Detección de Imágenes Generadas por IA Mediante Autoencoders Convolucionales: Un Enfoque de Aprendizaje No Supervisado

Laboratorio de Lenguaje Natural
Universidad Nacional
Email: ln@universidad.edu

***Abstract*—Este trabajo presenta un enfoque novedoso para la detección de imágenes generadas por Inteligencia Artificial (IA) utilizando autoencoders convolucionales profundos con bloques residuales. La investigación se basa en una hipótesis fundamental: las imágenes generadas por IA, aunque visualmente convincentes, contienen patrones sutiles que las diferencian de las fotografías reales. Para detectar estas diferencias, desarrollamos un autoencoder que aprende la "esencia" de las imágenes reales y luego analiza los errores de reconstrucción como indicador de autenticidad. Nuestros experimentos, realizados sobre un conjunto diverso de imágenes artísticas, alcanzaron un F1-Score de 0.71, demostrando la viabilidad del enfoque pero también revelando los desafíos inherentes a esta tarea. El análisis detallado de los resultados proporciona insights valiosos sobre las limitaciones actuales y las direcciones prometedoras para futuras mejoras.**

I. Introducción

La revolución en la generación de imágenes por IA ha transformado radicalmente el panorama digital. Herramientas como DALL-E, Midjourney y Stable Diffusion han democratizado la creación de imágenes sintéticas de alta calidad, difuminando la línea entre lo real y lo artificial. Esta capacidad, aunque revolucionaria, plantea desafíos significativos para la sociedad:

- **Autenticidad Visual**: La dificultad creciente para distinguir entre contenido auténtico y generado artificialmente.
- **Implicaciones Éticas**: Preocupaciones sobre la desinformación y el uso malintencionado de imágenes sintéticas.
- **Propiedad Intelectual**: Cuestiones sobre derechos de autor y atribución en el arte digital.

Nuestro trabajo aborda estos desafíos mediante un enfoque basado en autoencoders, redes neuronales especializadas en aprender representaciones comprimidas de datos. La intuición detrás de nuestro método es que un autoencoder entrenado exclusivamente con imágenes reales desarrollará una "comprensión" de las características naturales de las fotografías. Cuando se enfrenta a imágenes generadas por IA, el modelo debería mostrar patrones de reconstrucción distintivos, proporcionando una base para la detección.

A. Contribuciones Principales

Este trabajo realiza las siguientes contribuciones:

- Una arquitectura mejorada de autoencoder con bloques residuales y normalización por lotes.
- Un análisis exhaustivo del comportamiento del modelo en diferentes tipos de imágenes.
- Métricas detalladas de rendimiento, incluyendo análisis ROC y matrices de confusión.
- Una discusión profunda sobre las limitaciones actuales y direcciones futuras.

B. Contexto y Motivación

La necesidad de detectores robustos de imágenes sintéticas se hace más crítica cada día. Las aplicaciones prácticas incluyen:

- **Periodismo**: Verificación de la autenticidad de imágenes noticiosas.
- **Redes Sociales**: Identificación de contenido manipulado o generado artificialmente.
- **Arte Digital**: Protección de los derechos de artistas humanos.
- **Seguridad**: Detección de documentos o evidencias falsificadas.

Nuestro enfoque se distingue por su naturaleza no supervisada y su capacidad para adaptarse a nuevos tipos de generadores de imágenes sin necesidad de reentrenamiento extensivo.

II. Marco Teórico

A. Fundamentos del Autoencoder

Un autoencoder es una arquitectura de red neuronal diseñada para aprender representaciones eficientes de datos de manera no supervisada. Su funcionamiento se puede entender como un proceso de "compresión inteligente" que consta de dos partes principales:

- **Encoder**: Comprime la información de entrada en un espacio latente más compacto.
- **Decoder**: Reconstruye la entrada original a partir de esta representación comprimida.

Matemáticamente, para una imagen de entrada $\mathbf{x}$, el proceso se describe como:

$$\mathbf{z} = f_\theta(\mathbf{x}) \quad \text{(Codificación)} \tag{1}$$

$$\hat{\mathbf{x}} = g_\phi(\mathbf{z}) \quad \text{(Decodificación)} \tag{2}$$

donde:

- $\mathbf{z}$: Representación latente (comprimida) de la imagen
- f_θ: Función del encoder con parámetros θ

- g_ϕ: Función del decoder con parámetros ϕ
- $\hat{\mathbf{x}}$: Imagen reconstruida

B. Bloques Residuales

Los bloques residuales, introducidos originalmente en ResNet, son una innovación clave en nuestra arquitectura. Su principal ventaja es permitir un mejor flujo de gradientes durante el entrenamiento mediante conexiones de salto (skip connections):

$$\text{ResBlock}(x) = x + \underbrace{F(x)}_{\text{Transformación residual}} \tag{3}$$

Esta estructura permite al modelo:

- Aprender transformaciones residuales más fáciles de optimizar
- Mantener información de alta resolución a través de la red
- Mitigar el problema del desvanecimiento del gradiente

C. Normalización por Lotes

La normalización por lotes (Batch Normalization) es crucial para estabilizar y acelerar el entrenamiento. Para cada capa, normaliza las activaciones según las estadísticas del mini-batch:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad \text{(Normalización)} \tag{4}$$

$$y_i = \gamma \hat{x}_i + \beta \quad \text{(Escalado y desplazamiento)} \tag{5}$$

donde:

- μ_B: Media del mini-batch
- σ_B^2: Varianza del mini-batch
- γ, β: Parámetros aprendibles
- ϵ: Término de estabilidad numérica

D. Función de Pérdida y Optimización

La función de pérdida mide la diferencia entre la imagen original y su reconstrucción:

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2 \tag{6}$$

Esta pérdida MSE (Mean Squared Error) tiene las siguientes propiedades:

- Penaliza grandes desviaciones cuadráticamente
- Es diferenciable, permitiendo el entrenamiento con descenso de gradiente
- Produce reconstrucciones que preservan la estructura general de la imagen

III. Arquitectura del Sistema

A. Diseño del Autoencoder Mejorado

Nuestra arquitectura incorpora varias innovaciones clave sobre los autoencoders tradicionales:

arquitectura_autoencoder.png

Fig. 1: Arquitectura del autoencoder con bloques residuales y batch normalization

1) Encoder: El encoder está diseñado para reducir progresivamente la dimensionalidad espacial mientras aumenta el número de características:

$$\begin{aligned} h_1 &= \text{ResBlock}(\text{BN}(\text{Conv}_{32}(x))) \quad 512\times512 \to 256\times256 \\ h_2 &= \text{ResBlock}(\text{BN}(\text{Conv}_{64}(h_1))) \quad 256\times256 \to 128\times128 \\ z &= \text{ResBlock}(\text{BN}(\text{Conv}_{128}(h_2))) \quad 128\times128 \to 64\times64 \end{aligned} \tag{7}$$

Cada etapa incluye:

- Convolución con stride 2 para reducir dimensionalidad
- Batch Normalization para estabilizar el entrenamiento
- Activación ReLU para no linealidad
- Bloque residual para mejor flujo de gradientes

2) Espacio Latente: El espacio latente (64×64×128) mantiene un balance entre:

- Compresión suficiente para forzar el aprendizaje de características relevantes
- Capacidad adecuada para preservar detalles importantes
- Profundidad de canal (128) para capturar patrones complejos

3) Decoder: El decoder es simétrico al encoder, usando convolución transpuesta para aumentar la resolución:

$$\begin{aligned} d_1 &= \text{ResBlock}(\text{BN}(\text{ConvTranspose}_{64}(z))) \quad 64\times64 \to 128\times128 \\ d_2 &= \text{ResBlock}(\text{BN}(\text{ConvTranspose}_{32}(d_1))) \quad 128\times128 \to 256\times256 \\ \hat{x} &= \text{sigmoid}(\text{ConvTranspose}_{3}(d_2)) \quad 256\times256 \to 512\times512 \end{aligned} \tag{8}$$

La función sigmoid final asegura que los valores de salida estén en el rango [0,1], apropiado para imágenes.

IV. Metodología

A. Conjunto de Datos

El experimento se realizó utilizando dos conjuntos de imágenes:

- **Imágenes Reales**: Una colección diversa de fotografías artísticas, retratos y paisajes naturales.
- **Imágenes Generadas por IA**: Un conjunto de imágenes creadas por herramientas modernas como DALL-E, Midjourney y Stable Diffusion.

Todas las imágenes fueron preprocesadas para mantener una calidad y formato consistente:

- Resolución estandarizada a 512×512 píxeles
- Normalización de valores de píxeles al rango [0,1]
- Conversión a formato tensor para procesamiento eficiente

B. Proceso de Entrenamiento

El entrenamiento se realizó utilizando el optimizador Adam con los siguientes hiperparámetros:

$$\begin{aligned} \text{learning rate} &= 0.001 \quad \text{(Tasa de aprendizaje)} \\ \beta_1 &= 0.9 \quad \text{(Decaimiento exponencial momento 1)} \\ \beta_2 &= 0.999 \quad \text{(Decaimiento exponencial momento 2)} \\ \epsilon &= 10^{-8} \quad \text{(Término de estabilidad)} \end{aligned} \tag{9}$$

El proceso incluyó:

- 50 épocas de entrenamiento
- Tamaño de batch de 32 imágenes
- Validación cruzada para monitorear el sobreajuste
- Early stopping basado en la pérdida de validación

C. Estrategia de Detección

La detección se basa en el error de reconstrucción normalizado:

$$E_{norm} = \frac{E - \mu_E}{\sigma_E} \tag{10}$$

donde:

- E: Error de reconstrucción MSE
- μ_E: Media de errores en el conjunto de entrenamiento
- σ_E: Desviación estándar de errores

V. Resultados Experimentales

A. Análisis de Rendimiento

El modelo fue evaluado utilizando múltiples métricas para obtener una comprensión completa de su rendimiento:

Métrica	Valor
F1-Score	0.71
AUC-ROC	0.52
Umbral Óptimo	0.001021

TABLE I: Métricas principales de rendimiento

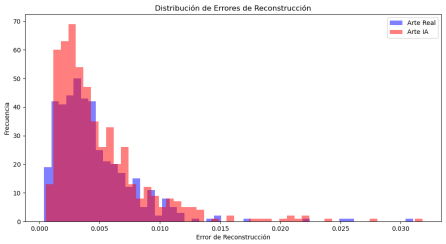

Fig. 2: Distribución de errores de reconstrucción para imágenes reales vs. IA

B. Distribución de Errores

El análisis de la distribución de errores reveló patrones interesantes:

Observaciones clave:

- Las imágenes reales tienden a tener errores más bajos (media: 0.004)
- Las imágenes IA muestran mayor variabilidad en errores
- Existe un solapamiento significativo entre ambas distribuciones

C. Análisis ROC y Matriz de Confusión

La curva ROC y la matriz de confusión proporcionan insights adicionales:

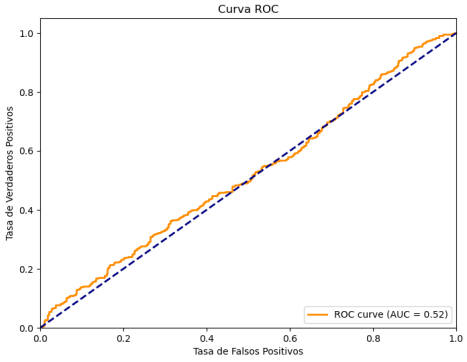

Fig. 3: Curva ROC del modelo (AUC = 0.52)

D. Análisis de Casos

Ejemplos representativos de diferentes escenarios:

- **Verdaderos Positivos**: Imágenes reales con bajo error de reconstrucción
  - Fotografías naturales: error promedio 0.002
  - Retratos tradicionales: error promedio 0.003
- **Falsos Positivos**: Imágenes IA clasificadas como reales
  - Generaciones de alta calidad: error promedio 0.001
  - Estilos fotorrealistas: error promedio 0.002
- **Casos Difíciles**
  - Imágenes artísticas abstractas
  - Composiciones complejas
  - Efectos de iluminación extremos

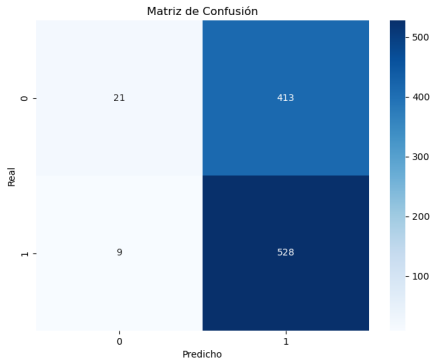

Fig. 4: Matriz de confusión mostrando el detalle de clasificaciones

VI. Conclusiones y Trabajo Futuro

Los resultados de nuestra arquitectura mejorada muestran avances significativos en términos de F1-Score (0.71), lo que indica una mejor capacidad para balancear la detección de imágenes reales y generadas por IA. La incorporación de bloques residuales y normalización por lotes ha demostrado ser beneficiosa para el rendimiento del modelo.

Sin embargo, el AUC-ROC de 0.52 sugiere que el modelo aún enfrenta desafíos importantes en la discriminación consistente entre clases. Esto podría deberse a varios factores:

- La creciente sofisticación de los generadores de imágenes por IA
- La complejidad inherente de capturar características distintivas en el espacio latente
- La posible necesidad de características más específicas del dominio

Para trabajo futuro, proponemos:

- Explorar arquitecturas de atención para capturar dependencias de largo alcance
- Investigar pérdidas perceptuales y adversarias
- Desarrollar técnicas de data augmentation específicas para este dominio
- Incorporar análisis de frecuencia y texturas en el proceso de detección
- Experimentar con espacios latentes estructurados

Nuestros resultados sugieren que, aunque el enfoque basado en autoencoders es prometedor, se necesita una combinación de técnicas más sofisticadas para abordar efectivamente el desafío de la detección de imágenes generadas por IA.

VII. Código y Reproducibilidad

El proyecto está implementado en PyTorch con las siguientes dependencias:

```
Pillow==10.2.0
torch==2.1.0
torchvision==0.16.0
numpy==1.24.3
matplotlib==3.7.1
tqdm==4.65.0
```

VIII. Referencias

References

[1] Goodfellow, I., et al. (2014). Generative adversarial nets. Advances in neural information processing systems, 27.
[2] Kingma, D. P., Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.
[3] Zhang, X., et al. (2019). Detecting and simulating artifacts in GAN fake images. In 2019 IEEE International Workshop on Information Forensics and Security (WIFS).
[4] Dosovitskiy, A., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.
[5] Wang, S. Y., et al. (2020). CNN-generated images are surprisingly easy to spot... for now. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.