

José Antonio López Saldaña
11 de Septiembre del 2025

ABSTRACT:

El presente documento presenta el desarrollo y evaluación de dos enfoques de aprendizaje profundo en clasificación de imágenes, con el objetivo de mejorar la comprensión y capacidad para la implementación de Redes Neuronales Convolucionales (CNN). En la primera etapa diseñé y entrené una CNN construida desde cero, mientras que en una segunda etapa hice uso de un modelo preentrenado (*transfer learning*).

Ambos enfoques se compararon utilizando métricas como la exactitud (accuracy), la pérdida (loss) durante el entrenamiento y, en su caso, medidas adicionales de desempeño, con el fin de analizar las ventajas y desventajas de implementar una arquitectura propia frente al uso de un modelo prefabricado ajustado a la tarea específica.

Keywords – *Clasificación de Imágenes, Deep Learning, Redes Neuronales Convolucionales (CNN), Transfer Learning, Modelo Pre Entrenado, Modelo desde Cero, Computer Vision, Aprendizaje Supervisado.*

1. Introducción

Con el crecimiento de redes sociales y más específicamente plataformas de video, el deepfake se ha convertido en un riesgo creciente. Este tipo de videos son generados con técnicas de inteligencia artificial, usados para desinformar, suplantar identidades y dañar la reputación de personas e

instituciones. Ha nacido una nueva rama de seguridad, y ahora la detección automática de deepfakes permite mitigar el impacto de contenidos manipulados protegiendo a las personas y su confianza en los medios visuales.

En este contexto, el aprendizaje profundo orientado a la clasificación de videos tiene una solución, el usar redes neuronales convolucionales (CNN), para encontrar patrones en los rostros y poder identificar uno real de uno falso.

Para este trabajo se emplea una versión del conjunto de datos **FaceForensics++**, disponible en Kaggle, que contiene videos reales y falsos generados con distintos métodos de deepfake. El dataset está organizado en varias carpetas correspondientes a diferentes técnicas de manipulación (como DeepFakeDetection, Deepfakes, Face2Face, FaceShifter, FaceSwap y NeuralTextures), además de una carpeta con videos originales y archivos CSV con metadatos. En total se incluyen alrededor de 7000 videos, comprimidos con un nivel intermedio (C23), lo que aproxima de forma realista las condiciones de distribución en línea.

La naturaleza controlada del dataset **FaceForensics++** permite entrenar y evaluar modelos de detección de deepfakes bajo distintos escenarios, manteniendo el balance entre videos reales y manipulados.

El objetivo del proyecto es diseñar, entrenar y evaluar una arquitectura de Deep Learning propia, teniendo como características ser ligera y eficiente, siendo capaz de clasificar entre 7 categorías distintas de Deepfake: video original y seis tipos de manipulación algorítmica.

2. ETL

ETL (Extracción, Transformación y Carga) es el proceso para dar inicio al análisis e interpretación de datos. A través de sus tres etapas, proporciona a los modelos de aprendizaje, datos íntegros, consistentes y uniformes, manteniendo los datasets centralizados. El resultado es un conjunto de datos de calidad, listo para alimentar el modelo de aprendizaje, con capacidad de incorporar nuevas fuentes de datos para enriquecerlo.

Extract

El dataset elegido fue: [FaceForensics++](#) el cual está disponible en *Kaggle (Plataforma donde)*. Se realizó un snapshot descarga completa y los archivos se almacenaron en una zona de *landing* sin modificaciones.

Carpetas	Descripción	Unidades
Deepfake Detection	Videos manipulados generados con método Deepfake detection baseline	1000 videos
Deepfakes	Videos creados por la técnica original de deepfake	1000 videos
Face 2 face	Videos manipulados mediante reenactment facial	1000 videos
Face shifter	Videos manipulados mediante reemplazo de rostro avanzado	1000 videos

Face Swap	Videos manipulados con técnica de intercambio de rostros	1000 videos
Neural Textures	Videos manipulados mediante texturas neuronales sintéticas	1000 videos
Original	Videos base sin manipular	1000 videos

Tabla 1. Clases del Dataset

Transform

Debido a lo complejo que es el procesar video completo, involucrando el manejar temporalidad y espacialidad, el deepfake moderno se identifica manifestándose fluidamente (cuadro a cuadro), tomé la decisión de trabajar con imágenes centradas en la región facial. Para ello trabajé en 3 fases:

1: Detección y Recorte de Rostros

El fondo de los videos conllevaba mucho ruido, al querer centrarme únicamente en las caras, usé el modulo DNN que es de OpenCV, esto con un modelo pre-entrenado llamado ResNet-10 SSD (Single Shot Detector), con este pude procesar cada frame seleccionado, detectar la región que si me interesaba (rostro), procesar cada frame seleccionada y aquellos que detectara una estructura facial hacerle un recorte, el tamaño estandarizado por imagen quedó en 224 x 224 píxeles.

2: Muestreo y paralelización

Si quisiera tener todos los cuadros de videos, tendria muchisimas imágenes las cuales me costaron mucho valor técnico (potencia de cómputo), tome la decisión de tomar 4 frames de un video, para acelerar el proceso,

usé procesamiento concurrente, no usando GPU, sino usando CPU, ahora era capaz de poder dividir varias imágenes mucho más rápido que de hacerlo de forma uno a uno.

3 Particionamiento y Distribución de las muestras

Para garantizar que los datos probados fueran correctos, hice la division de mis datos, por sus conjuntos tradicionales, Entrenamiento (70%), Validation (15%) y Prueba de (15%), esto por cada una de las categorías estudiadas divididas a nivel de video y no a nivel de imagenes. (Todos los frames extraídos de un mismo video pertenecen a un único subconjunto).

Quedando su proceso de la siguiente manera:

- Input: Videos .mp4 completos
- Proceso: Decodificación, Detección facial (SSD), Recorte, Resize.
- Output: Dataset de imágenes .jpg organizadas por clase y split (Train/Val/Test)

Clase	Train	Valid ation	Test	Total
Real	700	150	150	1000
DeepFak edetectio n	700	150	150	1000
Deepfak es	700	150	150	1000
Face2Fa ce	700	150	150	1000
FaceShif ter	700	150	150	1000
FaceSwa p	700	150	150	1000
NeuralT extures	700	150	150	1000

Total	4900	1050	1050	7000
-------	------	------	------	------

Tabla 2. Separación videos

3. Creación de modelo Deepfake (CNN Creación Manual)

Las redes Neuronales Convolucionales son un tipo de modelo que trabaja procesando datos con una estructura de rejilla, (imágenes representadas como matrices de píxeles), permitiendo el capturar patrones visuales gracias a múltiples capas que aprenden distintos tipos de características de forma progresiva de la imagen.

El primer modelo, fundamentado bajo los principios de una Red Neuronal Convolutiva, procesa cada imagen como una rejilla bidimensional y utiliza una secuencia de capas convolucionales para extraer representaciones distintas entre imágenes para poder ser capaz de distinguir entre una imagen real y un deepfake.

La arquitectura está conformada por 48 capas, tiene la particularidad de usar una estrategia de Convoluciones Separables en Profundidad (Depth Wise Separable Convolutions). Esta es la base de arquitecturas eficientes como lo es MobileNets.

A diferencia de una convolución estándar que filtra y combina los canales de entrada en un solo paso, una convolución separable en profundidad divide la operación en dos partes :

- Depthwise Convolution:

Encargado de aplicar un filtro por cada canal de entrada, siendo una etapa de filtrado únicamente espacial.

- Pointwise Convolution (1x1)

Combina los resultados de las salidas de la capa anterior, generando así nuevas características.

Esta arquitectura permite reducir el costo computacional (entre 8 y 9 veces menos que usando una convolución estándar de 3 x 3), siendo útil para disminuir el número de parámetros sin sacrificar la precisión en la detección de patrones complejos. como lo es el poder identificar deep fakes.

Preprocesamiento y Aumentación de Datos

En esta primera etapa, preparo las imágenes para evitar overfitting, haciendo una secuencia de transformaciones aleatorias a las imágenes, (rotación, zoom, contraste, brillo). Por último la capa Rescaling normaliza los valores de los píxeles de las imágenes pasando de 0-255 a un rango 0-1, facilitando la convergencia numérica en el entrenamiento.

Layer (type)	Output Shape	Param #
input (InputLayer)	(None, 128,128,3)	0
random_flip	(None, 128,128,3)	0
random_rotation	(None, 128,128,3)	0
random_zoom	(None, 128,128,3)	0
random_contrast	(None, 128, 128, 3)	0
random_brightness	(None, 128, 128, 3)	0
random_translation	(None, 128, 128, 3)	0

rescaling	(None, 128, 128, 3)	0
-----------	---------------------	---

Tabla 3. Preprocesamiento y Aumentación

Entrada Convolutiva (Feature entry)

El modelo utiliza una convolución estándar completa como punto de entrada, extrayendo las características más básicas (bordes y texturas), reduciendo la dimensión inicial de 128 x 128 a 64 x 64, mientras que el número de profundidad aumenta hasta los 48 filtros.

Layer (type)	Output Shape	Param #
Conv2D	(None, 64, 64, 48)	1296
batch_normalization	(None, 64, 64, 48)	192
activation (ReLU)	(None, 64, 64, 48)	0

Tabla 4. Feature Entry

Bloques de Extracción de Características

El modelo se compone de bloques repetitivos de convoluciones separables. Siguiendo los principios de diseño de arquitecturas eficientes, se implementa una estructura en la que las capas (depth wise y pointwise) tienen después una capa de batch normalization y su activación ReLU. Logrando tener estabilidad durante el entrenamiento.

La red mantiene la estructura piramidal, reduciendo progresivamente la resolución espacial mediante el uso de MaxPooling2D, mientras que aumenta la profundidad de los filtros para ser capaz de capturar las características semánticas de alto nivel. Después de cada bloque de reducción se integra un Dropout, dándole regularización al modelo.

Layer (type)	Output Shape	Param #
separable_conv 2D	(None, 64, 64, 64)	3504
batch_Normalization	(None, 64, 64, 64)	256
activation (ReLU)	(None, 64, 64, 64)	0
separable_conv 2D	(None, 64, 64, 64)	4672
batch_Normalization	(None, 64, 64, 64)	256
activation (ReLU)	(None, 64, 64, 64)	0
max_pooling2d	(None, 32, 32, 64)	0
dropout	(None, 32, 32, 64)	0
...

Tabla 5. Convoluciones Separables

Clasificador y Salida

Utilizo una capa de Global Average Pooling 2D, esta técnica condensa información promediando los mapas de características finales que ut

Esta técnica condensa la información espacial promediando los mapas de características finales, reduciendo la dimensión de salida a un vector de 256 valores. Esto tiene dos beneficios críticos: reduce drásticamente la cantidad de parámetros (evitando el crecimiento exponencial de pesos en la capa densa) y minimiza el riesgo de sobreajuste (overfitting).

Finalmente, el vector de características pasa por una capa densa de 128 neuronas antes de llegar a la capa de salida, la cual utiliza una función de activación Softmax para entregar

la probabilidad de pertenencia a cada una de las 7 clases (Original + 6 métodos de manipulación).

Resultados

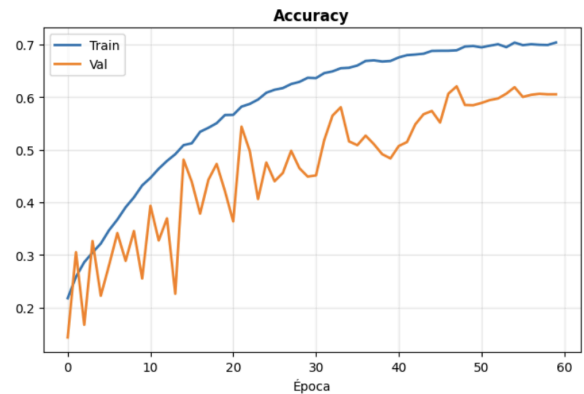


Imagen 1. Matriz de confusión

La curva de pérdida de entrenamiento (*Train Loss*) muestra un descenso constante y suave, indicando que el modelo está aprendiendo efectivamente a minimizar el error. Sin embargo, la pérdida de validación (*Val Loss*) presenta un comportamiento volátil con picos oscilatorios. Esto sugiere que, debido al reducido número de parámetros (175k), el modelo es sensible a ciertos lotes (*batches*) de validación que contienen características difíciles de generalizar, luchando por encontrar un mínimo global estable.

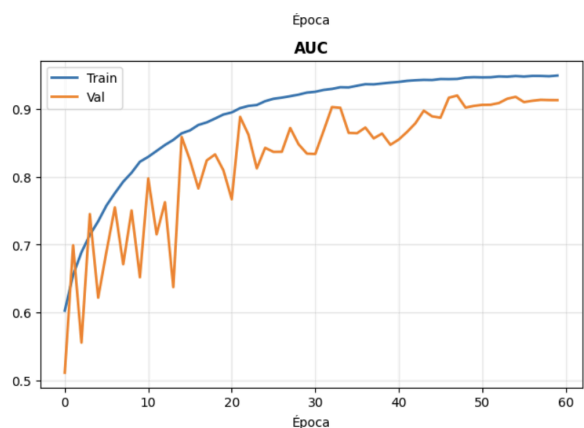


Imagen 2. Gráfica de AUC

Esta gráfica me ayuda a evaluar la capacidad discriminativa de mi CNN en el cálculo de probabilidad de la clase correcta contra incorrectas. Aunque el modelo si se

equivoca, el modelo ha aprendido a identificar patrones de Deepfake frente a las imágenes reales base.

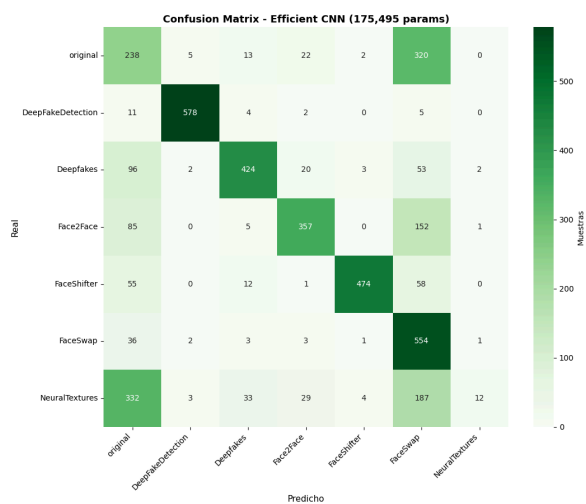


Imagen 3. Matriz de confusión

La matriz de confusion, creada con resultados de frames provenientes de videos se observa una fuerte capacidad de discriminación en ciertas categorías específicas, tomando en cuenta como relevante el buscar una alta densidad de resultados en forma diagonal.

DeepFakeDetection logra el mejor rendimiento, con 578 predicciones correctas, además del menor nivel de fallos. La razón de esto es que este método de Deepfake deja artefactos visuales muy evidentes que el modelo puede capturar.

Deepfakes, FaceSwap, FaceShifter y Face2Face muestran un buen rendimiento aunque no siendo el mejor como lo muestra DeepFakeDetection, indicando que el modelo ha aprendido características robustas para identificar este tipo de intercambio de rostros.

El modelo presenta dos clases las cuales no fue capaz de asignar correctamente. NeuralTextures en donde 332 fueron clasificadas erróneamente como originales, mientras que sólo 187 fueron detectadas

correctamente y FaceSwap en donde 320 fueron clasificadas incorrectamente como originales, al hacer el análisis directo en estas dos clases, se encontró que este tipo de Deepfake contiene una mejor calidad de imagen y pertenecen a ser una técnica más reciente.

4. Mejora del modelo Creación de modelo Deepfake (CNN Transfer Learning)

Este segundo modelo surge de la necesidad de superar las limitaciones provenientes del primer modelo, en donde la arquitectura, no es capaz de detectar manipulaciones sutiles como lo hace NeuralTextures.

En lugar de incrementar mas y mas la profundidad de la red, (mayor tiempo en entrenamiento y un riesgo de generar overfitting), decidí aprovechar una arquitectura previamente entrenada, el cual en este caso elegí EfficientNetB0, para comparar con sus 4 Millones de parámetros la diferencia, y ver el balance entre precisión y eficiencia computacional de mi primer modelo.

Arquitectura del modelo

Layer	output	Param #
input layer	(None, 224,224,3)	0
rescaling	(None, 224, 224, 3)	0
random_flip	(None, 224, 224, 3)	0
random_rota tion	(None, 224, 224, 3)	0
random_zoo m	(None, 224, 224, 3)	0

random_jpeg	(None, 224, 224, 3)	0
Rescaling	(None, 224, 224, 3)	0
efficientnetb0	(None, 7, 7, 1280)	4,049,571
GlobalAveragePooling2D	(None, 1280)	0
Dropout	(None, 1280)	0
Dense	(None, 7)	8,967

Tabla 6. Arquitectura efficientnetb0

EfficientNetB0, requiere para operar imágenes (224 x 224), por lo tanto se tuvo que hacer un aumento de la resolución de entrada. Esto lo usa la red para aprovechar los filtros que ya tiene pre-aprendidos detectando texturas y artefactos minúsculos que se pierden en el anterior modelo.

Primero, una capa de Rescaling normaliza en rango 0-1, posteriormente las capas de aumentación de datos (RandomFlip, RandomZoom y RandomJPEG) operan de forma adecuada.

Estrategia del Entrenamiento en Dos Fases

Para utilizar el modelo Pre-entrenado de forma correcta evitando destruir lo ya aprendido del modelo EfficientNetB0, se diseñó una estrategia de entrenamiento secuencial, en donde:

- Se congelan todos los pesos de EfficientnetB0, donde solo se entrenen las nuevas capas del clasificador, esto para poder tener los pesos de la capa de salida sin

distorsiones excesivas generadas por la red base.

- Fine Tuning después de haber hecho el calentamiento, aplicado a la mitad superior únicamente, siendo el 50% de las capas finales, haciendo que únicamente el filtro de alto nivel pueda identificar los patrones del deepfake.

Clasificador y Salida

Se eliminó la capa superior original de ImageNet, sustituyendo por una nueva estructura personalizada. Global Average Pooling 2D, los uso para condensar los mapas de características profundos. Dropout para prevenir el overfitting durante el entrenamiento.

Softmax de nuevo termina siendo el que genera las probabilidades de las 7 clases.

Resultados

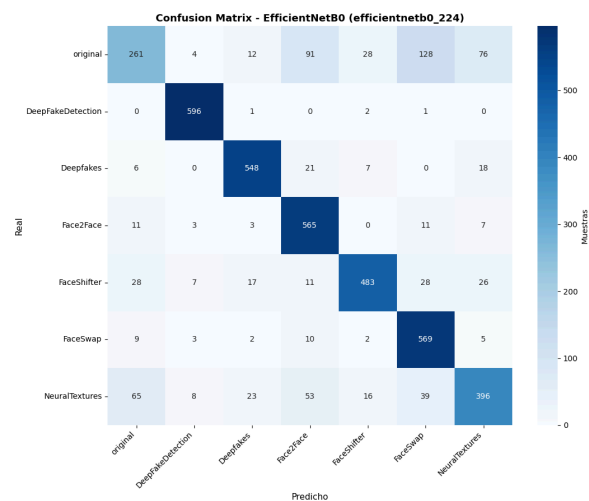


Imagen 4. Matriz de confusión

Este segundo modelo, evidencia una diagonal principal sólida en las clases.

DeepFakeDetection y FaceSwap mantienen su rendimiento, con 596 y 569 predicciones correctas respectivamente, se observa que la mejora más significativa está en la clase NeuralTextures, punto crítico del primer modelo, ahora con un resultado positivo,

duplicando la eficacia, pasando de 187 a 396 resultados verdaderos.

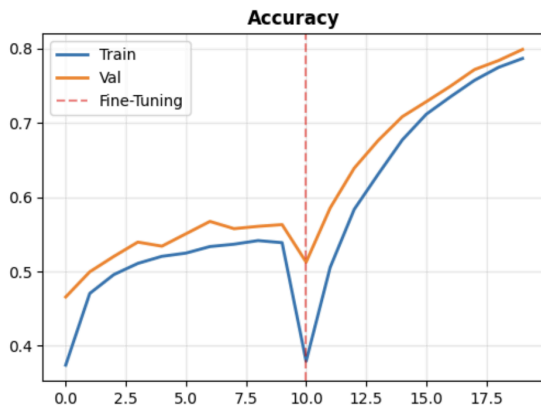


Imagen 5. Gráfico de Accuracy

La gráfica de Accuracy valida exitosamente la estrategia de entrenamiento en dos fases, durante la fase de feature extraction, se ve un aprendizaje moderado en estas capas congeladas.

La línea roja nos indica el inicio del fine-tuning, en donde el crecimiento es paralelo en la segunda etapa del entrenamiento llegando el modelo a un 78%.

5. Resultado final

Con base en la arquitectura optimizada, decidí implementarlo en un pipeline, el cual es capaz de procesar archivos de videos completos, transformando la entrada cruda en diagnósticos visuales y estadísticos.

Siguiendo lo ya antes desarrollado, se trabaja primero en la lectura de fotogramas de forma secuencial, detectando la cara con el detector DNN de OpenCV, recorta la cara a (128x128), la preprocesa para entonces pasar por el modelo ya antes entrenado para identificar sus 7 clases. Este modelo ya se encontraba previamente guardado.

Aunque la red se clasifica entre 7 categorías específicas se genera una nueva métrica binaria de autenticidad que facilita el identificar fake o verdadero.

Es así que hago este cálculo de probabilidad en donde $P_{fake} = 1 - P_{original}$.

Si P_{fake} supera el umbral de 0.5 entonces el fotograma se etiqueta como manipulado, además de nombrar el tipo específico de Deepfake(Faceswap como ejemplo).

Salida

El pipeline genera dos productos principales, video anotado y reporte CSV con análisis cuadro a cuadro. Para generar el video, se arma de nuevo con **cv2.VideoWriter** para reconstruirlo frame a frame.

En el video se superpone en cada frame un recuadro negro sobre el rostro y metricas de probabilidad calculadas, permitiendo en tiempo real observar el comportamiento del modelo.

Aquí se carga el modelo CNN y con ayuda del writer, los frames anotados se van ya ejecutándose y escribiendo de forma secuencial, hasta juntar el video final.

Archivos finales:

- Video, con recuadro y datos superpuestos.
- Reporte csv, (archivo con datos estructurados), donde se tiene el análisis detallado cuadro por cuadro, útil si se quiere analizar un video complejo.

Ejemplo Video prediccion_7clases

[Video](#)

[CSV](#)



Imagen 6. Video Resultante

6. Conclusión

Sobre como una CNN puede alimentar y obtener información acerca de imágenes me forme profesionalmente desarrollando el modelo creado desde 0, este me demuestra que es posible lograr una alta precisión en detección de DeepFake manteniendo siempre un costo computacional muy bajo, 175k parámetros.

No obstante, también lidie con los problemas ocasionados por la baja resolución de la imagen (128 x 128), sumado a la compresión del modelo que finalmente limitan su capacidad para detectar Deepfake, con manipulaciones sutiles como NeuralTextures, generando una alta tasa de falsos negativos.

Aprender a trabajar con transfer learning enriqueció mi conocimiento sobre Machine Learning, encontré una forma ideal para capturar las características de los videos, sin tener que modificar mi dataset original encontrando de forma exitosa, las características distintivas de videos, me hubiera gustado, tener un mejor accuracy, sin embargo, sí implicaría el trabajar con videos para poder analizar movimientos graves en cámara.

7. Bibliografía

Repositorio: [Bowist27/Deep-Learning: Arquitectura de deep learning para solucionar problema comun de kaggle \(Imágenes\)](#)

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient convolutional neural networks for mobile vision applications* (arXiv:1704.04861). arXiv. <https://arxiv.org/abs/1704.04861>

Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2019). *FaceForensics++: Learning to detect manipulated facial images*. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 1–11. <https://doi.org/10.1109/ICCV.2019.00009>

Tobing, F. A. T., Kusnadi, A., Zuhdi Pane, I., & Winantyo, R. (2025). Deepfake detection using convolutional neural networks: A deep learning approach for digital security. *Indonesian Journal of Electrical Engineering and Computer Science*, 39(2), 1092–1099. <https://doi.org/10.11591/ijeecs.v39.i2.pp1092-1099>

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient convolutional neural networks for mobile vision applications*. arXiv. <https://arxiv.org/abs/1704.04861>

Vivekananda, G. N., & Kumar, P. (2025). Refining digital security with EfficientNetV2-B2 deepfake detection architecture. *Journal of King Saud University – Computer and Information Sciences*, 37(6), 101874. <https://doi.org/10.1016/j.jksuci.2025.101874>

\Maheswari, S., & Kumar, C. S. (2025). Real-time deepfake detection using a hybrid MobileNet-LSTM architecture. *Atlantis Press*, 126017010. <https://doi.org/10.1109/ICACCS61007.2025.10609723>

Marko, K. (2024). Cross-dataset deepfake detection: Evaluating the generalization of CNN-based detectors. *Proceedings of the Computer Vision Winter Workshop*, 24. <https://doi.org/10.3217/3fn2-2g34>