ISSN 1870-4069

Entrenamiento de un clasificador de videos DeepFake en un equipo de cómputo con recursos limitados

Odón D. Carrasco-Limón, Maya Carrillo-Ruiz,
María de Lourdes Sandoval-Solis

Benemérita Universidad Autónoma de Puebla,
Facultad de Ciencias de la Computación,
México

{odond.carrasco, maya.carrilloruiz,
mariad.sandovalsolis}@viep.com.mx

Resumen. En la actualidad la utilización de aplicaciones móviles para generar videos DeepFake al alcance de cualquiera, suscita la problemática de no poseer herramientas que nos permitan discernir si un video es o no un DeepFake, por lo que surge la necesidad de poner al alcance de usuarios técnicas para su detección. En este documento se presenta un método, para la detección de videos modificados por la técnica DeepFake. Dicho método utiliza redes neuronales que pueden ser entrenadas en ordenadores personales, por medio de una técnica que denominamos aprendizaje particionado. Los resultados obtenidos muestran que la exactitud obtenida con el aprendizaje particionado es aceptable además de reducirse considerablemente el tiempo de entrenamiento.

Palabras clave: Visión por computadora, clasificadores, CNN, aprendizaje particionado, deepfake.

Training a DeepFake Video Classifier on a Computer with Limited Resources

Abstract. Nowadays, the use of mobile applications to generate DeepFake videos accessible to anyone raises the issue of not having tools that allow us to discern whether a video is a DeepFake or not. Therefore, there is a need to provide users with techniques for its detection. This document presents a method for detecting videos modified by the DeepFake technique. This method utilizes neural networks that can be trained on personal computers using a technique called partitioned learning. The results obtained show that the accuracy achieved with partitioned learning is acceptable, and the training time is significantly reduced.

Keywords: Computer vision, classifiers, CNN, partitioned learning, deepfake.

1. Introducción

Actualmente existen aplicaciones para dispositivos móviles que generan videos DeepFake [1], técnica de intercambio de identidad, de manera completamente automática. Sin embargo, no existen herramientas que nos permitan detectar dicha manipulación.

Aunque se encuentra diversas aproximaciones como las generadas en el concurso de detección de DeepFake (DFDC) [2], estas soluciones requieren de poder de cómputo considerable para el entrenamiento de redes neuronales convolucionales (CNN).

Dado que no siempre se cuenta con tal poder de cómputo, el presente trabajo propone un método de detección de videos falsos que puede ser entrenado en computadoras personales para evitar posibles estafas y manipulación de la información.

Desde el 2017 que apareció este término, se encontraron más de 14,698 videos manipulados, como menciona la cadena de noticias BBC en su reportaje del 2019[3]. Además, del total de videos DeepFake encontrados en la red en ese momento, el 96% eran de naturaleza pornográfica. Ante estos eventos, métodos como el propuesto, día a día serán de mayor utilidad.

En la sección 2, se presenta tres aproximaciones del estado del arte, se describen brevemente sus métodos y a partir de ellas se esquematiza un modelo general. En la sección 3 se describen las etapas del método propuesto, así como las herramientas utilizadas. En la sección 4, se describe el conjunto de datos empleado, el procesamiento realizado y su organización, para emplearlo en aprendizaje automático. En la sección 5 se analizan los resultados de los experimentos y finalmente en la sección 6 se presentan las conclusiones y trabajo futuro.

2. Trabajos relacionados

Tolosana et al. en [1] describen 4 técnicas de manipulación facial en videos, las cuales son: a) Síntesis facial completa, la cual crea rostros inexistentes realistas por medio de poderosas redes adversarias generativas (GAN). b) Intercambio de identidad, la cual cambia el rostro de una persona por otro, mediante técnicas de aprendizaje profundo también conocido por DeepFake. c) Manipulación de atributos, que cambia características como: color de pelo, color de ojos, edad, añade gafas, etc., por medio de redes GAN. d) Intercambio de expresiones, esta manipulación cambia las expresiones de una persona por la de otra, esta es la técnica más reciente con técnicas de Neural-Textures, mediante poderosas redes GAN. De estas técnicas la más accesible al público es la de DeepFake existiendo aplicaciones móviles que permiten crear videos modificados de manera automática, por esto se buscará un detector para esta técnica.

A continuación, se describen brevemente los trabajos más relevantes, en los que podrá observarse que existen etapas comunes en las soluciones propuestas, las cuales son: una etapa de extracción de rostros en la cual se separa el rostro para determinar si un video es real o no, una etapa de extracción de características en la cual típicamente se utiliza alguna red CNN para que las determine y una etapa de clasificación donde a partir de los resultados de la extracción de características se determina si el video es real o no, dichas etapas se pueden observar en la Figura 1.

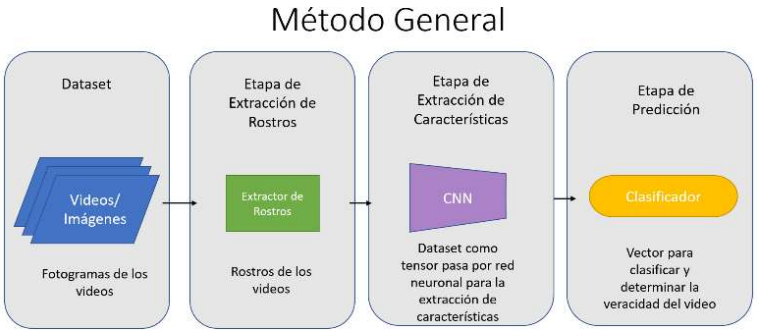

Fig. 1. Etapas del modelo general para detección de DeepFake. Creación propia.

Tolosana et al. en [4], parten de otros estudios que consideran características de bajo y alto nivel en el rostro del individuo, establecen que la región de los ojos, la boca y la pose de la cabeza, dan muchas características para discernir si el video es falso o no, así decide extraer estas 4 regiones y en la última coloca el resto del rostro.

Toma todos los fotogramas del video, y obtiene los rostros mediante el toolkit OpenFace2 [5], continúa con una etapa de extracción de características utilizando dos Redes CNN, una XceptionNet preentrenada con pesos de ImageNet [4] y una red cápsula VGG19. A continuación mediante reglas de inferencia obtiene el área bajo la curva (AUC) de la gráfica Receiver Operating Characteristic (ROC), de 0.994 para la base de datos FF ++, 0.91 para la base de datos DFDC y 0.836 para la base de datos de Celef-db.

El equipo ganador del concurso DFDC, fue Seferbekov, con una propuesta que aparece en GitHub como Selimsef [6] del año 2020, la cual consta de una etapa de extracción de todos los fotogramas del video original, extracción de rostros por medio de una red CNN, aplicación de aumentos a estos rostros, como ruido gaussiano, transformaciones mediante rotación, escalamiento, escala de grises etc.

Para la etapa de extracción de características, el autor propone EfficientNet [7], que mejora al preentrenarla con Noisy Student (una aproximación de aprendizaje semi-supervisados), finalmente realiza la clasificación por medio de reglas de inferencia, logrando un desempeño con Log Loss[1] de 0.42798.

El autor ganador del segundo lugar en DFDC, H. Zhao y su equipo WM [8], en la etapa de extracción de rostros utilizaron una red CNN RetinaFace, aplican aumentos a los rostros extraídos, posteriormente en la etapa de extracción de características utilizan una XceptionNet preentrenada, EfficientNet y XceptionNet con una red de aumento de datos (WSDAN). Crean una interpolación bilineal con estas tres redes, obteniendo un desempeño con Log Loss de 0.42842.

Li et al. en [11] utilizan Dlib como extractor de rostros, en la etapa de extracción de características utilizan una Red Resnet101, finalmente en la clasificación utilizan reglas de inferencia midiendo el desempeño en AUC en el conjunto UADFV obtienen 97.4.

Si bien los trabajos que se presentan no muestran el tiempo que tomó el entrenamiento de cada modelo, cada uno intenta tener la mejor exactitud ocupando los

[1] Métrica específica de desempeño, la fórmula de Log Loss es -1 * el logaritmo de la función de probabilidad, para cualquier problema dado, un valor de pérdida logarítmica más bajo significa mejores predicciones

Tabla 1. Tabla comparativa del estado del arte.

Articulo	Extractor de Rostros	Extractor de Características	DB	Medida de Desempeño
Tolosana- Facial Regions Features (2020)[4]	OpenFace2+ Alineamiento	Xception P. y Capsule Network (VGG19)	FF ++ | DFDC | Celb-DF	0.994 | 0.91 |0.836
S. Seferbekov - Selimsef-(2020)[5]	MTCNN + Aumentos	EfficientNet B7 P. con Noisy Student	DFDC	0.42798 LogLoss
H. Zhao -WM- (2020)[7]	RetinaFace + Alineamiento + Aumentos	Xception P. y EfficientNet B3 + WSDAN y Xception P. + WSDAN	DFDC	0.42842 LogLoss
Li-Face Warping Features (2019)[11]	Dlib	ResNet101	UADFV	0.974

mejores algoritmos y herramientas disponibles en su momento, mismas que implícitamente buscan una mejora en tiempo.

3. Arquitectura del sistema

Para el análisis a profundidad de las etapas mencionadas en la sección anterior, se compararon los métodos utilizados por los autores como se muestra en la Tabla 1. Así se identificaron el extractor de rostros, el extractor de características y el conjunto de datos adecuado para el entrenamiento.Para la selección se eligieron las herramientas con mayor exactitud. A continuación, se describe brevemente en qué consisten las herramientas seleccionadas de la Tabla 1.

Openface2, es un extractor de rostros seleccionado para este trabajo, se compone de una serie de herramientas las cuales implementan los mejores algoritmos para la estimación de Landmarks, utilizando una CE-CLM (CNN de expertos en cascada), incluso en imágenes con rostros de perfil y zonas no visibles.

Openface2 tiene algunos errores en videos, por lo que es recomendable utilizar el extractor dos veces ocupando la salida del primero como la entrada al segundo, para reducir los errores. A pesar de la existencia de estos fallos, Openface2 logra ser el mejor extractor de rostros según [4] en el cual se comparan diversos extractores de rostros.

3.1 Extracción de características

En el campo de visión por computadora se han creado diversas arquitecturas de CNN y se han puesto a prueba en el conjunto de datos ImageNet con el fin de detectar objetos en imágenes. Tales CNN han sido ResNet, InceptionNet y XceptionNet, por dar algunos ejemplos.

Todas estas redes buscan tener mayor precisión en la detección de objetos, trabajando con redes robustas incrementando continuamente su tamaño. Al escalar la red en profundidad, ancho (tamaño de vector de salida) o tamaño de canal (tamaño de vector de entrada); se obtiene mayor precisión. Sin embargo, el tiempo de entrenamiento se incrementa, así como el número de parámetros y el coste

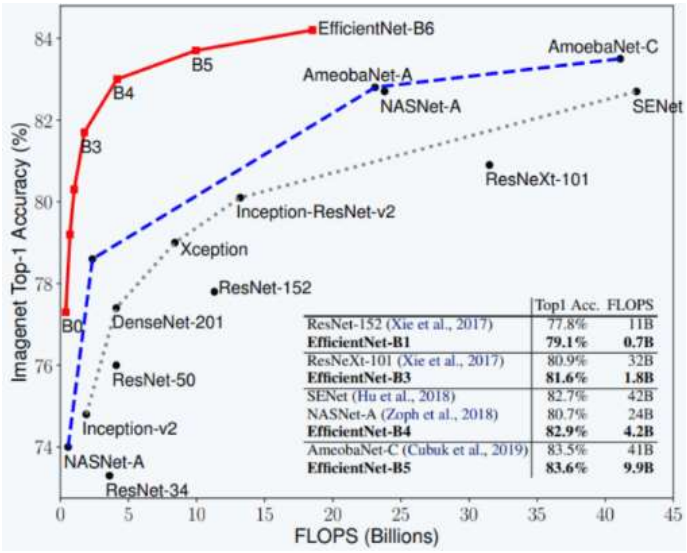

Fig. 2. Comparación de la Familia de CNN EfficientNet comparada con otras CNN en número de operaciones de punto flotante, extraída de [7].

computacional. La mayoría de las CNN buscan escalar un solo valor de los mencionados, pero nunca una combinación de estos [7].

Por otro lado, EfficientNet surge como una red que intenta encontrar un factor de escalamiento para los parámetros antes mencionados, buscando maximizar la precisión, minimizando el número de parámetros y el coste computacional. Esta red se modeló en forma de función, y de manera empírica se encontraron valores para escalar la red, logrando una exactitud superior a las CNN mencionadas previamente.

Con estos factores de escala, los autores propusieron un caso base y aplicando estos factores en forma recursiva crearon una familia de CNN llamada EfficientNet. B0 es el caso base, B1 el siguiente factor, y así sucesivamente hasta el B7, teniendo similar o mejor exactitud que otras CNN con un número menor de parámetros y número de operaciones de punto flotante como se puede ver en la Figura 2.

En el concurso DFDC los ganadores utilizan EfficientNet como extractor de características demostrando que esta familia de arquitecturas no solo logra discernir objetos en imágenes, si no también logra extraer características en rostros humanos.

Esta disminución de parámetros y la reducción significativa de operación de punto flotante hacen que EfficientNet, como extractor de características, pueda utilizarse en cualquier equipo. Aunque la disminución de poder de cómputo requerido no es suficiente para utilizar los modelos más grandes en equipos personales.

3.2 Problemas por tamaño de información en extracción de características

Según lo observado en la Figura 2 utilizando una arquitectura de familia más grande se obtendrá más exactitud. El problema es que el tamaño necesario para procesar la información crece, por ejemplo, en B0 la entrada requiere que las imágenes tengan un tamaño de 224 pixeles de largo y ancho. Como la imagen tiene 3 canales (RGB), el

Tabla 2. Crecimiento de potencia para EfficientNet.

Base modelo	Resolución	Potencia necesaria para 34,778 imágenes
EF B0	224	4.88GB +modelo
EF B1	240	5.60 GB +modelo
EF B2	260	6.57 GB +modelo
EF B3	300	8.75 GB +modelo
EF B4	380	14.03 GB +modelo
EF B5	456	20.20 GB +modelo
EF B6	528	27.09 GB +modelo
EF B7	600	34.98 GB +modelo

tamaño del tensor para entrenar la red tendría un tamaño de 224* 224 * 3 * número de imágenes en la base de datos ocupadas para entrenar.

Así el poder de cómputo necesario para poder manejar el tensor crece rápidamente al tener una base de datos con más imágenes y al cambiar la arquitectura como se puede observar en la Tabla 2, lo que vuelve inviable el manejo del tensor entero para computadoras personales.

Suponiendo que la base de datos tenga una cantidad de 34,778 imágenes, como es la base de UADFV tras extraer los rostros, entonces se necesitará la potencia de cómputo que se observa en la Tabla 2.

Dado que pocas computadoras personales cuentan con tal potencia de cómputo para trabajar con el tensor entero, existen alternativas para poder trabajar sin contar con dicha potencia. Una alternativa es incrementar la potencia del equipo con el cual se entrena la red, pero muchas veces esto no es posible. Por lo que se propone el Aprendizaje particionado.

3.3 Aprendizaje particionado

En el presente trabajo como alternativa para operar con tensores (conjuntos de datos) que superen la potencia de cómputo disponible se propone ocupar lo que llamamos Aprendizaje particionado (AP). El AP se realiza dividiendo un tensor para la resolución del problema, entrenando la red con segmentos del tensor y ocupando los modelos resultantes con otros segmentos distintos del mismo, para reducir la potencia de cómputo necesaria y así disminuir el tiempo de entrenamiento. En la Tabla 3 se mencionan el algoritmo de AP.

4. Experimentos

El objetivo será medir la exactitud que alcanza una red con la técnica AP comparada con una red que no utiliza AP, dado que el conjunto de datos esta balanceado.

Las características del equipo con el que se realizaron las pruebas son las siguientes, la computadora tiene un procesador Ryzen 5 2600, cuenta con 32 Gb de memoria RAM, tiene una tarjeta de video dedicada Nvidea RTX 3070 con 8Gb de VRAM y posee una unidad de almacenamiento de 240Gb de espacio, el lenguaje utilizado para programar fue Python.

Tabla 3. Algoritmo de AP.

1	Partir el tensor T en dos subconjuntos A y B, asignándoles el D% e ID% de datos respectivamente.
2	Crear un modelo M entrenado en 15 épocas, con el D% de datos del subconjunto A.
3	Utilizar el ID% de datos del subconjunto B para entrenar el modelo M, en 5 épocas. Y producir los Modelos D% + ID%

Tabla 4. Nombre de los modelos y cantidad de imágenes.

Nombre Modelo	Imágenes
EF0 10%	3477
EF0 20%	6956
EF0 50%	17389
EF0 100%	34778

Aunque las características del equipo ocupado parecieran alejadas a lo que se pudiera entender por computadoras personales, los requerimientos de hardware que mencionan los equipos de Seferbekov[6] y Zhao H. et. al [8] hacen inviable el entrenamiento de sus modelos con una sola tarjeta de video dedicada, misma razón por la cual surge la idea de ocupar el AP. A continuación, se describe el dataset utilizado para las pruebas.

4.1 Dataset

Para la técnica de intercambio de identidad existen dos generaciones de conjuntos prueba: la primera generación, con pocos videos de baja resolución y con personas hablando a la cámara directamente; la segunda generación, con videos de resoluciones más altas, diversos contextos de iluminación, profundidad y poses de personas.

Con la finalidad de iniciar la experimentación se seleccionó base de datos UADFV, que es comparable con la base de datos FF++ por ser de primera generación, y obtener un AUC de 99.4% en ambas.

UADFV es una colección de 98 videos etiquetados, 49 reales y 49 modificados por la técnica de intercambio de identidad DeepFake, creada por Y. Li et al. [10], cada video tiene una duración de entre 6-16 segundos, con tamaños variados entre 300x200 pixeles a 600x400 pixeles, con poca variedad de movimiento y poca variedad de luminosidad, con rostros lo más alineados a la cámara, considerada como base de datos de primera generación.

4.2 Procesamiento del Dataset

Se utilizó OpenFace2 para extraer los rostros y se garantizó que todas las imágenes fueran rostros humanos, y posteriormente se creó un tensor el cual se forma con imágenes colocadas en arreglos con valores de entre 0 y 255, y con dimensiones como se muestra en la Tabla 2. Se colocó las etiquetas de dichas imágenes en un arreglo de vectores donde [1,0] identificara rostros falsos, y [0,1] rostros reales o verdaderos.

Tabla 5. Exactitud de los modelos 10% + 10 contra modelos sin AP 20%.

	Modelo 20 %			**Modelo 10% + 10**			**Diferencia**		
	Entrenamiento	Test	Tiempo	Entrenamiento	Test	Tiempo	Entrenamiento	Test	Tiempo
EF0	96.85%	96.77%	7.74'	**97.48%**	**97.31%**	**6.2**'	0.65%	0.55%	19.90%
EF1	**96.60%**	**96.51%**	11.96'	96.26%	96.09%	**9.53**'	-0.35%	-0.44%	20.32%
EF2	96.80%	96.72%	15.39'	**98.13%**	**97.96%**	**11.95**'	1.36%	1.27%	22.35%
EF3	97.31%	97.22%	37.07'	**97.82%**	**97.65%**	**20.84**'	0.52%	0.44%	43.78%

Tabla 6. Exactitud de los modelos AP 20% + 30 contra modelos sin AP 50%.

	Modelo 50 %			**Modelo 20% + 30**			**Diferencia**		
	Entrenamiento	Test	Tiempo	Entrenamiento	Test	Tiempo	Entrenamiento	Test	Tiempo
EF0	**97.33%**	**97.62%**	23.5	96.98%	96.93%	**12.49**	-0.36%	-0.71%	46.85%
EF1	96.99%	**97.67%**	36.82	**97.32%**	97.24%	**19.99**	0.34%	-0.44%	45.71%
EF2	**97.70%**	**97.67%**	47.4	97.56%	97.51%	**24.29**	-0.14%	-0.16%	48.76%
EF3	**97.83%**	**97.80%**	78.95	97.14%	97.10%	**52.94**	-0.71%	-0.72%	32.94%

Tabla 7. Exactitud de los modelos AP 50% + 50 contra modelos sin AP 100%.

	Modelo 100 %			**Modelo 50% + 50**			**Diferencia**		
	Entrenamiento	Test	Tiempo	Entrenamiento	Test	Tiempo	Entrenamiento	Test	Tiempo
EF0	**98.30%**	**98.16%**	50.3	98.16%	98.15%	**41.41**	-0.14%	-0.01%	17.67 %
EF1	**98.38%**	**98.36%**	125.06	97.37%	97.35%	**75.79**	-1.04%	-1.04%	39.40 %
EF2	**95.32%**	**95.16%**	320.91	93.07%	93.06%	**158.47**	-2.42%	-2.26%	50.62 %

Para el entrenamiento de cada modelo, se seleccionó una muestra de imágenes hasta alcanzar una cantidad de imágenes como se puede observar en la Tabla 4 donde el porcentaje corresponde a la totalidad de imágenes de la base de datos UADFV. De esta cantidad de imágenes la mitad son falsas y la mitad son verdaderas para tener un subconjunto balanceado.

Se utilizó clasificación a 5 pliegues utilizando 80% de los datos para entrenamiento y 20% para pruebas.

Para crear los modelos base, para la comparación con AP, se utilizó EfficientNet, desde B0 hasta B3 que se nombraron como EF seguido del número de familia. Cada modelo se creó con pesos aleatorios, utilizando el optimizador Adam, con un total de 15 épocas y con un batch_size[2] de 16.

Dichos valores se seleccionaron por dar la mejor relación exactitud-tiempo en experimentos realizados previamente. Los modelos se definieron con el 10%, 20%, 50% y 100% de los datos.

Para los modelos AP se seleccionaron los tensores correspondientes al 20%, 50% y 100% de datos. Los tensores se particionaron en subtensores A y B con el 10% , 10%; 20%, 30% y 50%, 50% de los datos respectivamente. Para compararlos con los modelos base del 20%, 50% y 100%, paso 1 de la Tabla 3.

Posteriormente se definieron los modelos M con los modelos base de 10%, 20% y 50%, dado que ya estaban entrenados, paso 2 de la Tabla 3.

A continuación, a los modelos M se les agregaron el 10%, 30% y 50% para alcanzar el 20%, 50% y 100% de datos respectivamente de los modelos base. En este proceso se garantizó que los datos agregados fueran los correspondientes a los modelos base asociados.

[2] El número de ejemplos en un batch o lote, es decir conjunto de ejemplos usados en una iteración del entrenamiento del modelo.

Por ejemplo, para el modelo base creado con el 10% de los datos, se le agrego el subtensor B correspondiente para tener los mismos datos que el modelo base del 20% de datos. Al modelo generado se le nombro "Modelo 10% +10". Y de manera semejante para el resto de los modelos, paso 3 de la Tabla 3.

Los resultados de los experimentos realizados se muestran en las Tabla 5, 6 y 7 así como el tiempo de entrenamiento donde el mejor resultado esta resaltado en negritas. El tiempo se muestra con el objetivo de visualizar que este se reduce, lo que hace viable realizar el entrenamiento en equipos personales.

5. Discusión de resultados

De los resultados obtenidos puede rescatarse lo siguiente.

El modelo con la mayor diferencia es EF2 Modelo 50% + 50 con EF2 Modelo 100% el cual difiere en exactitud con un -2.42% en entrenamiento y -2.26 en test. Sin embargo, la diferencia promedio en exactitud entre los modelos sin AP a los modelos con AP es de -0.21% en entrenamiento y -0.32% en test.

Por otro lado, los modelos AP, logran superar siempre en tiempo a los modelos sin AP, logrando una mejora significativa del 25.21% en promedio. Siendo la mejora más pequeña en tiempo de 17.67 % en el modelo EF0 Modelo 50% + 50 respecto a EF0 Modelo 100% y la mejora más grande en tiempo de 64.06% en el modelo EF2 Modelo 50% + 50 respecto de EF2 Modelo 100 %.

6. Conclusiones y trabajo futuro

Con los resultados observados en las Tablas 5, 6 y 7 respecto a la ganancia en tiempo de entrenamiento comparado con la perdida de exactitud, respecto de los resultados de los modelos base, donde la máxima exactitud es 98.38% contra 97.37% del modelo AP, se puede concluir que el AP es una alternativa para entrenar modelos en computadoras personales.

El AP pueden ser utilizado para entrenar modelos en equipos con menor potencia de cómputo, con respecto a los utilizados por los equipos de Seferbekov [6] y Zhao H. et al. [8], dado que al trabajar con subtensores se requiere menor espacio en memoria para el entrenamiento.

Para trabajos futuros se buscarán maneras de alcanzar un equilibrio entre la exactitud, el tiempo de entrenamiento y la potencia de cómputo requerida. De la misma manera se planea expandir los experimentos a la base de datos de segunda generación y utilizar la métrica AUC en conjuntos de datos no balanceados.

Referencias

1. Tolosana, R., Vera-Rodriguez, R., Fierrez, J., Morales, A., Ortega-Garcia, J.: Deepfakes and beyond: A survey of face manipulation and fake detection. Information Fusion, vol. 64, pp. 131-148 (2020) doi: 10.1016/j.inffus.2020.06.014
2. Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M., Canton-Ferrer, C.: The deepfake detection challenge (DFDC) dataset. (2020) doi: 10.48550/arXiv.2006.07397

3. Cellan-Jones, R.: Deepfake videos double in nine months. BBC News (2019) https://www.bbc.com/news/technology-49961089
4. Tolosana, R., Romero-Tapiado, S., Fierrez, J., Vera-Rodriguez, R.: Deepfakes evolution: Analysis of facial regions and fake detection performance. Pattern Recognition, In: ICPR International Workshops and Challenges (ICPR 2021) Lecture Notes in Computer Science, vol. 12665, pp. 442–456 (2021) doi: 10.1007/978-3-030-68821-9_38
5. Baltrusaitis, T., Zadeh, A., Lim, Y., Morency, L. P: OpenFace 2.0: Facial behavior analysis toolkit. In: 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), pp. 59–66 (2018) doi: 10.1109/FG.2018.00019
6. Seferbekov, S.: Deepfake detection (DFDC) solution. (2021) https://github.com/selimsef/dfdc_deepfake_challenge
7. Tan, M., Le, Q.: EfficientNet: rethinking model scaling for convolutional neural networks. In: Proceedings of the 36th International Conference on Machine Learning (PMLR), vol. 97, pp 6105–6114 (2019) doi: 10.48550/arXiv.1905.11946
8. Zhao, H., Cui, H., Zhou, W.: 2nd place solution for Kaggle deepfake detection challenge. (2021) https://github.com/cuihaoleo/kaggle-dfdc accesado el 20/08/2021
9. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001 (2001) doi: 10.1109/CVPR.2001.990517
10. Li, Y., Yang, X., Sun, P., Qi, H., Lyu, S.: Celeb-DF: A large-scale challenging dataset for deepfake forensics. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3207–3216 (2020) doi: 10.1109/CVPR42600.2020.00327
11. Li, Y., Lyu, S.: Exposing deepfake videos by detecting face warping artifacts. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition *(*2019)