Apuntes semana 11 Clase #2

Eder Vega Suazo
Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
IC-6200 - Inteligencia Artificial Gr2

***Resumen*—Este documento condensa la segunda lección de la semana 11 centrada en autoencoders y su aplicación a imágenes y texto. Se explican la estructura encoder–espacio latente–decoder, variantes prácticas (denoising, VAE, under/overcomplete) y arquitecturas relacionadas (U-Net, skip-connections). Se discuten tareas y aplicaciones: reducción de dimensionalidad, detección de anomalías, super-resolución y segmentación, además de la transición a representaciones de texto (tokenización y embeddings) y modelos de lenguaje. El apunte incluye recomendaciones experimentales y criterios de evaluación prácticos orientados a la implementación de proyectos y a la replicación de resultados.**

***Index Terms*—Autoencoder, VAE, Denoising, Reducción de dimensionalidad, Tokenización, Embeddings, U-Net, Detección de anomalías.**

I. Introducción

Este documento sintetiza los conceptos trabajados en la sesión sobre arquitecturas basadas en redes convolucionales aplicadas a autoencoders y la extensión hacia representaciones para texto. El documento ofrece una guía práctica con definiciones, fórmulas y recomendaciones operativas para la implementación de experimentos en imágenes y texto. Se sugiere acompañar este documento con las figuras referenciadas para facilitar la comprensión de arquitecturas y visualizaciones de espacios latentes.

II. Repaso de la clase

La sesión inició con un repaso de los temas vistos anteriormente, en los que se introdujeron los fundamentos de los autoencoders y su relación con las redes convolucionales. Se recordó que estas arquitecturas son una aplicación directa de las CNN en un contexto no supervisado, donde el objetivo principal es reconstruir la entrada original a partir de una representación comprimida. El profesor enfatizó que, a diferencia de los modelos de clasificación, los autoencoders no utilizan etiquetas externas, sino que aprenden de los propios datos, permitiendo capturar patrones y regularidades internas.

Durante el repaso, se analizó la estructura general de un autoencoder compuesta por un **encoder**, un **espacio latente** y un **decoder**. El encoder transforma la entrada en una representación de menor dimensionalidad que concentra la información esencial; el decoder, a su vez, reconstruye la imagen a partir de esa representación. Este proceso de codificación y decodificación se comparó con una forma de "compresión aprendida" donde el modelo decide qué información conservar y cuál descartar.

El profesor destacó además las aplicaciones prácticas revisadas: la reducción de dimensionalidad como alternativa a métodos tradicionales, la detección de anomalías mediante el análisis del error de reconstrucción, y la restauración de imágenes afectadas por ruido o baja resolución. Finalmente, se repasó el concepto de **espacio latente continuo**, introducido en los autoencoders variacionales (VAE), el cual permite generar nuevas muestras mediante la interpolación entre puntos del espacio latente, estableciendo así la base para los modelos generativos que se profundizarían en la sesión actual.

III. Apuntes de clase

III-A. Organización y avisos

- Las revisiones del proyecto serán presenciales, el profesor avisó a los apuntadores faltantes que lo tomen en cuenta ya que una semana se deberá de apartar para la revisión del proyecto.
- Próximamente habrá dos entregables principales: un ejercicio práctico con autoencoders (imágenes) y una tarea más compleja sobre texto y agentes, esta puede que valga más porcentaje. Ya que esta implica planificar experimentos y validaciones con tiempo para revisiones en laboratorio.
- **Nota:** Zoom limita la validez de los enlaces; el profesor generó un link que ya está en el grupo de Telegram para las lecciones y la duración típica de la sesión es $\sim$40 min, cuando acabe hay que ingresar nuevamente en el mismo link.

III-B. Autoencoders: idea y componentes

Un autoencoder aprende una función $f : x \mapsto \hat{x}$ donde $\hat{x}$ intenta aproximarse a x. Internamente:

- **Encoder:** transforma x en $z = g_\theta(x)$.
- **Espacio latente:** z es un vector de baja dimensión que condensa características relevantes.
- **Decoder:** reconstruye $\hat{x} = h_\phi(z)$.

En imágenes el encoder usa convoluciones y pooling para reducir resolución y aumentar canales. El decoder usa operaciones de upsampling o convolución transpuesta para recuperar la forma espacial. Ver Figura 1 para un esquema general de encoder/decoder.

III-C. Entrenamiento y funciones de pérdida

El objetivo del entrenamiento es reducir la diferencia entre la entrada original y la reconstrucción que produce el modelo. Los elementos que generan las señales evaluadas por la

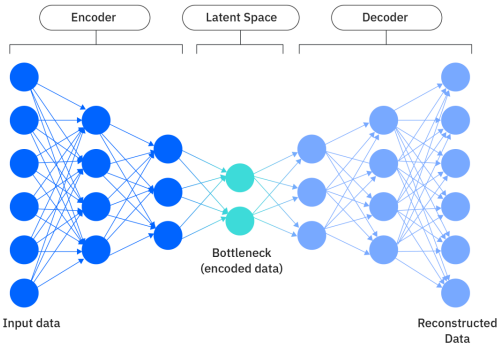

Figura 1: Esquema general de encoder, espacio latente y decoder.

función de pérdida son el **encoder** —que produce el vector latente o, en variantes probabilísticas, los parámetros de una distribución— y el **decoder** —que genera la reconstrucción $\hat{x}$ a partir de ese latente.

III-C0a. Pérdida de reconstrucción.: Es la medida principal que compara la entrada y la salida del autoencoder. Su función es indicar cuánto error comete el modelo al reconstruir. Según el tipo de datos y su normalización se elige la forma práctica de esta pérdida:

- Para imágenes normalizadas en $[0, 1]$ es común usar una pérdida basada en la comparación píxel a píxel (mencionada en clase como la opción directa). El profesor comparó esta técnica con el enfoque que usábamos en regresión para penalizar diferencias entre valores.
- Para imágenes con valores binarios o interpretadas como probabilidades se emplea una pérdida adecuada a ese caso (la alternativa binaria que se mencionó en la presentación).

Lo que se obtiene con esta pérdida es un indicador directo de calidad de reconstrucción. En aplicaciones como detección de anomalías se usa ese error (o una métrica derivada) para decidir si una muestra es atípica.

III-C0b. Regularización en VAE (pérdida adicional).: En la variante variacional el encoder no entrega un vector determinista sino parámetros de una distribución en el espacio latente. Además de la pérdida de reconstrucción, se incorpora un término que obliga a que la distribución latente siga una referencia (el profesor lo describió como forzar una distribución continua, por ejemplo, normal). Ese término de regularización:

- Proviene directamente de los parámetros que calcula el encoder (media y dispersión en la clase).
- Su propósito es estructurar el espacio latente para que sea continuo y muestreable, permitiendo interpolación y generación controlada.

III-C0c. Notas prácticas (mencionadas en clase).:

- Normalizar los píxeles al rango adecuado facilita la elección de la pérdida.
- La elección entre comparación píxel a píxel y una pérdida para datos binarios depende del rango y la interpretación de los píxeles.
- Cuando la calidad visual importa, además de la pérdida de entrenamiento suele evaluarse la reconstrucción con métricas perceptuales (p. ej. SSIM) para complementar la evaluación numérica.

III-D. Variantes y su propósito

III-D0a. Denoising: Entrenar con $x_{ruidosa}$ y objetivo x limpio. El modelo aprende a eliminar ruido específico (ej. Salt-and-Pepper).

III-D0b. Under-/Overcomplete: Latente más pequeño obliga a comprimir; latente mayor puede memorizar en exceso.

III-D0c. VAE: Permite muestrear y hacer interpolación en un espacio continuo útil para generación. La combinación de reconstrucción y KL genera latentes con estructura estadística. Ver Figura 2 para ilustración de muestreo y espacio latente continuo.

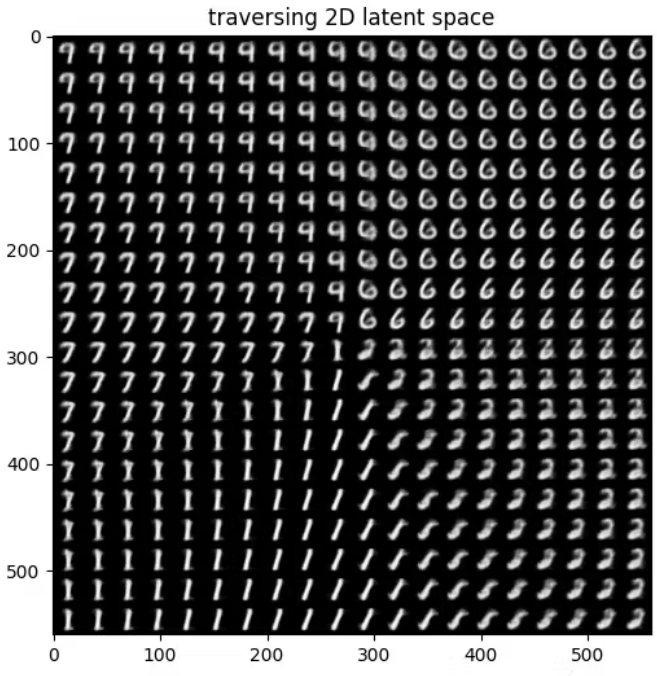

Figura 2: Representación del muestreo en VAE

III-E. Aplicaciones prácticas

III-E0a. Reducción de dimensionalidad: Guardar z en una base de datos vectorial. Comparar vectores con similitud de coseno:

$$\text{sim}(u, v) = \frac{u \cdot v}{\|u\|\|v\|}.$$

Usos: búsqueda por similitud, indexación y como entrada comprimida para clasificadores simples (KNN).

III-E0b. Detección de anomalías (ejemplo bancario): Entrenar con transacciones válidas. Para una transacción nueva x: calcular $err = L_{rec}(x, \hat{x})$. Si el error err es mayor a un umbral, lo marca como posible fraude. Selección del umbral τ mediante ROC o validación manual. Importante evaluar tasa de falsos positivos y costo operativo.

III-E0c. Denoise y super-resolution: Para super-resolution el objetivo puede ser una imagen de alta resolución x_{HR} y la entrada x_{LR}. Arquitecturas con skip-connections (U-Net style) mejoran la preservación de detalles. Se recomienda usar una figura comparativa de entrada/resultado en el informe experimental (ver Figura 3).

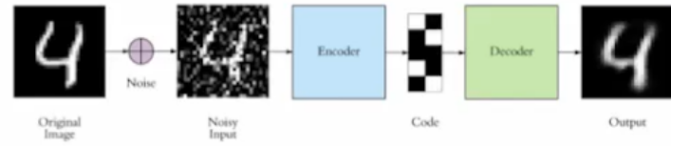

Figura 3: Ejemplo sugerido: comparación entrada ruidosa / salida reconstruida / referencia.

Nota: aplicaciones forenses (p. ej. mejora de cámaras) un compañero plantea las consideraciones legales sobre manipulación de evidencia.

III-E0d. Segmentación (U-Net): U-Net concatena mapas de características del encoder en el decoder. Esto restaura información espacial perdida por pooling y mejora mapeo de máscaras para segmentación de objetos. (Se sugiere incluir una figura de arquitectura U-Net y un ejemplo de máscara en la entrega.)

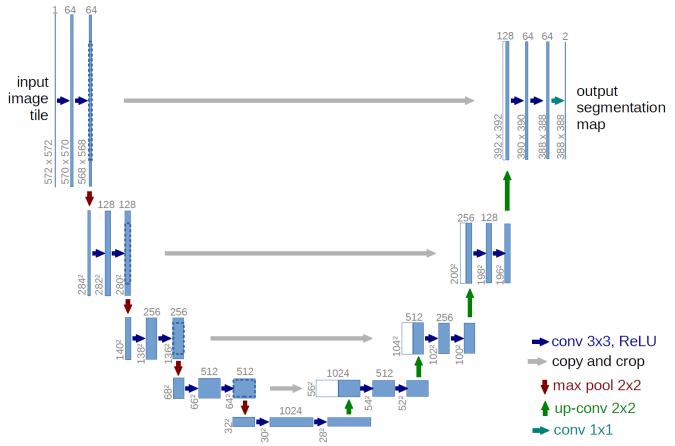

Figura 4: Representación de U-Net

III-F. Espacios latentes: visualización y utilidad (ampliado)

Visualizar z con t-SNE/UMAP facilita ver separabilidad por clases. Cuando los clusters son nítidos un clasificador simple sobre z funcionará bien. En VAE la continuidad del espacio permite interpolar entre muestras y generar imágenes plausibles no vistas. Ver Figura 5 para un ejemplo de visualización t-SNE.

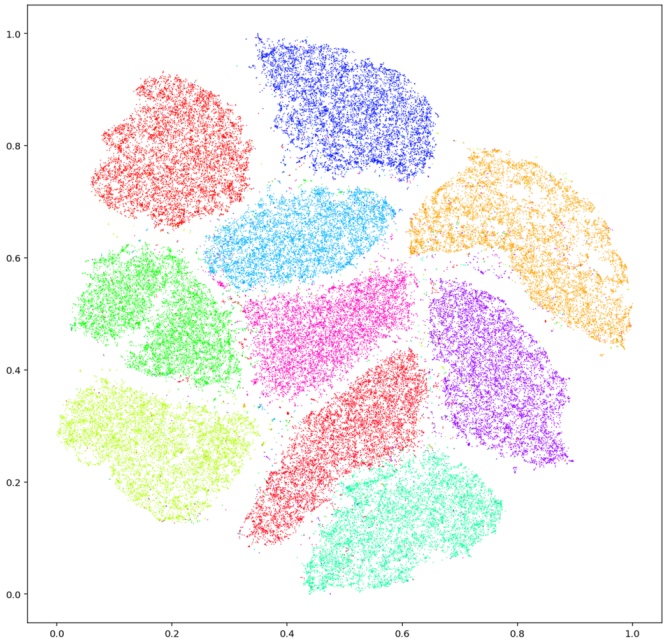

Figura 5: Ejemplo de visualización t-SNE de vectores latentes.

III-G. Transición a NLP: tokenización y embeddings

III-G0a. Tokenización: Estrategias: palabra completa, subword (BPE), carácter, bytes. Subword reduce OOV y controla longitud de secuencia. Ver Figura 6 para un esquema de tokenización subword.

III-G0b. Embeddings: Cada token se mapea a un vector $e \in \mathbb{R}^d$ mediante la capa `Embedding`. Para representar frases se puede usar promedio de embeddings o agregadores más complejos.

III-G0c. Modelos de lenguaje: Evolución: RNN/LSTM $\rightarrow$ Transformers con self-attention. La self-attention permite capturar dependencias largas y producir embeddings contextuales; esos embeddings sirven para recuperación, clasificación y agentes.

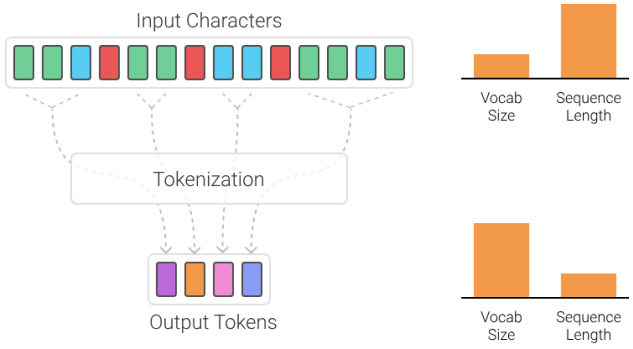

Figura 6: Esquema ilustrativo de tokenización subword y mapeo a IDs.

III-H. Recomendaciones operativas para la tarea

- Probar al menos dos configuraciones: (1) denoising autoencoder, (2) VAE con latente de prueba (p. ej. 32, 64) dependiendo de la GPU.
- Guardar checkpoints y curvas de pérdida. Evaluar MSE y SSIM.
- Para anomalías, definir umbral con validación y reportar precisión/recall.
- Para texto, experimentar tokenización subword y entrenar un embedding básico antes de usar modelos preentrenados.

IV. Conclusiones

Los autoencoders representan una herramienta fundamental dentro del aprendizaje profundo no supervisado, al permitir que un modelo aprenda representaciones compactas de los datos sin depender de etiquetas externas. Durante la sesión se destacó cómo la arquitectura encoder–decoder constituye la base para múltiples aplicaciones, desde la reducción de dimensionalidad hasta la generación y reconstrucción de imágenes. La comprensión del espacio latente resulta esencial, ya que en él se concentra la información más relevante de las entradas y se posibilita la detección de patrones, la identificación de anomalías o la generación de nuevos ejemplos a partir de distribuciones continuas como en los VAE.

Asimismo, se vio la importancia de seleccionar correctamente las funciones de pérdida y de interpretar el error de reconstrucción según el contexto de aplicación. En tareas visuales, arquitecturas como U-Net o las variantes con skip-connections amplían el potencial del modelo, mientras que en procesamiento de texto la noción de codificación latente se traslada a los embeddings y a la tokenización como pasos previos a los modelos de lenguaje. En conjunto, los autoencoders ofrecen una base conceptual y práctica para desarrollar soluciones que integren visión e información textual, avanzando hacia sistemas más autónomos e interpretativos.

Referencias

[1] Steven Pacheco P, “Autoencoder” 2025.
[2] Steven Pacheco P, “RAGs y agentes usando LLMs” 2025.
[3] Compañeros D. Clase, “11_Semana_AI_20251014_(1,2,3).,” 2025.