individuales.

El informe deberá explicar los modelos que elegiste, describir los experimentos que realizaste y mostrar los resultados descritos arriba.

Analizar tus Modelos

Después de haber entrenado tus modelos, deberías analizarlos para intentar obtener información sobre qué tan bien están funcionando y en qué situaciones fallan. Deberías analizar al menos dos modelos diferentes: el modelo con mejor rendimiento y al menos otro modelo de tu elección.
Deberías mostrar lo siguiente:

- Ejemplos cualitativos de imágenes que fueron clasificadas correctamente e incorrectamente. Podés mostrar tanto imágenes seleccionadas manualmente para resaltar características interesantes de tu modelo, como imágenes seleccionadas al azar para dar una mejor idea del rendimiento promedio de tu modelo.
- Matriz de confusión: Una matriz 2D de categoría predicha versus categoría real, donde cada entrada muestra la fracción de imágenes del conjunto de validación que caen en esa situación; esto muestra los tipos de errores que comete tu modelo.
- Métodos de atribución: Usar uno o más métodos de atribución para dar ejemplos de qué porciones de las imágenes utiliza el modelo para tomar decisiones de clasificación.

El informe debería incluir secciones que describan y muestren cada uno de los métodos de análisis anteriores, así como cualquier otro tipo de análisis que realices.

Evaluación en el Conjunto de Prueba

Después de todo tu análisis, deberías correr tu mejor modelo en el conjunto de prueba. Si tu modelo tiene un rendimiento muy diferente en los conjuntos de validación y prueba, deberías intentar explicar por qué podría ser ese el caso.

Entregables

En resumen, en este proyecto esperamos que:

- Crees tu propio conjunto de datos de clasificación de imágenes con al menos 100 imágenes por categoría. En tu informe, discutí el problema de clasificación que elegiste resolver y cómo recolectaste el conjunto de datos. Mostrá imágenes de ejemplo del conjunto.
- Entrená al menos 6 modelos de clasificación de imágenes en tu conjunto de datos; en tu informe, mostrá curvas de entrenamiento para cada modelo (pérdida y precisión), así como una comparación de precisión en el conjunto de validación.
- Analizá el rendimiento de tus modelos mostrando ejemplos de clasificación correcta/incorrecta, matrices de confusión y métodos de atribución.
- Probá tu mejor modelo en el conjunto de prueba y discutí los resultados.

2. Super-Resolución de Imágenes Individuales

En la tarea de super-resolución de imágenes individuales, una red recibe como entrada una imagen de baja resolución y genera como salida una versión de mayor resolución de la misma (¡una imagen más grande!). Este es en general, un problema subespecíficado, es decir, hay muchas posibles imágenes de alta resolución que podrían corresponder a la misma imagen de baja resolución. Sin embargo, al entrenar en conjuntos de datos de imágenes de alta y baja resolución correspondientes, una red neuronal puede aprender patrones sobre el mundo visual que le permiten predecir una salida plausible de alta resolución para cualquier entrada de baja resolución.

De forma similar a la segmentación semántica, se puede usar una red totalmente convolucional para construir un modelo de super-resolución de imágenes individuales. La entrada del modelo es una imagen de baja resolución de forma 3 x H x W; la imagen pasa por una serie de capas de convolución y capas de aumento de escala (usando interpolación de vecino más cercano, bilineal, o convolución traspuesta) y finalmente produce una salida de alta resolución de la forma 3 x **f**H x **f**W, donde **f** es el factor de super-resolución (en general tenemos que **f**=2, **f**=3 o **f**=4). Durante el entrenamiento, la salida de la red se compara con la salida real usando una función de pérdida L2, L1 o alguna otra.

El objetivo de este proyecto es implementar y entrenar una red convolucional para super-resolución de imágenes individuales. Se espera que entrenen modelos para al menos dos factores de super-resolución (**f**=2 y otro a elección).

Para esto deberán diseñar y proponer una arquitectura inspirándose en artículos científicos académicos. Deberán mostrar resultados de entrenamiento y validación utilizando métricas particularmente relevantes para este tipo de tarea específica. Les recomendamos seguir el siguiente trabajo científico cómo guía metodológica:

SRCNN: Dong et al, “Image super-resolution using deep convolutional networks”, TPAMI 2015. [arXiv]

Artículos recomendados

A continuación listamos artículos fundacionales de la técnica de super-resolución, cada uno propone una nueva metodología y están en orden ascendente de complejidad:

- **SRCNN**: Dong et al, “Image super-resolution using deep convolutional networks”, TPAMI 2015. [arXiv]

- **FSRCNN**: Dong et al, “Accelerating the super-resolution convolutional neural network”, ECCV 2016. [arXiv]
- **SRGAN**: Ledig et al, “Photo-Realistic Single Image Super-Resolution using a Generative Adversarial Network”, CVPR 2017. [arXiv]

Pueden buscar otras soluciones en las cuales inspirarse. Se recomienda comenzar copiando una de las arquitecturas más sencillas y luego probar otras arquitecturas o cambios.

Conjunto de datos

Podés entrenar tu modelo en cualquier conjunto de datos de imágenes que elijas; los conjuntos de datos de ImageNet o COCO son opciones bastante populares. La super-resolución generalmente no requiere conjuntos de datos muy grandes para el entrenamiento.

Por ejemplo, **FSRCNN** se entrena en un conjunto de datos de 100 imágenes llamado General100 (http://mmlab.ie.cuhk.edu.hk/projects/FSRCNN.html), que podría resultarte útil para tu propia implementación. Consultá la Sección 4.1 del artículo de **FSRCNN** para obtener más detalles sobre cómo preparar tu conjunto de datos de entrenamiento.

Deberán evaluar su modelo en los conjuntos de datos **Set5** y **Set14**. Durante la evaluación, la manera precisa en la que prepararás las imágenes de baja y alta resolución es importante; por esta razón, deberías usar las versiones de los conjuntos de datos **Set5** y **Set14** de este repositorio de GitHub de Jia-Bin Huang. Tené en cuenta que existen versiones diferentes de los conjuntos de datos para cada factor de super-resolución.

Modelo

Deberías implementar un modelo similar a SRCNN o FSRCNN; seguí los detalles en los artículos lo más cerca posible.

Como una extensión opcional, también podrías considerar usar una red generativa adversarial para super-resolución como se describe en el artículo de SRGAN. Entrenar el modelo adversarialmente contra un discriminador aprendido puede producir imágenes de mayor calidad perceptual, aunque con un PSNR o SSIM ligeramente inferior en comparación con un modelo entrenado solo con pérdidas L2 o L1.

El informe debe incluir una sección que describa la arquitectura de tu modelo, las funciones de pérdida de entrenamiento y cualquier otro detalle de implementación importante de tu modelo. Deberías experimentar con distintos optimizadores, funciones de pérdidas, arquitecturas de modelo u otras variaciones

que se te ocurran. Deberías mostrar los resultados de al menos **dos** variantes diferentes del modelo para cada factor de super-resolución que consideres.

Evaluación

Deberías evaluar tu modelo tanto en imágenes no vistas de tu conjunto de entrenamiento como también en los conjuntos de datos Set5 y Set14. Podés opcionalmente elegir reportar resultados adicionales en los conjuntos de datos BSD100 o Urban100.

Deberías evaluar tus modelos utilizando las métricas relación señal-ruido pico (PSNR) y la similitud estructural (SSIM). Podés utilizar las implementaciones de estas métricas de scikit-image.

Deberías comparar tu modelo con el aumento de escala usando interpolación bilineal, bicúbico y de vecinos más cercanos; podés usar OpenCV o scikit-image para esto. Tu mejor modelo debería superar ambos métodos de referencia en todos los factores de super-resolución que consideres.

Concretamente, para un factor de super-resolución de 2 deberías alcanzar un PSNR de al menos 34 en Set5; para un factor de super-resolución de 3 deberías alcanzar un PSNR de al menos 31 en Set5; y para un factor de super-resolución de 4 deberías alcanzar un PSNR de al menos 29 en Set5.

Canales de Color

Un detalle importante de los métodos de super-resolución es el canal de color que el modelo utiliza para el entrenamiento y la evaluación. Hasta ahora en este curso, todas las imágenes se han representado en el espacio de color RGB estándar; sin embargo, existen otros espacios de color que se utilizan comúnmente para representar imágenes. Uno importante es el espacio de color YCbCr. En este espacio de color, Y es un canal de color en escala de grises (luma) y Cb y Cr son dos canales que dan el color de la imagen (croma). Podés convertir entre RGB y YCbCr usando funciones de OpenCV o scikit-image.

Para la super-resolución de imágenes individuales, una configuración común es transformar la imagen del espacio de color RGB al espacio de color YCbCr; la red neuronal recibe el canal Y de baja resolución y predice un canal Y de alta resolución, mientras que los canales Cb y Cr se amplían mediante interpolación bicúbica. La pérdida de entrenamiento compara los canales Y predicho y el real, y durante la evaluación calculamos PSNR y SSIM entre los canales Y predicho y real. Deberías seguir esta convención en tu implementación. También es posible entrenar modelos de super-resolución usando entradas y salidas en color; para más detalles, consultá la Sección 4.5 del artículo de SRCNN. Podés realizar experimentos adicionales con canales de color adicionales de forma opcional.

Conclusión

Para resumir, en este proyecto se espera que:

- Implementen modelos para super-resolución de imágenes individuales. Deberías entrenar modelos para al menos dos factores de super-resolución diferentes (2x, 3x o 4x). Para cada modelo deberías mostrar curvas de entrenamiento (pérdida en el conjunto de entrenamiento y PSNR en un conjunto de validación no visto a lo largo del entrenamiento). Para cada modelo deberías reportar el rendimiento en los conjuntos de datos Set5 y Set14 usando PSNR y SSIM.
- Deberían comparar tu modelo con el aumento de escala con interpolación bilineal, bicúbico y vecinos más cercanos, y tu modelo debería superar estos métodos de referencia.
- Para factores de super-resolución 2x / 3x / 4x, deberías lograr PSNR en Set5 de al menos 34 / 31 / 29.
- También deberían mostrar resultados cualitativos. Por ejemplo, podés mostrar la imagen de baja resolución, la imagen de alta resolución real y la imagen de alta resolución predicha para cada método que consideres (y los métodos bilineal / bicúbico / vecinos más cercanos) para varios ejemplos de un conjunto de validación no visto, así como para cada imagen en Set5.

3. Segmentación de patologías en imágenes médicas

Uno de los casos de aplicación más exitosos de las redes convolucionales es la de segmentación de imágenes médicas para detección de patologías. Numerosos estudios han demostrado el potencial de los modelos de aprendizaje profundo en la detección de cáncer, clasificación de tejidos, identificación de diagnósticos estructuras llamativamente relevantes e incluso infiriendo subtipos genéticos.

La particularidad de este tipo de problemas es que, a diferencia de la clasificación, se busca asignar una etiqueta a cada píxel de la entrada.

Con los años se han propuesto diferentes arquitecturas y estrategias para resolverlo, en este trabajo deberán explorar dos arquitecturas fundacionales:

1. Uno de los primeros enfoques para realizar la segmentación semántica mediante una red neuronal profunda se propuso en el artículo de CVPR de 2015 Fully Convolutional Networks for Semantic Segmentation (FCN, Redes totalmente convolucionales para la segmentación semántica).

2. Del desarrollo y mejora de las redes FCN, se propuso luego una de las arquitecturas convolucionales más exitosas, la arquitectura conocida como U-Net: Convolutional Networks for Biomedical Image Segmentation

En base a la arquitectura U-Net una gran explosión de propuestas han surgido en la historia de las redes neuronales. Una que podemos mencionar que se enfoca en aplicar Transfer Learning utilizando una red VGG es la TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation

Para este trabajo deberán seleccionar un *dataset* de segmentación y analizar el desempeño de las dos arquitecturas fundacionales mencionadas anteriormente y proponer una variación diseñada por ustedes.

Conjuntos de datos

La cátedra recomienda dos *datasets* de segmentación de imágenes:

1. PanNuke Dataset: Uno de los *datasets* más grandes y mejor organizados con 205.343 células cancerígenas marcadas en varios diferentes tipos de tejidos:
   Repositorio Kaggle
   Torrent libre del *dataset*
   Paper de su última extensión en 2020
   Presentación del trabajo en conferencia internacional

2. Figshare Brain Tumor Dataset (una alternativa de **tamaño moderado**):
   El conjunto de datos contiene 3064 pares de imágenes de resonancia magnética cerebral y su respectiva máscaras binarias que indican el tumor:
   Repositorio Figshare
   Repositorio Kaggle

Alternativamente pueden buscar algún otro que les sea de interés. Pueden inspeccionar soluciones preexistentes que les faciliten el tratamiento y procesamiento de los datos, en todos los casos deben citar fuentes y analizar las metodologías aplicadas.

En caso de obtener buenos resultados pueden proponer su solución en Kaggle.

4. Monitoreo de vida silvestre utilizando Imágenes Termales

La utilización de drones (aeronaves no tripuladas) se han convertido en herramientas eficaces para el monitoreo y la conservación de la vida silvestre. La detección y clasificación automatizada de animales mediante inteligencia artificial (IA) ha demostrado ser muy precisa y reduce sustancialmente los costos logísticos.

En este trabajo deberán trabajar sobre el conjunto de datos Aerial Wildlife Image Repository de la universidad de Mississippi. Este conjunto de datos está compuesto por tres subconjuntos para el monitoreo de tres tipos de animales distintos: vacas, ciervos y caballos; y está compuesto por una colección de imágenes RGB, imágenes termales y coordenadas de los animales en las imaǵenes etiquetadas por biólogos expertos.

Las cámaras utilizadas para conformación del dataset son cámaras estéreo con dos sensores (DJI Zenmuse XT2), un sensor de imágenes RGB y otro Termal, por este motivo se lo conoce como un conjunto de datos RGB-T. Las cámaras fueron calibradas previamente y el conjunto de datos provee imágenes rectificadas, es decir, las imágenes termales se encuentran *warpeadas* para coincidir perfectamente con las imágenes RGB (un determinado objeto/animal visto en la imagen RGB se encontrará en la misma coordenada en la correspondiente imagen Termal).

Detección de objetos utilizando YOLO

Para este trabajo deberán implementar una solución utilizando la famosa arquitectura de detección de objetos **YOLO** introducida por Redmon, Joseph y Farhadi, Ali en 2018, You Only Look Once: Unified, Real-Time Object Detection.

A modo histórico les dejamos el enlace a la web original mantenida por Redmon, https://pjreddie.com/darknet/yolo/, el cual siempre tuvo un atractivo particular para la comunidad hacktivista, destacando el tono humorístico de Redmon en sus publicaciones.

Con los años la arquitectura YOLO fue cambiando y se fue mejorando por la comunidad, siendo hoy Roboflow el principal mantenedor:
What is YOLO? The Ultimate Guide

Se advierte que **versiones posteriores a la YOLOv4 podrían presentar componentes en su arquitectura que no hemos visto en la materia**.
Material de referencia de la YOLOv4: How to Train YOLOv4 on a Custom Dataset

La particularidad del trabajo reside en modificar YOLO para trabajar con imágenes RGB y Termales, de manera de mejorar la detección de objetos correlacionando esta información.

YOLOv4-RGB-T

El reciente trabajo científico: Adopting the YOLOv4 Architecture for Low-Latency Multispectral Pedestrian Detection in Autonomous Driving analiza la adaptación de la red YOLOv4 (solo RGB) para la detección de peatones utilizando imágenes RGB y Termales.

El trabajo realiza un estudio exhaustivo de varias alternativas para la adaptación de la arquitectura YOLOv4.

Primeramente analiza la implementación de una versión YOLOv4-T (Sección 3.2, *YOLOv4 with Thermal Images*) que considera únicamente las imágenes termales en una versión de YOLO de 1 canal. Esto es posible modificando la cantidad de canales de los filtros de la primera capa convolucional (de 3 canales a solo 1). Pero esto no considera las imágenes RGB.

Luego el artículo propone una serie de alternativas para fusionar la información RGB y Termal:

1. YOLO4-HST y YOLO4-GST: Comprime la información RGB-T en solo tres canales de manera de conformar con el pipeline habitual de entrenamiento de la red de 3 canales.

2. YOLO4-RGB-T Fusion: La información termal es incluida en un cuarto canal de información, se realiza el entrenamiento reutilizando los pesos de la red YOLOv4-RGB original para todas las partes comunes de la arquitectura.

3. YOLO4-Late Fusion: Aplica las versiones YOLO4-RGB y YOLO4-T (de 3 canales y solo 1 canal) en paralelo para comparar el solapamiento de las detecciones al final del *pipeline*.

4. YOLO4-Middle Fusion: Finalmente propone una solución superadora con un mayor grado de complejidad correlacionando la información Termal en capas intermedias de la red.

Los autores proveen solo la implementación de la solución llamada YOLOv4-Middle.

Se aconseja idear una metodología de exploración del problema aplicando en orden progresivo de complejidad las soluciones.
La exploración de una solución entre: YOLO-RGB, YOLO-T y YOLO4-HST/YOLO4-GST para el conjunto de datos particular de su problema, ya es suficiente para la aprobación del trabajo.

Un orden podría ser empezar con el entrenamiento de una red YOLO-RGB y YOLO-T por separado para el conjunto de datos específico sobre el que están trabajando, y luego explorar la solución YOLO4-HST / YOLO4-GST.

Dejar para el final, en caso de éxito, las soluciones YOLO4-RGB-T Fusion y YOLO4-Late Fusion. En caso de considerar la YOLOv4-Middle, hacerlo solo en modo de inferencia.

5. UdeSA 3D: *Structure-from-Motion*

En este trabajo podrán aplicar contenidos vistos en la primera mitad de la materia aplicados a la estimación de movimiento libre de una cámara en el espacio y la reconstrucción 3D del entorno observado.

La universidad de Maryland propone todos los años a sus estudiantes reconstruir una porción de la universidad utilizando una cámara monocular:
Buildings built in minutes - An SfM Approach

Esta descripción de la solución permite realizar una reconstrucción del entorno sin un objeto conocido en la escena.

En este trabajo deberán calibrar una cámara y tomar imágenes de la universidad, para luego implementar la solución de estimación de movimiento y reconstrucción 3D.

Pueden solicitar las cámaras estéreo de la cátedra para la realización del trabajo, teniendo que adaptar los algoritmos (de monocular a estéreo).
NOTA: la metodología se vuelve más sencilla en el caso estéreo dado que es posible triangular puntos con un solo frame de la cámara.

Se les pide comparar resultados con el sistema de SfM COLMAP, pudiendo también aplicar métodos de reconstrucción de redes neuronales cómo SfMLearner (Unsupervised Learning of Depth and Ego-Motion from Video), DeepSFM (DeepSFM: Structure From Motion Via Deep Bundle Adjustment).

Habiendo realizado una estimación de movimiento de toda la trayectoria de la cámara podrían realizar la reconstrucción 3D de la escena aplicando una variante de las novedosas neural radiance fields. Siendo las BAFs: Bundle-Adjusting Radiance Fields capaces de trabajar con poses de cámara ruidosas (implementación).

6. Propuesta de trabajo por estudiantes

Los grupos que se decidan por la opción de proponer un trabajo práctico no listado en los puntos anteriores, deberán proponer un problema a resolver que involucre los temas vistos en la materia.

Es obligatorio que el trabajo involucre la implementación de una arquitectura de redes neuronal. Pudiendo realizar un trabajo de análisis del estado del arte y la utilización de soluciones preexistentes en “modo inferencia”

Es un requisito que el tema sea revisado y aprobado previamente por el Profesor a cargo.

Para estructurar mejor el trabajo, se sugiere que el mismo siga el ciclo de desarrollo de un proyecto de inteligencia artificial el cual cuenta de los siguientes pasos:

1. Definición del problema a resolver
2. Estado del arte y trabajos relacionadas
3. Definición de un conjunto de datos
4. Creación del modelo
5. Evaluación del modelo y sus hiper parámetros
6. Despliegue y optimización

Tener especial cuidado en sobre-dimensionar la solución ni querer abarcar un problema demasiado grande. Lo mejor es un problema de tamaño moderado planificando una solución que les permita aprender de una determinada área o arquitectura.

1. Definición del problema a resolver

Para esto se deben plantear claramente los objetivos y metas a alcanzar durante el desarrollo del trabajo práctico. Además, se deben plantear hipótesis y proponer una metodología de trabajo que permita alcanzar los objetivos.

El problema a resolver tiene que poder definirse concretamente, debe ser acotado y plausible de implementar en menos del 30% del tiempo designado para la entrega.

2. Estado del arte y trabajos relacionadas

Es menester que el planteo del tema y las propuestas de resolución se basen en bibliografía y artículos publicada con referato, es decir, no cuentan como estado del arte páginas web Wikipedia, Medium, TowardsDatascience, ElRiconDelVago, Blogspot, posteos en Likdin o redes sociales, etc. En estos últimos casos, se recomiendo buscar siempre la referencia original y los artículos a dónde se hace mención.

Tener especial cuidado con ArXiv. En muchos casos, los artículos subidos a esta plataforma no fueron revisados todavía (no tienen referatos). Los autores de los trabajos suelen subir versiones preliminares de los artículos previas al envío a una revista o conferencia, en las cuales el trabajo podría ser rechazado, por lo que se recomienda hacer una búsqueda en internet para verificar el estado de publicación de los mismos.

Una fuente de trabajos para la exploración del estado del arte es Google Scholar: https://scholar.google.com/

3. Definición del conjunto de datos

Una de las tareas más importantes de este proyecto es la creación o definición del conjunto de datos de imágenes (o video) a utilizar. Prestar especial atención, durante la confección del estado del arte, a menciones sobre dataset públicos o sitios web donde los autores liberan los conjuntos de datos.

El dataset de imágenes a utilizar deberá estar estrechamente ligado al problema a resolver y contar con anotaciones que permitan extraer métricas de evaluación durante la evaluación y testeo del modelo.

Pueden tomar fotos ustedes mismos, buscar imágenes en internet, pensar en alguna otra fuente de imágenes como kaggle, huggingface, paperswithcode, etc. Tener en cuenta, en caso de utilizar un dataset propio o con anotaciones distintas a las necesarias, que el grupo deberá agregar manualmente las etiquetas deseadas en un número significante de las mismas (este número dependerá del problema a resolver, consultar con el Profesor a cargo).

Luego de recolectar y etiquetar tu conjunto de datos, deberían dividirlo en conjuntos de entrenamiento, validación y prueba. Una regla general para dividir conjuntos de datos es asignar el 70% de tus datos a entrenamiento, 10% a validación y 20% a prueba.

4. Creación del modelo

El desarrollo del modelo se deberá realizar en Python y se recomienda utilizar la librería PyTorch. Otras librerías pueden utilizarse si el grupo de trabajo lo considera pertinente pero la cátedra no se compromete a brindar soporte sobre las mismas.

Se espera que el modelo sea entrenado sobre el conjunto de datos propuesto anteriormente. Pueden considerar tres maneras diferentes de entrenarlo:

- **Entrenar desde cero**: Inicializar el modelo aleatoriamente y entrenarlo desde cero en tu conjunto de entrenamiento.
- **Extracción de características**: Inicializar el modelo usando pesos preentrenados, extraer características del modelo y entrenar un modelo lineal sobre las características extraídas. Esto puede implementarse de varias maneras: podés usar PyTorch para agregar una nueva capa al modelo, congelar todas menos la última capa y entrenar en PyTorch; también podés extraer características del modelo y guardarlas en algún formato, luego entrenar un clasificador lineal con otro paquete como scikit-learn.
- **Ajuste fino (Fine-Tuning)**: Inicializar el modelo usando pesos preentrenados en ImageNet; ajustar todo el modelo en tu conjunto de entrenamiento. Esto debería funcionar al menos tan bien como la extracción de características.

Es posible que necesites experimentar con diferentes optimizadores, tasas de aprendizaje, regularización y estrategias de aumento (augmentation) de datos

Se deberían mostrar las curvas de entrenamiento para los modelos, graficando la función de pérdida de entrenamiento por iteración, así como la precisión del modelo en los conjuntos de entrenamiento y validación en cada época.

5. Evaluación del modelo y sus hiperparámetros

Analizar el rendimiento de los modelos propuesto mostrando ejemplos de clasificación o detección correcta/incorrecta, matrices de confusión y todo

lo visto en clase. Discutir los resultados de los mejores modelos en el conjunto de pruebas y analizar cuán bien generalizan los mismos.

También se desea mostrar resultados cualitativos para analizar en detalle los casos buenos, malos y patológicos. Es necesario que propongan hipótesis o ideas de de cómo se comportan modelos de manera analizando los resultados de forma visual.

6. Despliegue y optimización

Se desea que se incluye claramente el hardware utilizado tanto para el entrenamiento como para el momento de prueba o inferencia. Es decir, se debe especificar donde va a desplegar el modelo desarrollado para su ejecución (GPU, CPU, Mobile, Browser, etc).

Indicar además, el tiempo consumido para el entrenamiento (total y por época) junto con el tiempo de procesamiento de inferencia por imagen.