Temario del Curso

Clasificador de imágenes con redes neuronales

Universidad Nacional Autónoma de México

Facultad de Ciencias

Materia: Proyecto I

Carrera: Matemáticas Aplicadas (Plan 2017)

Grupo: 6013
Modalidad: Presencial
Horario: Lunes a Viernes, 17:00 a 18:00

Profesor: José Eduardo Rodríguez Barrios
Ayudante: Francisco Pérez Carbajal

Objetivo del Seminario-Taller

Este seminario-taller tiene como objetivo que los estudiantes **desarrollen un proyecto de titulación** en el área de aprendizaje profundo aplicado a la visión por computadora, explorando tanto arquitecturas convolucionales clásicas (CNNs) como modelos modernos basados en atención (Vision Transformers) y enfoques multimodales.
A través de la lectura, síntesis y apropiación de conocimiento de artículos científicos y textos especializados, combinado con el desarrollo práctico de un proyecto, los estudiantes:

- **Investigarán y sintetizarán** el estado del arte en modelos de aprendizaje profundo para visión por computadora, desde CNNs hasta Vision Transformers y modelos fundacionales
- **Desarrollarán** un proyecto completo de clasificación de imágenes que pueda servir como trabajo de titulación
- **Implementarán** una solución end-to-end: desde el procesamiento de datos hasta el despliegue
- **Experimentarán** con diferentes arquitecturas (CNNs, ViTs, modelos híbridos) para comparar rendimiento y eficiencia
- **Documentarán** su trabajo con rigor académico y técnico
- **Presentarán** sus avances y resultados de forma profesional

Alcance y Enfoque

Este curso cubre el **espectro completo de arquitecturas modernas** para visión por computadora:

- **Redes Neuronales Convolucionales (CNNs):** Desde arquitecturas fundacionales (LeNet, AlexNet, ResNet) hasta modelos eficientes (EfficientNet, MobileNet)
- **Vision Transformers (ViTs):** Modelos basados en mecanismos de atención (ViT, DeiT, Swin Transformer)
- **Modelos fundacionales y multimodales:** CLIP, DINOv2, SAM y otros modelos pre-entrenados a gran escala
- **Aprendizaje auto-supervisado:** Técnicas modernas como MAE, SimCLR y DINO
- **Arquitecturas híbridas:** Modelos que combinan convoluciones y atención

Los estudiantes explorarán estas diferentes familias de modelos, compararán sus fortalezas y debilidades, y seleccionarán las arquitecturas más apropiadas para su proyecto específico.

Formato del Curso

El curso se estructura como un **seminario-taller** donde:

- Los estudiantes leen y presentan artículos científicos relevantes
- Se discuten conceptos, técnicas y resultados de investigaciones actuales
- Cada estudiante desarrolla su propio proyecto de clasificación de imágenes
- Las sesiones incluyen revisión de avances, retroalimentación y mentoría
- El proyecto final puede ser utilizado como trabajo de titulación

Pre-requisitos

Para un aprovechamiento óptimo del curso, se requiere que los participantes cumplan con los siguientes requisitos:

Conocimientos Matemáticos

- **Álgebra lineal:** vectores, matrices y transformaciones lineales
- **Cálculo:** diferencial e integral
- **Probabilidad y estadística:** conceptos básicos
- **Regresión logística:** deseable experiencia previa

Habilidades Técnicas

- Capacidad para leer y entender documentación técnica y artículos en inglés
- Conocimientos básicos de programación en Python
- Familiaridad con control de versiones (Git/GitHub)

Herramientas de Apoyo

- Se utilizará un **aula virtual de Google Classroom** como apoyo para el curso
- Cada alumno deberá subir su código de tareas y proyectos a un **repositorio en GitHub**
- Se recomienda tener una cuenta activa en GitHub

Estructura del Seminario-Taller

El seminario-taller se organiza en torno a **cuatro ejes principales**: lectura y discusión de literatura científica, desarrollo del proyecto de titulación, presentaciones de avances, y documentación técnica.

Fase 1: Fundamentos y Estado del Arte (Semanas 1–4)

Actividades de lectura y síntesis:

- **Arquitecturas fundacionales:** LeCun et al. (1998) - LeNet, Krizhevsky et al. (2012) - AlexNet, He et al. (2016) - ResNet
- **Vision Transformers:** Dosovitskiy et al. (2021) - ViT, Touvron et al. (2021) - DeiT, Liu et al. (2021) - Swin Transformer
- **Aprendizaje auto-supervisado:** Chen et al. (2020) - SimCLR, He et al. (2022) - MAE
- Síntesis del estado del arte 2020-2025: transición de CNNs a Transformers
- Elaboración de fichas bibliográficas y mapas conceptuales

Desarrollo del proyecto:

- Definición del problema de clasificación a resolver (dominio de aplicación)
- Búsqueda y selección de datasets apropiados
- Análisis exploratorio de datos (distribución de clases, sesgos potenciales)
- Configuración del entorno de desarrollo (PyTorch/TensorFlow)
- Establecimiento de baselines con arquitecturas clásicas (CNNs) y modernas (ViTs)

Fase 2: Procesamiento de Datos y Experimentación (Semanas 5–8)

Actividades de lectura y síntesis:

- **Data augmentation:** Shorten & Khoshgoftaar (2019) - Survey on image data augmentation
- **Transfer learning y fine-tuning:** Tan & Le (2019) - EfficientNet, Touvron et al. - DeiT distillation
- **Modelos fundacionales:** Radford et al. (2021) - CLIP, Oquab et al. (2023) - DINOv2
- Análisis de casos de estudio en dominios específicos

Desarrollo del proyecto:

- Implementación del pipeline de procesamiento de datos con augmentation
- Experimentación con transfer learning usando modelos pre-entrenados (ImageNet, CLIP)
- Comparación de familias de arquitecturas: CNNs (ResNet, EfficientNet) vs. ViTs (DeiT, Swin)
- Exploración de modelos fundacionales y fine-tuning especializado
- Uso de MLOps: experiment tracking (Weights & Biases, MLflow)
- Documentación de experimentos y resultados comparativos

Fase 3: Optimización, Evaluación y Fairness (Semanas 9–12)

Actividades de lectura y síntesis:

- **Eficiencia:** Howard et al. (2019) - MobileNetV3, técnicas de quantization y pruning
- **Interpretabilidad:** Selvaraju et al. (2017) - Grad-CAM, análisis de atención en ViTs
- **Fairness:** Buolamwini & Gebru (2018) - Gender Shades, Meta AI (2023) - FACET benchmark
- **Robustness:** Madry et al. (2018) - Adversarial training, estudios sobre distribution shifts
- Métricas de evaluación y mejores prácticas

Desarrollo del proyecto:

- Ajuste de hiperparámetros para las arquitecturas seleccionadas (grid search, Bayesian optimization)
- Evaluación rigurosa: accuracy, precision, recall, F1-score, confusion matrices
- Análisis de fairness: evaluación de sesgos en subgrupos demográficos
- Interpretabilidad específica por arquitectura: Grad-CAM para CNNs, attention maps para ViTs
- Análisis comparativo de eficiencia computacional y rendimiento entre arquitecturas
- Análisis de errores y diagnóstico del modelo
- Comparación con baselines y trabajos relacionados del estado del arte

Fase 4: Deployment, MLOps y Documentación Final (Semanas 13–16)

Actividades de lectura y síntesis:

- **Model serving:** BentoML, TensorFlow Serving, ONNX Runtime
- **MLOps:** Data versioning (DVC), monitoring (Evidently AI), continuous learning

- **Casos de estudio:** Tesla Autopilot MLOps, Uber Michelangelo platform
- **Consideraciones éticas:** Deployment responsable, privacidad, fairness en producción

Desarrollo del proyecto:

- Selección y optimización del modelo final basado en experimentos comparativos
- Optimización para producción: quantization, pruning, conversión a ONNX
- Desarrollo de API REST para servir el modelo (FastAPI/Flask + Docker)
- Implementación de monitoring y drift detection
- Creación de interfaz de usuario/demo (Gradio/Streamlit, opcional)
- Documentación técnica completa: justificación de elección de arquitectura, comparaciones experimentales
- Preparación del reporte final (formato de tesis: 50–80 páginas)
- Presentación final del proyecto ante el grupo

Evaluación del Seminario-Taller

La evaluación se centra en el **desarrollo del proyecto de titulación** y la **apropiación de conocimiento** a través de la lectura y síntesis de literatura científica.

Actividad	Fase	Descripción	%
Presentaciones de Papers	1	Exposición y discusión de artículos científicos asignados. Debe incluir síntesis, análisis crítico y conexión con el proyecto personal.	20 %
Avances del Proyecto	2–3	Entregas parciales del proyecto en cada fase: (1) Propuesta y análisis de datos, (2) Pipeline y modelo base, (3) Optimización y evaluación, (4) Implementación completa.	30 %
Documentación Técnica	1–4	Reporte técnico progresivo que documenta el proyecto con formato de tesis. Incluye introducción, estado del arte, metodología, experimentos y resultados.	20 %
Proyecto Final	4	Proyecto completo de clasificación de imágenes con: código documentado, modelo entrenado, API/interfaz, reporte final y presentación. Debe ser de calidad para titulación.	30 %

Cuadro 1: Distribución de la evaluación del seminario-taller

Componentes del Proyecto de Titulación

El proyecto final debe ser un trabajo completo y de calidad suficiente para servir como tesis, incluyendo:

1. **Revisión bibliográfica** exhaustiva del estado del arte
2. **Planteamiento del problema** claro y bien motivado
3. **Metodología** detallada y justificada científicamente
4. **Implementación** completa y reproducible (código en GitHub)
5. **Experimentación** rigurosa con análisis de resultados
6. **Comparación** con trabajos relacionados y baselines
7. **Conclusiones** y trabajo futuro
8. **Producto funcional** (API, aplicación web, etc.)
9. **Reporte final** con formato de tesis (50–80 páginas)

Criterios de Evaluación de Presentaciones

- Comprensión profunda del artículo(s) presentado(s)
- Calidad de la síntesis y capacidad de explicar conceptos complejos
- Análisis crítico de fortalezas y limitaciones
- Conexión con el proyecto personal y aplicabilidad
- Participación activa en discusiones

Políticas del Seminario-Taller

- La calificación mínima aprobatoria es de 60 %
- Se requiere asistencia mínima del 80 % y participación activa
- Los avances deben entregarse en las fechas establecidas
- El proyecto es individual, aunque se fomenta la colaboración y discusión
- Cada estudiante debe mantener un repositorio de GitHub actualizado
- Las presentaciones de papers son obligatorias y se asignan rotativamente
- Se espera honestidad académica; el plagio resultará en reprobación automática

Temas de Ayudantía

Las sesiones de ayudantía están diseñadas para complementar el contenido teórico del curso con habilidades prácticas y herramientas esenciales para el desarrollo profesional de proyectos de ML. Se organizan en 5 módulos temáticos:

Módulo 1: Fundamentos de Herramientas

1. Git y GitHub desde cero

- Conceptos básicos: repositorio, commit, branch, push, pull, merge
- Creación de cuenta GitHub y configuración inicial (SSH keys)
- Primer repositorio: estructura y organización para proyectos de ML
- Archivo `.gitignore` para proyectos de ML (modelos, datos, checkpoints, `__pycache__`)
- Flujo de trabajo básico: add, commit, push
- Comandos esenciales que se usarán durante el semestre
- Resolución de conflictos comunes

2. Markdown y documentación técnica

- Sintaxis básica de Markdown: encabezados, listas, código, enlaces, imágenes
- Estructura de un README.md profesional
- Documentación de experimentos y resultados en notebooks
- Plantilla de README para el proyecto del curso
- Tablas y fórmulas matemáticas en Markdown
- GitHub Pages básico (opcional para portafolio personal)

3. Entorno de desarrollo Python

- **Gestores de paquetes modernos:** `venv` vs `uv` (Astral) - ventajas y casos de uso
- Instalación y configuración de `uv`
- Gestión de dependencias: `requirements.txt` y `pyproject.toml`
- Creación de ambientes virtuales con `uv venv`
- Instalación rápida de paquetes con `uv pip`

- Jupyter Notebooks vs scripts (`.py`): cuándo usar cada uno
- Estructura de directorios profesional para proyectos de ML
- Buenas prácticas: nomenclatura, modularización, separación de código y datos

4. Google Colab y recursos en la nube

- Configuración de Google Colab (gratuito y Pro)
- Conexión con Google Drive para persistencia de datos
- Instalación y gestión de librerías (PyTorch, Hugging Face, etc.)
- Gestión de sesiones y uso eficiente de GPU/TPU
- Límites y mejores prácticas de uso
- Alternativas gratuitas: Kaggle Notebooks, Paperspace Gradient

Módulo 2: APIs y Servicios Web

5. Consumo de APIs REST

- ¿Qué es una API REST? Conceptos fundamentales
- Métodos HTTP: GET, POST, PUT, DELETE
- Librería `requests` en Python
- Manejo de autenticación y headers
- Procesamiento de respuestas JSON
- Ejemplo práctico: consumo de API pública

6. Creación de APIs con FastAPI

- Instalación y configuración de FastAPI
- Estructura básica de una aplicación
- Definición de endpoints y modelos de datos (Pydantic)
- Testing local con Uvicorn
- Documentación automática (Swagger UI)
- Integración con modelos de ML
- Consumo de la API desde otras aplicaciones

Módulo 3: Hugging Face Ecosystem

7. Primeros pasos con Hugging Face

- Introducción al ecosistema Hugging Face
- Navegación del Hub de modelos y datasets
- Carga de modelos preentrenados con `transformers`
- Primeras inferencias: clasificación de imágenes y texto
- Exploración y carga de datasets con `datasets`
- Autenticación y uso de modelos privados

8. Uso avanzado de modelos preentrenados

- Fine-tuning vs Transfer Learning: conceptos clave
- Selección del modelo adecuado según la tarea
- Configuración de hiperparámetros
- Uso de `pipeline` para inferencia rápida
- Optimización de modelos: cuantización y pruning básico
- Manejo de diferentes modalidades (visión, texto, multimodal)

Módulo 4: Despliegue y Producción

9. Docker y contenedores para ML

- **Conceptos básicos:** imágenes, contenedores, volúmenes, redes
- Instalación de Docker y Docker Desktop
- Creación de `Dockerfile` para proyectos de ML
- Gestión de dependencias y layers
- **Best practices:**
  - Multi-stage builds para reducir tamaño
  - Cache de capas para builds rápidos
  - Archivo `.dockerignore` optimizado
- Push a Docker Hub o GitHub Container Registry
- Docker Compose para orquestar múltiples servicios

10. Despliegue de modelos en producción

- Opciones de deployment: Hugging Face Spaces, Render, Railway
- Containerización de API con modelo integrado
- Variables de entorno y configuración
- Monitoreo básico y logs
- Consideraciones de costo y escalabilidad

Módulo 5: Presentación de Resultados

11. Preparación de la presentación final (LaTeX)

- Estructura de presentación técnica
- Visualizaciones efectivas
- Ensayo y retroalimentación

Objetivo de las Ayudantías

Las ayudantías proporcionan una formación práctica integral en herramientas modernas de desarrollo, permitiendo a los estudiantes:
- Gestionar proyectos de ML de forma profesional
- Trabajar con frameworks y plataformas del estado del arte
- Desplegar modelos en producción
- Documentar y presentar resultados de manera efectiva

Estas habilidades complementan el contenido teórico del curso y son esenciales para el desarrollo exitoso del proyecto de titulación.

Bibliografía

Artículos Científicos Fundamentales (Papers para Presentar)

Arquitecturas Fundacionales (Era CNN)

1. **LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P.** (1998). *Gradient-based learning applied to document recognition.* Proceedings of the IEEE.
   Paper fundacional sobre redes convolucionales y LeNet.
2. **Krizhevsky, A., Sutskever, I., & Hinton, G. E.** (2012). *ImageNet classification with deep convolutional neural networks.* NeurIPS.
   AlexNet: inicio de la era moderna de deep learning en visión.
3. **He, K., Zhang, X., Ren, S., & Sun, J.** (2016). *Deep residual learning for image recognition.* CVPR.
   ResNet: conexiones residuales que permiten redes muy profundas.
4. **Tan, M., & Le, Q.** (2019). *EfficientNet: Rethinking model scaling for convolutional neural networks.* ICML.
   Escalamiento eficiente con compound scaling (depth, width, resolution).

Vision Transformers y Modelos Fundacionales

5. **Dosovitskiy, A., et al.** (2021). *An image is worth 16×16 words: Transformers for image recognition at scale.* ICLR.
   Vision Transformer (ViT): transformers puros aplicados a visión por computadora.
6. **Touvron, H., et al.** (2021). *Training data-efficient image transformers.* ICML.
   DeiT: hace ViT práctico con knowledge distillation, 85.2 % en ImageNet-1K.
7. **Liu, Z., et al.** (2021). *Swin Transformer: Hierarchical vision transformer using shifted windows.* ICCV (Best Paper).
   Complejidad lineal con shifted windows, dominante para detección y segmentación.
8. **Kirillov, A., et al.** (2023). *Segment Anything.* ICCV.
   SAM: modelo fundacional para segmentación, entrenado en 1.1B máscaras.

Aprendizaje Auto-Supervisado

9. **Chen, T., et al.** (2020). *A simple framework for contrastive learning of visual representations.* ICML.
   SimCLR: aprendizaje contrastivo con grandes batch sizes y augmentations.
10. **He, K., et al.** (2022). *Masked autoencoders are scalable vision learners.* CVPR.
    MAE: enmascara 75 % de patches, 87.8 % ImageNet, 3× más rápido que contrastivos.

11. **Oquab, M., et al.** (2023). *DINOv2: Learning robust visual features without supervision.* DINOv2: entrenado en 142M imágenes, features robustos para percepción estructurada.

12. **Caron, M., et al.** (2021). *Emerging properties in self-supervised vision transformers.* ICCV.
DINO: auto-distilación sin labels, emergencia de segmentación semántica.

Modelos Multimodales

13. **Radford, A., et al.** (2021). *Learning transferable visual models from natural language supervision.* ICML.
CLIP: 400M pares imagen-texto, zero-shot transfer revolucionario.

14. **Li, J., et al.** (2022). *BLIP: Bootstrapping language-image pre-training.* ICML.
Unifica comprensión y generación visión-lenguaje con CapFilt.

Eficiencia y Deployment

15. **Howard, A., et al.** (2019). *Searching for MobileNetV3.* ICCV.
MobileNetV3: 75.2 % ImageNet con 5.4M parámetros, optimizado para móviles.

16. **Shorten, C., & Khoshgoftaar, T. M.** (2019). *A survey on image data augmentation for deep learning.* Journal of Big Data.
Revisión exhaustiva de técnicas de data augmentation.

Interpretabilidad, Fairness y Robustness

17. **Selvaraju, R. R., et al.** (2017). *Grad-CAM: Visual explanations from deep networks via gradient-based localization.* ICCV.
Interpretabilidad: visualización de regiones importantes para predicción.

18. **Buolamwini, J., & Gebru, T.** (2018). *Gender shades: Intersectional accuracy disparities in commercial gender classification.* FAT.
Revela sesgos en sistemas comerciales: 20-34 % disparidad de error.

19. **Madry, A., et al.** (2018). *Towards deep learning models resistant to adversarial attacks.* ICLR.
Adversarial training con PGD, fundamento de robustness adversarial.

Libros de Texto y Referencia

20. **Goodfellow, I., Bengio, Y., & Courville, A.** (2016). *Deep Learning.* MIT Press.
Referencia fundamental sobre deep learning (disponible en línea).

21. **Chollet, François.** (2021). *Deep Learning with Python, 2nd Edition.* Manning Publications.
Guía práctica con Keras/TensorFlow.

22. **Szeliski, Richard.** (2022). *Computer Vision: Algorithms and Applications, 2nd Edition.* Springer.
Texto completo sobre visión por computadora.

23. **Prince, Simon J. D.** (2023). *Understanding Deep Learning.* MIT Press.
Introducción moderna y accesible al deep learning.

Recursos Adicionales

24. **Papers With Code:** `https://paperswithcode.com/`
Plataforma con papers, código e implementaciones.

25. **arXiv:** `https://arxiv.org/` (secciones cs.CV, cs.LG)
Repositorio de preprints en Computer Vision y Machine Learning.

26. **PyTorch & TensorFlow Documentation**
Documentación oficial de los frameworks principales.