PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

DISEÑO DE UNA ARQUITECTURA DE APRENDIZAJE AUTOMÁTICO QUE BRINDE SOPORTE PARA LA DETECCIÓN DE MENTIRAS MEDIANTE EL ANÁLISIS DE VIDEO.

TESIS PARA OPTAR POR EL TÍTULO DE MAGÍSTER EN INGENIERÍA INFORMÁTICA CON MENCIÓN EN CIENCIAS DE LA COMPUTACIÓN

DIEGO ANDRÉS SALAS GUILLÉN

ASESOR: DR. IVAN ANSELMO SIPIRAN MENDOZA

Lima, julio de 2019

RESUMEN

La justicia y la búsqueda de la verdad en la investigación criminal requiere del uso de una herramienta fundamental para su éxito, el interrogatorio. En un interrogatorio, un experto hace uso de su experiencia y su juicio para, mediante el cuestionamiento del acusado, obtener una verdad explícita o implícita de parte de este sobre el hecho a investigar.

El presente proyecto de investigación apunta a diseñar un modelo de aprendizaje automático que brinde soporte para la detección de mentiras en interrogatorios mediante el análisis de video. Es una contribución a los trabajos de investigación realizados por el grupo IA-PUCP (Grupo de Investigación en Inteligencia Artificial) de la Pontificia Universidad Católica del Perú. Se utilizó un conjunto de datos puesto a disponibilidad por Rada Mihalcea del grupo “Language and Information Technologies” de la Universidad de Michigan.

La propuesta de arquitectura para el modelo consiste en una capa de preprocesamiento de datos que utiliza un algoritmo de reconocimiento facial para extraer los rostros del video, limitando el espacio de características. Luego, se utiliza una red convolucional preentrenada para realizar la extracción de características. Finalmente, se utiliza una red recurrente LSTM para procesar las características y luego una red neuronal para clasificar los videos.

Se experimentó con cinco redes convolucionales (Resnet, InceptionV3, Xception, VGG16 y VGG19), el mejor fue InceptionV3. Este obtuvo una exactitud de 78.6 %, valor que supera varios de los resultados obtenidos por los modelos, presentados en la publicación “A Multi-View Learning Approach to Deception Detection” de N. Carissimi, que no aplicaron entrenamiento en la extracción convolucional. Esto, utilizando menos información y automatizando la extracción de la misma.

Dedicado a mi padre y a mi madre por su apoyo incondicional.
Por confiar en mi y hacer un gran esfuerzo para darme
la oportunidad de salir adelante. A Zully Justino,
por su comprensión, cariño y apoyo.

Agradezco al doctor Ivan Sipiran por
su asesoría y dirección en el trabajo de investigación
y a la doctora Loreta Gasco por su interés en el tema y
por su apoyo en temas de estadística.

Índice General

1. Planteamiento del proyecto **1**

1.1. Definición del Problema 1

1.2. Objetivos 4

1.3. Herramientas, Métodos, Metodologías y Procedimientos 5

1.4. Alcance 7

1.5. Justificativa y viabilidad del proyecto 9

2. Marco conceptual **12**

2.1. Marco teórico 12

2.2. Estado del arte 21

3. Extracción de características **27**

3.1. Descripción del conjunto de datos 27

3.2. Componente de extracción de catacterísticas 32

3.3. Propuesta de Arquitectura 36

3.4. Discusión de resultados 37

4. Aprendizaje automático **38**

4.1. Modelo de aprendizaje automático 38

4.2. Arquitectura de análisis de video 39

4.3. Discusión de resultados 41

5. Capítulo 5: Ajuste y evaluación **43**

5.1. Estrategia de ajuste del modelo 43

5.2. Ajuste y evaluación del modelo predictivo 43

5.3. Discusión de resultados 47

6. Capítulo 6: Conclusión y trabajos futuros **50**

6.1. Conclusión 50

6.2. Trabajos futuros 51

7. Bibliografía **52**

Índice de Figuras

1. Valor de activación de una neurona[26] 13
2. Arquitectura de una red convolucional[15] 15
3. Fórmula de recurrencia una red neuronal recurrente[13] 17
4. Fórmula de una red LSTM[13] 18
5. Fórmula de una red GRU[13] 19
6. Características Haar[30] 19
7. Imagen Integral[30] 20
8. Clasificadores en cascada[30] 21
9. Nube de palabras: Herramientas para procesamiento de expresiones faciales en imágenes y video 24
10. Ejemplo de los frames obtenidos del video de muestra 32
11. Ejemplo de un mapa de características obtenidos del frame de muestra . . 34
12. Ejemplo de los mapas de características obtenidos del frame de muestra . 35
13. Propuesta de arquitectura 36
14. Arquitectura implementada: Preprocesamiento de datos 39
15. Arquitectura implementada: Extracción y entrenamiento 40
16. Resulado base: Perdida en el conjunto de validación y entrenamiento . . . 41
17. Resultado de la extracción de rostros 44
18. Ruido en la extracción de rostros 45
19. Reducción del espacio de características: Perdida en el conjunto de validación y entrenamiento 46
20. Pruebas con una red GRU: Perdida en el conjunto de validación y entrenamiento 46
21. Tabla de resultados presentada en [4] 48

Índice de Cuadros

1. Resultados esperados por objetivo 5
2. Herramientas, Métodos, Metodologías 5
3. Riesgos y contingencia 8
4. Revisión sistemática: Bases de conocimiento 22
5. Conjunto de entrenamiento 28
6. Conjunto de validación 29
7. Resultados de la experimentación por modelo convolucional 42
8. Resultados de la segunda experimentación por modelo convolucional . . . 47
9. Cuadro de diagnóstico 48

1. Planteamiento del proyecto

1.1. Definición del Problema

1.1.1. Problemática

1.1.1.1. Introducción

El papa Juan Pablo II en el contexto de la Jornada Mundial para las Comunicaciones Sociales nos regala una reflexión sobre la búsqueda de la verdad. ”La libertad de buscar y decir la verdad es un elemento esencial de la comunicación humana, no sólo en relación con los hechos y la información, sino también y especialmente sobre la naturaleza y destino de la persona humana, respecto a la sociedad y el bien común, respecto a nuestra relación con Dios”[1].

La justicia y la búsqueda de la verdad en la investigación criminal requiere del uso de una herramienta fundamental para su éxito, el interrogatorio. En un interrogatorio, un experto hace uso de su experiencia y su juicio para, mediante el cuestionamiento del acusado, obtener una verdad explícita o implícita de parte de este sobre el hecho a investigar.

Existen muchas estrategias para lograr este objetivo. Principalmente se realiza un análisis entre el lenguaje verbal y no verbal para encontrar fugas no verbales. Una fuga no verbal es una forma de lenguaje corporal que ocurre cuando una persona verbaliza una cosa pero su lenguaje corporal indica otra. Otra estrategia es la comparación de hechos para encontrar contradicciones. Una historia inventada requiere de mucho trabajo mental para mantener la coherencia. El interrogador puede hacer uso de un listado de preguntas estratégicas y su percepción para combinar estas estrategias y encontrar contradicciones en la narración o fugas verbales que delaten al acusado.

En este proyecto de tesis se busca aplicar técnicas de aprendizaje automático para brindar soporte al interrogador en el contexto descrito. Los algoritmos de aprendizaje automático han obtenido resultados beneficiosos en muchos contextos relacionados al análisis de expresiones faciales[32][16][11][33][12]. Se plantea un experimento para de-

[1] MENSAJE DEL SANTO PADRE JUAN PABLO II PARA LA 37ª JORNADA MUNDIAL DE LAS COMUNICACIONES SOCIALES - 2003

terminar si es posible diseñar un modelo de aprendizaje automático que brinde soporte para facilitar la tarea de determinar si una persona miente o dice la verdad en un interrogatorio.

1.1.1.2. Situación Actual

El proceso de detección de mentiras se basa en el supuesto de que hay hechos contradictorios, falsos y verdaderos, que generan conflicto en el sujeto que miente[8]. La tarea del interrogador es buscar estas contradicciones en los distintos canales de comunicación que utiliza el interrogado[17]. Puede hacer una análisis de la voz, las expresiones corporales y la coherencia de los hechos relatados para acumular evidencia suficiente para concluir que el interrogado miente[17].

En esta situación el interrogador se vale de sus capacidades y su talento. Se puede entonces detectar posibles problemas en su rol de observador.

La capacidad auditiva del observador podría no permitir detectar y analizar en simultáneo todos los signos no verbales auditivos que delatan la mentira. Personas con la capacidad de controlar su vocalización y timbre de voz pueden ocultar fácilmente la mentira[17].

La capacidad visual, mnémica y de concentración del observador podría no permitir detectar y analizar en simultáneo todos los signos no verbales visuales que delatan la mentira. Personas con gran capacidad de control sobre los movimientos de su cuerpo y rostro pueden ocultar fácilmente la mentira[17].

Se requiere de alta capacidad mnémica y de herramientas auxiliares, por lo general, de operación manual para hacer seguimiento a los hechos relatados en un interrogatorio[8]. Personas con gran capacidad creativa y buena memoria pueden construir historias irreales y darles coherencia[8]. Resulta entonces difícil y se requiere un análisis muy incisivo para encontrar contradicciones.

Otro problema detectado es que la información de los interrogatorios es confidencial por lo que no se cuenta con ejemplos concretos para utilizar en investigación o capacitación.

El interrogador se basa solo en su experiencia y su talento, esto reduce la cantidad de personal apto y entorpece la tarea de descubrir la verdad.

El análisis que realiza el interrogador es sobre una captura cruda de la realidad, esto es posible porque nuestra mente tiene la capacidad de análisis para procesar esta información. Un algoritmo no puede trabajar de manera tan eficiente sobre la información cruda obtenida en un video. Es necesario un preprocesamiento de esta información para realizar un análisis más preciso.

La precisión de la herramienta es crítica pues se corre un alto riesgo al afirmar que una persona miente cuando esta dice la verdad. El interrogador requiere de herramientas cuya precisión sea alta para que la retroalimentación no lo lleve a concluir de manera errónea.

1.1.1.3. Situación Deseada

El experimento se plantea con el fin de validar la posibilidad de ofrecer al interrogador una herramienta que permita detectar un posible indicio de mentira en base a la información visual. Para realizar el análisis, la herramienta requerirá un preprocesamiento de la información. Este deberá ser, en la medida de lo posible, automatizado. Este preprocesamiento incluye manipulación de la información y extracción de características. Esta herramienta debe tener una alta precisión pues sus resultados influyen en una decisión crítica.

Se trabajará el problema visual, con un conjunto de videos de personas mintiendo y diciendo la verdad. Estos contarán con capturas del rostro de un individuo en una situación real ocultando la verdad o mintiendo. También se contará con videos de individuos diciendo la verdad. Ambos conjuntos de datos estarán etiquetados para saber cuales contienen mentiras y cuáles no. Esta información permitirá entrenar y evaluar un modelo de aprendizaje automático que analice la información visual durante un testimonio para determinar en qué momentos es probable que el interrogado mienta.

Se utilizará un conjunto de datos puesto a disponibilidad por Rada Mihalcea del grupo

“Language and Information Technologies” de la Universidad de Michigan. Este conjunto de datos está disponible en: `http://web.eecs.umich.edu/~mihalcea/downloads.html#RealLifeDeception`.

1.2. Objetivos

1.2.1. Objetivo General

Diseñar una arquitectura de aprendizaje automático que brinde soporte para la detección de mentiras en interrogatorios mediante el análisis de video.

1.2.2. Objetivos Específicos

1.2.2.1. Objetivo Específico 1

Diseñar los componentes de una arquitectura de aprendizaje automático que automatice la extracción de características de capturas de una persona mintiendo.

1.2.2.2. Objetivo Específico 2

Diseñar los componentes de una arquitectura de aprendizaje automático que permita determinar la probabilidad de que se esté mintiendo en una captura de video de una persona mintiendo.

1.2.2.3. Objetivo Específico 3

Realizar el ajuste del modelo propuesto para optimizar el resultado segun las métricas pertinentes.

1.2.3. Resultados Esperados

En el cuadro 1 se presentan los resultados esperados propuestos para cada objetivo específico.

Cuadro 1: Resultados esperados por objetivo

Objetivos	Resultados	
OE1	RE1	Descripción y exploración del conjunto de datos.
	RE2	Evaluacion del componente de extracción de características.
	RE3	Propuesta de arquitectura e integración del componente.
OE2	RE4	Evaluación del componente de predicción en base a las características extraidas.
	RE5	Integración del componente a la arquitectura propuesta.
OE3	RE6	Estrategia de ajuste de los resultados arquitectura.
	RE7	Evaluación del modelo predictivo final

1.3. Herramientas, Métodos, Metodologías y Procedimientos

1.3.1. Introducción

El presente proyecto de investigación aplicada se desarrolla en el área de ciencias de la computación, particularmente en la subárea de aprendizaje automático y visión computacional. Las principales herramientas, que serán el núcleo de la investigación, son algoritmos de aprendizaje automático para extracción de características, algoritmos de aprendizaje automático para clasificación y métodos de validación de modelos de aprendizaje automático.

A continuación se detallarán, además de las principales herramientas a utilizar, metodologías y métodos auxiliares que permitirán alcanzar los resultados esperados.

Cuadro 2: Herramientas, Métodos, Metodologías

Resultado Esperado	Herramientas, Métodos, Metodologías
RE1	Aplicativo de procesamiento de videos ffmpeg. Python, opencv.
RE2	Aplicativo de procesamiento de videos ffmpeg. Google colab. Python, paquetes keras y tensorflow.
RE3	Python, paquetes keras y tensorflow. Google Draw.io.

Resultado Esperado	Herramientas, Métodos, Metodologías
RE4	Python, paquetes keras y tensorflow. Grid-search para ajuste de hiperparámetros. Métricas de evaluación de modelos de aprendizaje.
RE5	Google colab. Google Cloud Platform. Python, paquetes keras y tensorflow.
RE6	Revisión del estado del arte. Métricas de evaluación de modelos de aprendizaje.
RE7	Python, paquetes keras y tensorflow. OpenCV (haarcascades).

1.3.2. Herramientas

Las principales herramientas son el lenguaje de programación Python con los respectivos paquetes y Pycharm como IDE principal. Adcionalmente se utilizó git para el versionamiento del código. Como infraestructura, se realizó pruebas y ejecuciones en google colab y google cloud platform. Se utilizaron los siguientes paquetes de python:

- Scikit-learn[22], herramientas de aprendizaje automático.
- Opencv[3], herramientas para análisis de video.
- Keras[6], herramientas de aprendizaje automático.
- Tensorflow[1], herramientas de aprendizaje automático.

Se utilizó los IDEs PyCharm[2] para desarrollo local en Python y Google Colab[3] para desarrollo en python. Este último permite contar con un entorno de trabajo estilo Jupiter Notebook y permite contar con GPU Tesla K80 para el procesamiento de imágenes y video.

Principalmente se utilizó infraestructura alojada en la nube proveeida por Google. Se cuenta con acceso Google Drive con capacidad ilimitada como medio de almacenamiento y con el espacio de trabajo proporcionado por Google Colab y Google Cloud Platform.

[2]Disponible en https://www.jetbrains.com/pycharm/

[3]Disponible en: https://colab.research.google.com/

Se trabajó con carpetas sincronizadas entre ambos servicios de Google. Adicionalmente se cuenta con una MacBook Air core i7 (2.2GHz) con 8 GB de RAM y un disco externo de 500GB.

1.4. Alcance

A continuación se detallará el alcance del proyecto, justificando las limitaciones que existen para el dominio en cuestión y los riesgos a los que está sujeto.

El proyecto a desarrollar trata del diseño de un modelo de aprendizaje automático para dar soporte a la detección de mentiras durante un interrogatorio. Se centra en el preprocesamiento de los datos para el entrenamiento y el diseño de un modelo de aprendizaje automático que pueda clasificar los fragmentos de grabaciones de video en dos categorías, miente y no miente. Es un proyecto de investigación aplicada en la subárea de visión computacional, en el área de la inteligencia artificial.

Para la realización de este proyecto, se ha identificado las limitaciones y alcance para que sea viable su desarrollo en el transcurso del periodo académico definido por la universidad.

El alcance está sujeto a las siguientes decisiones:

- Utilizar un problema reducido y simulado, no se trabajará con interrogatorios criminales por la complejidad de acceder a información confidencial y crítica.
- Se utilizará un conjunto de grabaciones de juicios públicos clasificados en mentiras y verdades.

1.4.1. Limitaciones

Los obstáculos y restricciones que limitaron el alcance son los siguientes:

- La naturaleza del fenómeno a analizar es compleja y exiten casos en los que la persona puede encubrir muy bien las fugas no verbales.
- La calidad de los videos y la ubicación de la cámara son muy variables en la muestra a utilizar. Los rostros no siempre se enfocan desde la misma perspectiva y en

algunos videos hay frames que no contienen el rostro del acusado.

- El tiempo y la capacidad de procesamiento para realizar el entrenamiento de un modelo de aprendizaje automático son bastante elevados.

1.4.2. Riesgos

En el cuadro 3 se presentan los riesgos identificados para el proyecto, así como su probabilidad de ocurrencia, impacto en el proyecto y las medidas de contingencia que se tomarán en caso ocurran.

Cuadro 3: Riesgos y contingencia

Id	Riesgo	Probabilidad (0..1)	Impacto (0..1)	Contingencia
1	Resultados forzados	0.3	1.0	Se mitigará manteniendo la integridad del conjunto de pruebas y mediante el uso de herramientas que garanticen la replicabilidad del experimento.
2	Disponiblidad de recursos de infraestructura	0.5	0.3	Se cuenta con dos posiblilades en la nube (Google Colab y Google Cloud Platform). La primera ofrece instancias por un tiempo limitado y la segunda instancias por subasta que pueden ser asignadas a otro usuario. En ambos casos se automatizará el despliegue de la arquitectura para evitar problemas al cambiar de instancia.

Id	Riesgo	Probabilidad (0..1)	Impacto (0..1)	Contingencia
3	Ambigüedad sobre la relación causal	0.3	0.8	Una fuga no verbal asociada a la mentira puede ser producto de los nervios u otra causa. Este riesgo es difícil de mitigar pues es ineherente a cada individuo. Se asumirá el riesgo ya que el conjunto de pruebas corresponde a situaciones de la vida real.
4	Disponibilidad de recursos económicos	0.1	0.8	Se cuenta con un fondo propio de contingencia en caso alguno de los ingresos de dinero programados se retrase.

1.5. Justificativa y viabilidad del proyecto

1.5.1. Justificativa

A continuación se presentan los motivos de selección del tema de investigación, la motivación del proyecto y los beneficios que nos impulsan a culminar con éxito esta iniciativa.

- El contexto nacional donde la corrupción y la mentira son pan de cada día motivó el dominio de aplicación.
- Contribuir con la resolución de problemas haciendo uso de las posibilidades que brinda el aprendizaje profundo.
- Sentar un precedente en el estudio de la mentira, en el contexto de la universidad, utilizando herramientas de aprendizaje profundo.

La labor del interrogador en el proceso de descubrir la verdad pone a prueba muchas de sus habilidades en simultáneo. Tiene que estar atento a varios canales: visual, auditivo y

al contenido de lo que relata el interrogado. Esto genera una carga mental fuerte y pone a prueba su concentración.

El principal objetivo de la investigación es aplicar las tecnologías en cuestión para dar soporte a este proceso, optimizando la captura de alguno de estos canales. Mediante herramientas de captura y procesamiento de video, el experto podrá contar con una fuente de información más precisa sobre el canal visual. Esto no significa que el investigador vaya a dejar de lado este canal, pues es necesario que lo considere para contrastarlo con los otros en busca de contradicciones.

Sin embargo, la herramienta permitirá extraer del video información útil para resolver el problema, basandose en modelos y valores ajustados durante el entrenamiento. Este enfoque permitirá que el interrogador cuente con retroalimentación sobre el efecto que causó su pregunta en el interrogado. Le permitirá saber si ha logrado tocar un tema crítico y le permitirá ser más incisivo cuando lo requiera. Se apunta a potenciar su capacidad de analizar el canal visual para mejorar su desempeño.

Finalmente, cabe resaltar que el proyecto contribuirá a potenciar, con el aporte de investigación en el tema de visión computacional, los trabajos de investigación realizados por el grupo IA-PUCP (Grupo de Investigación en Inteligencia Artificial) de la Pontificia Universidad Católica del Perú.

1.5.2. Viabilidad

A continuación se analizará la viabilidad del proyecto basada en tres aspectos generales: Recursos, Tiempo y Herramientas.

Respecto a los recursos disponibles para el desarrollo del proyecto se cuenta con acceso a la información necesaria para la construcción del modelo, así como con conjunto de datos que ha sido utilizado en experimentos afines. Se cuenta además con recursos materiales y humanos que provee la universidad, y principalmente IA-PUCP, como acceso a bases de conocimiento, equipos de cómputo, asesores e investigadores dispuestos a brindar su apoyo y ambientes de trabajo.

Por parte del equipo del proyecto se cuenta con computadoras personales y de escritorio, todas con un desempeño satisfactorio. También se cuenta con disponibilidad económica para costear los gastos a lo largo del desarrollo del proyecto.

El proyecto se desarrollará en diez meses, duración del semestre académico, tiempo que se considera razonable para cumplir con los objetivos dado el alcance determinado para el proyecto. La disponibilidad de recursos humanos es parcial, aproximadamente 20 horas-hombre semanales. Se espera que la motivación personal del ejecutor del proyecto impulse significativamente el progreso del mismo.

Finalmente, las herramientas de software a utilizarse para el desarrollo del proyecto son de código abierto (OpenSource) por lo que no limitarán económicamente el desarrollo del proyecto. Otras herramientas serán desarrolladas por lo que su costo corresponde a las horas dedicadas a su elaboración. La mayor parte del costo corresponde a la adquisición de los equipos físicos e infraestructura en la nube.

2. Marco conceptual

2.1. Marco teórico

2.1.1. Aprendizaje automático

El aprendizaje es un fenómeno multifacético que incluye la adquisición de conocimiento declarativo, desarrollo de habilidades cognitivas a través de la práctica, la organización de nuevo conocimiento en representanciones efectivas y el descubrimiento de nuevos hechos a través de la observación y la experimentación[18].

Desde los inicios de la era de la computación, los investigadores han tratado de simular estas capacidades en las computadoras[18]. El estudio y modelado computacional del proceso de aprendizaje constituye el sujeto de estudio del aprendizaje automático[18].

2.1.2. Redes neuronales

Un acercamiento al problema del aprendizaje automático ha sido el desarrollo de modelos matemáticos simplificados de sistemas similares al cerebro[26]. Estos han sido estudiados para entender como pueden ser utilizados para resolver varios problemas computacionales[26].

Aunque los detalles de la propuesta varían, el grueso de los modelos utilizan la neurona como la unidad básica de procesamiento[26]. Cada unidad de procesamiento se caracteriza por un nivel de actividad (que representa su estado de polarización), un valor de salida (que representa el ratio de disparo de la neurona), un conjunto de conexiones (que representan la sinapsis entre la célula y su dendrita), un sesgo (que representa un nivel interno de reposo de la neurona) y un conjunto de conexiones de salida (que representan las proyecciones axonales de la neurona)[26].

Cada uno de esos aspectos de la unidad están representados matemáticamente por números reales. Cada conexión tiene asociado un peso (fuerza sináptica) que determina el efecto de la entrada en el nivel de activación de la neurona[26]. Los pesos pueden ser positivos (exitadores) o negativos (inhibidores)[26]. Frecuentemente, las entradas se suman linealmente obteniendo un valor de activación para la unidad i en el tiempo t dado

$$Ni(t) = \sum Wij * Xj + Bi$$

Imagen 1: Valor de activación de una neurona[26]

por 1:

Donde Wij es el peso de la conexión entre la unidad i y la j, Bi es el sesgo de la unidad y Xj es la salida de la unidad j.[26].

El problema de aprendizaje en las redes neuronales se reduce al problema de encontrar un conjunto de pesos de las conexiones de la red que le permita realizar el cálculo computacional deseado[26].

Las redes neuronales pueden ajustar su comportamiento para el reconocimiento de patrones, toma de decisiones, control de sistemas, predicción y más[31]. La auto-optimización le permite a la red neuronal ajustarse a sí misma[31]. El encargado del diseño del sistema define la arquitectura de la red y determina como la red se concatena a otras partes del sistema, luego elige una metodología de entrenamiento[31]. Entonces, la red neuronal se adapta a la aplicación[31].

2.1.3. Aprendizaje profundo

Los métodos de aprendizaje profundo son métodos de aprendizaje por representación con múltiples niveles de representación obtenidos mediante la composición de módulos simples pero no lineales que transforman una representación de un nivel (entradas crudas inicialmente) a otro de mayor abstracción[15]. Con la composición de estas transformaciones se puede aprender funciones muy complejas[15]. Para tareas de clasificación, capas altas de representación amplifican aspectos de la entrada que son importantes para la discriminación y suprimen variaciones irrelevantes[15].

El aprendizaje profundo está logrando grandes avances resolviendo problemas que han resistido por años los mejores intentos de la comunidad de la inteligencia artificial[15]. Ha demostrado ser un método muy bueno para descubrir estructuras intrincadas en es-

tructuras altamente dimensionales de datos y, por tanto, aplicable a muchos dominios[15].

Una arquitectura de aprendizaje automático es una pila multicapas de módulos simples, la mayoría sujeto a aprendizaje, muchos de los cuales computan mapeos no lineales entre sus entradas y salidas[15]. Cada módulo en la pila transforma su entrada para incrementar la selectividad y la invarianza de la representación[15]. Con múltiples capas no lineales (profundidades de 5 a 20 por ejemplo) un sistema puede implementar funciones extremadamente complejas de su entrada que son sensibles a detalles mínimos (diferencias entre samoyedos y lobos blancos) y no son sensibles a variaciones irrelevantes (el fondo, posición o iluminación)[15].

2.1.4. Redes neuronales convolucionales

Las redes convolucionales están diseñadas para procesar data que viene estructurada en arreglos múltiples, por ejemplo, una imagen a color compuesta de tres arreglos 2D con la intensidad de píxeles en cada canal de color[15]. Hay cuatro ideas principales detrás de las redes convolucionales que aprovechan las propiedades de las señales naturales: conexiones locales, pesos compartidos, agrupamiento (pooling) y el uso de múltiples capas.[15].

En la imagen 2 se presenta la arquitectura de una red convolucional típica estructurada en etapas. Las primeras etapas están compuestas de dos tipos de capas: convolucionales y de agrupamiento[15]. Las unidades en una capa de una red convolucional se organizan en mapas de características, cada unidad está relacionada a un conjunto de unidades locales en la capa anterior a través de un conjunto de pesos llamado filtro[15].

El resultado de esta suma local se pasa por una función no lineal como ReLU, por ejemplo[15]. Todas las unidades en un mapa de características comparten el mismo filtro[15]. Diferentes mapas de características en una capa utilizan diferentes filtros[15].

Hay dos motivos detrás de esta arquitectura. En datos estructurados en arreglos como lo son las imágenes, grupos de valores locales están altamente correlacionados, formando patrones locales que se pueden detectar fácilmente[15]. Por otro lado, los valores

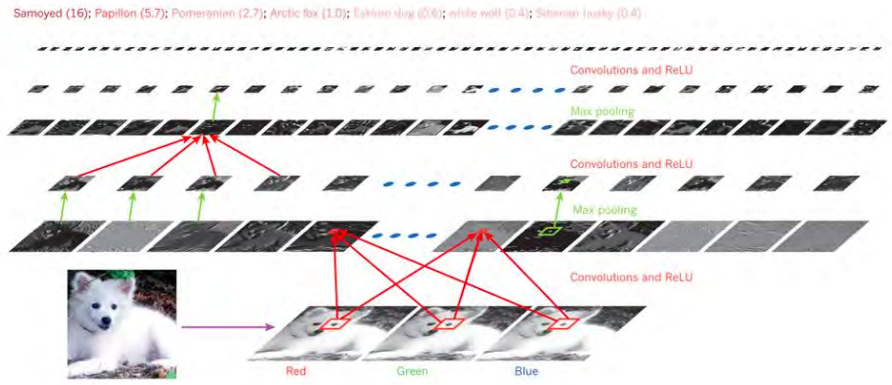

Imagen 2: Arquitectura de una red convolucional[15]

estadísticos locales de las imágenes y otras señales son invariantes respecto a su ubicación[15]. Si un patrón está presente, puede aparecer en una parte de la imagen o puede que aparezca en cualquier otro lado[15]. Es por esto que se plantea la idea de usar unidades en diferentes lugares compartiendo los mismos pesos y la detección de los mismos patrones en diferentes partes del arreglo[15]. Matemáticamente la operación de filtrado realizada por un mapa de características es una convolución discreta, de aquí el nombre de la red[15].

El rol de las capas convolucionales es detectar conexiones locales de características de capas previas[15]. Por otro lado, el rol de las capas de agrupamiento es agrupar características similares en una sola[15]. La posición relativa de características formando un patrón puede ser variable, detectar de manera confiable el patrón puede realizarse granularizando bruscamente la posición de cada característica[15]. Una unidad de agrupamiento típica contiene el valor máximo de un conjunto de unidades locales en un mapa de características (o en unos cuantos mapas)[15]. Las unidades vecinas de un agrupamiento reciben su entrada de grupos que son desplazados por una o más columnas, reduciendo de esta manera la dimensión de la representación y generando invarianza a pequeños cambios o distorsiones[15]. Se suelen repetir etapas de convoluciones y agrupamiento seguidas de capas completamente conectadas[15]. Realizar la propagación hacia atrás en una red convolucional no tiene mayor complejidad que hacerlo en una red neuronal profunda regular, esto permite entrenar los pesos en los filtros[15].

Las redes neuronales profundas explotan la propiedad de que muchas señales naturales son composiciones jerárquicas en las que características de alto nivel se obtienen componiendo características de bajo nivel[15]. En imágenes, combinaciones locales de ejes forman patrones, los patrones se ensablan en partes y las partes forman objetos[15]. El agrupamiento permite que las representaciones varíen muy poco cuando elementos en la capa previa varían en posición y apariencia[15].

Las capas convolucionales y de agrupamiento en las redes neuronales están directamente inspiradas por nociones clásicas de células simples y complejas en neurociencia visual[15]. Cuando se presenta una imagen a un simio y a una red neuronal la activación de las unidades de alto nivel de la red neuronal explican la mitad de la varianza de conjuntos aleatorios de 160 neuronas en la corteza inferotemporal del simio[15].

2.1.5. Redes pre-entrenadas

Un supuesto en muchos algoritmos de aprendizaje automático y minería de datos es que el entrenamiento y la data futura deben estar en el mismo espacio de características y tener la misma distribución. Sin embargo, en muchas aplicaciones del día a día, este supuesto no se cumple[19]. Por ejemplo, en ocasiones el problema de un dominio de interés con datos limitados cuenta con suficientes datos en otro dominio de interés, aunque el espacio de características o la distribución de las mismas varíe[19]. En esos casos, la transferencia de conocimiento, de realizarse correctamente, mejora el desempeño del aprendizaje ahorrando esfuerzos en etiquetado de datos[19].

2.1.6. Redes neuronales recurrentes

Las redes neuronales recurrentes, específicamente la variante Memoria a largo y corto plazo (LSTM por sus siglas en inglés) figuran recientemente como un modelo efectivo en aplicaciones que involucran datos secuenciales[13]. Entre estos, están incluidos el modelamiento de lenguaje, el reconocimiento de escritura manual, la traducción automática el reconocimiento del habla, el análisis de video y el subtitulado de imágenes[13].

El modelo más simple de una red recurrente organiza vectores de estados ocultos $h(l,t)$ en una red de dos dimensiones donde $t\epsilon 1...T$ se considera el tiempo y $l\epsilon 1...L$ es la

$$h_l^t = tanhW^l \binom{h_t^{l-1}}{h_{t-1}^l}$$

Imagen 3: Fórmula de recurrencia una red neuronal recurrente[13]

profundidad[13]. La primera fila de vectores $h(0,t) = x(t)$ en la profundidad cero, contiene los vectores de entrada $x(t)$ y cada vector en la fila superior $h(L,t)$ es usado para predecir un vector de salida $y(t)$[13]. Todos los vectores intermedios $h(l,t)$ son calculados con una formula de recurrencia basada en $h(l,t-1)$ y $h(l-1,t)$ [13]. A través de estos vectores ocultos cada salida $y(t)$ en el tiempo t se convierte en una función de todos los vectores de entrada hasta $t\{x(1), x(2)..., x(t)\}$[13].

La fórmula matemática de recurrencia $(h(l,t-1), h(l-1,t)) \rightarrow h(l,t)$ varía de modelo a modelo, en la imagen 3 se presenta la fórmula de recurrencia de una red neuronal recurrente común, se asume que todos los $h \epsilon R(n)$[13]. La matriz de parámetros WI en cada capa tiene dimensiones $[nx2n]$ y tanh se aplica elemento por elemento[13]. WI varía entre capas pero es compartida a través del tiempo[13]. Las entradas, capa inferior $h(l-1,t)$ y capa previa en el tiempo $h(l,t-1)$ se transforman mediante interacción aditiva antes de ser comprimidas por tanh[13].

2.1.7. Redes LSTM y GRU

Una de las variantes más exitosas de redes neuronales recurrentes es la red de memoria a largo y corto plazo la cual (LSTM), en principio, almacena y recupera información sobre largos periodos de tiempo con mecanismos explícitos de compuertas y un carrusel de error constante incorporado[13].

La red LSTM fue diseñada para enfrentar las dificultades de entrenar una red neuronal recurrente[13]. Se observó que las dinámicas de retro propagación causaban que desaparezcan o exploten las gradientes en una RNN. Se descubrió luego que este problema de que exploten las gradientes podría ser mitigado truncando las gradientes en un valor máximo[13]. Por otro lado, las redes LSTM fueron diseñadas para mitigar el problema de

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \qquad \begin{aligned} c_t^l &= f \odot c_{t-1}^l + i \odot g \\ h_t^l &= o \odot \tanh(c_t^l) \end{aligned}$$

Imagen 4: Fórmula de una red LSTM[13]

desvanecimiento de gradiente[13]. Además del vector oculto de estado $h(l,t)$ las redes LSTM mantienen un vector de memoria $c(l,t)$[13]. En cada etapa la red puede escoger si leer de, escribir en, o restaurar la celda usando mecanismos explícitos de compuertas.

En la imagen 4 se muestra la fórmula utilizada para la actualización, las funciones sigm y tanh se aplican elemento a elemento y la matriz de pesos Wl tiene tamaño $[4n * 2n]$[13]. Los vectores i, f , $o \epsilon R(n)$ representan compuertas binarias que controlan si cada celda de memoria es actualizada, restaurada a cero y si su estado local es revelado en el vector oculto, respectivamente[13]. Su activación se basa en la función sigmoide y su valor oscila entre 0 y 1[13].

El vector $g \epsilon R(n)$ se encuentra en el rango -1 y 1, y es utilizado para modificar de manera aditiva los contenidos de la memoria[13]. Esta interacción aditiva es importante pues durante la retro propagación, una suma solo distribuye las gradientes. Esto permite a las gradientes fluir por las celdas de memoria c a través del tiempo ininterrumpidas durante largos periodos hasta que se active una compuerta de olvido[13]. Es importante notar que la red LSTM debe mantener dos vectores $c(l,t)$ y $h(l,t)$ en cada punto de la red[13].

Una unidad recurrente de compuertas (GRU, por sus siglas en inglés) es una alternativa más simple a la red LSTM[13]. En la imagen 5 se muestra la fórmula correspondiente a una red GRU, $W(l,r)$ es una matriz de $[2n * 2n]$ y $W(l,x)$ es una matriz de $[n * n]$[13]. La red GRU calcula un vector oculto candidato e interpola suavemente hacia el utilizando la compuerta z[13].

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \qquad \begin{aligned} c_t^l &= f \odot c_{t-1}^l + i \odot g \\ h_t^l &= o \odot \tanh(c_t^l) \end{aligned}$$

Imagen 5: Fórmula de una red GRU[13]

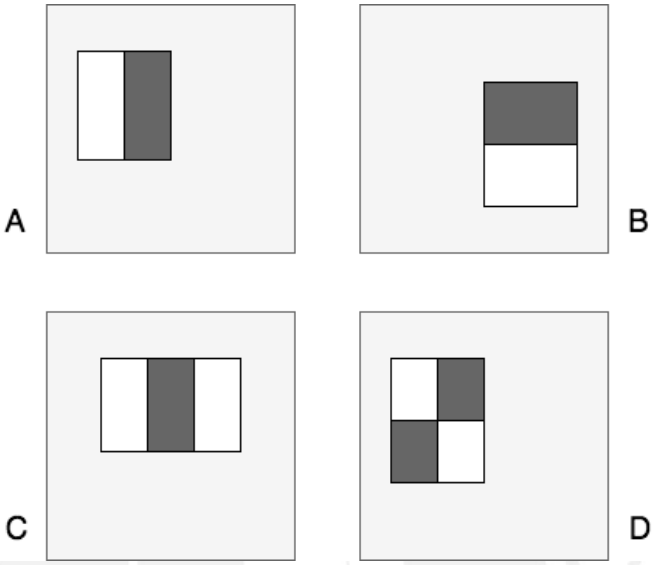

Imagen 6: Características Haar[30]

2.1.8. Características Haar para detección de objetos

En [30] se propone un procedimiento de clasificación de imágenes basado en el valor de características simplificadas. Estas características usualmente contienen conocimiento ad-hoc del dominio del problema que es difícil de asimilar usando una cantidad finita de datos de entrenamiento[30]. Además, es mucho más rápido trabajar con características que con píxeles[30].

Las características utilizadas para detección de objetos están basadas en funciones Haar[30]. En la imagen 6 se presenta un ejemplo de la extracción de estas características[30]. Se suma los píxeles de los rectángulos blancos y se restan de la suma de píxeles de los rectángulos negros[30].

Se utilizan tres tipos de características, de dos regiones, de tres regiones y de cuatro

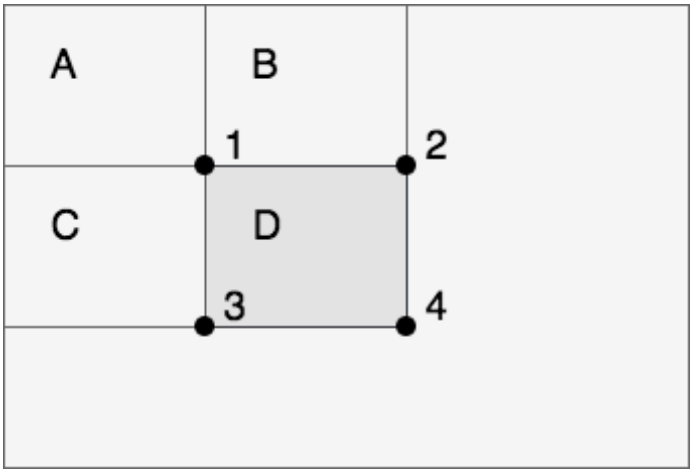

Imagen 7: Imagen Integral[30]

regiones[30]. El cálculo de este tipo de características se puede realizar rápidamente usando una representación intermedia de la imagen llamada imagen integral[30].

La imagen integral contiene, en un punto (x, y), la suma de los píxeles hacia arriba y hacia la derecha del punto[30]. De esta manera se puede calcular la suma de píxeles de una región D, como se muestra en la imagen 7, realizando la operación $4+1-(2+3)$[30].

Dado un conjunto de características y un conjunto de imágenes etiquetadas se pueden usar muchas aproximaciones para aprender una función de clasificación[30]. En el sistema planteado se selecciona un conjunto reducido de características, para esto se utiliza un mecanismo llamado cascada atencional[30]. Esto permite reducir significativamente el número de características procesadas[30].

La cascada atencional se basa en la intuición de que clasificadores más simples pueden rechazar muchas sub ventanas negativas (partes de la imagen que no contienen el objeto a detectar) mientras, detectando casi todas las ventanas positivas[30]. Los clasificadores simples se utilizan para rechazar la mayoría de sub ventanas mientras que clasificadores más complejos se centran en determinar las ventanas positivas en el conjunto restante[30].

Un resultado positivo del primer clasificador activa la evaluación del clasificador siguiente

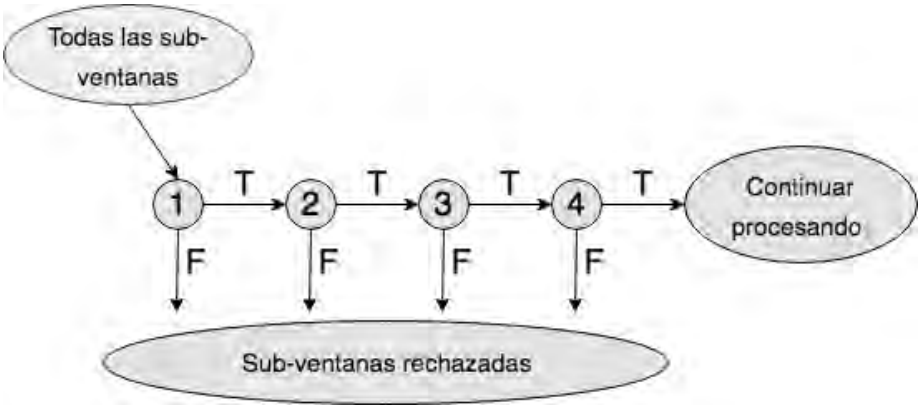

Imagen 8: Clasificadores en cascada[30]

y así en adelante[30]. Un resultado negativo en cualquier punto lleva al rechazo de la sub ventana[30]. Cada etapa de la cascada se construye entrenando clasificadores usando AdaBoost y ajustando el límite para minimizar falsos negativos[30].

En la imagen 8 se muestra el funcionamiento de los clasificadores en cascada. Cada etapa se encarga de eliminar ejemplos negativos[30]. A mayor profundidad, mayor es la complejidad del clasificador[30].

2.2. Estado del arte

2.2.1. Objetivos

Con el fin de conocer las herramientas más adecuadas para hacer frente al problema planteado se revisará el reciente desenvolvimiento de las tecnologías de interés en los últimos años.

Los objetivos de la revisión del estado del arte son:

- Examinar qué herramientas de aprendizaje automático se utilizan en el análisis de expresiones faciales tanto en imágenes como en video.
- Examinar qué herramientas permiten obtener mejores resultados.
- Examinar qué bases de datos se han utilizado y cuales podrían contribuir a resolver el problema planteado utilizando la estrategia de transferencia de aprendizaje.

2.2.2. Revisión sistemática

El método utilizado para la revisión del estado del arte es la revisión sistemática. Se realizará una búqueda de fuentes primarias relevantes siguiendo una pregunta que oriente la búsqueda. Se planteó la siguiente pregunta para direccionar la búsqueda:

¿Qué herramientas de aprendizaje profundo se utilizan en la actualidad para el procesamiento de rostros en imagen y video?

Al tratarse de un tema vigente que avanza a pasos agigantados se consideró limitar la revisión a publicaciones de los años 2017 y 2018 (año actual). A continuación se listan las bases de conocimiento revisadas y la consulta realizada en cada una:

Cuadro 4: Revisión sistemática: Bases de conocimiento

Base de conocimiento	Consulta	Resultados
Scopus	TITLE-ABS-KEY (automatic AND recognition AND (facial OR face OR expression) AND (image OR video)) AND (LIMIT-TO (SUBJAREA , ÇOMP ")) AND (LIMIT-TO (PUBYEAR , 2018) OR LIMIT-TO (PUBYEAR , 2017))	245 - 2017 108 - 2018
ACM	query: { acmdlTitle:(+recognition +automatic image video face facial expression) AND recordAbstract:(+recognition +automatic image video face facial expression) } filter: {publicationYear:{ gte:2017 }}, {owners.owner=HOSTED}	22 - 2017 1 - 2018
IEEE	(.Abstract":automatic AND recognition AND (face OR facial OR expression) AND (image OR video) OR p_Title:automatic AND recognition AND (face OR facial OR expression) AND (image OR video))	223 - 2017 37 - 2018

Los criterios de inclusión considerados son los siguientes:

- Presenta análisis de imágenes o videos de rostros utilizando feature extraction y deep learning.
- Utiliza herramientas de deep learning/aprendizaje automático para analizar rostros en imágenes o video.
- Presenta herramientas de automatización de extracción de características para análisis de imágenes y videos.

2.2.3. Conclusiones de la revisión sistemática

De una base de 150 artículos examinados, se incluyó 50 en la revisión de acuerdo a los criterios de inclusión definidos. Se llegó a las siguientes conclusiones:

Se utiliza dos enfoques para abordar el problema, ambos parten de la caracterización del conjunto de datos y el aprendizaje en base a esta caracterización. El primer enfoque realiza una extracción de características y aplica algún método de aprendizaje automático sobre las características extraídas.

El segundo enfoque utiliza aprendizaje profundo para realizar dentro de una misma arquitectura la caracterización y el aprendizaje. De este modo uno puede retroalimentarse de los resultados del otro, obteniendo mejores resultados. Este segundo enfoque también permite evitar errores humanos en la caracterización pues deja en manos del modelo la decisión de elegir las mejores características y su relevancia para resolver el problema.

En la imagen 9 se puede observar un resumen de las herramientas listadas durante el proceso de revisión. Se puede observar que las herramientas más utilizadas son las redes neuronales. Dentro de las herramientas de extracción de características una de las más utilizadas es el filtro de Gabor, también presente en la imagen. Por el lado de las herramientas de aprendizaje automático se tiene las máquinas de soporte vectorial y la redes neuronales, estas son las que se utilizan para el aprendizaje en el primer enfoque presentado.

Imagen 9: Nube de palabras: Herramientas para procesamiento de expresiones faciales en imágenes y video

En relación al segundo enfoque se puede ver la presencia de la palabra “convolutional” y las siglas cnn haciendo referencia a las redes neuronales convolucionales, que son la herramienta principal de este enfoque. También se puede observar las siglas lstm (long short term memory), un componente que permite mantener una relación temporal durante el análisis de características. Este se suele utilizar en el procesamiento de lenguaje natural, aunque también ha demostrado ser útil en el análisis de video. Se encontró dos artículos que trabajaron con el mismo conjunto de datos a utilizar. El primero [23], plantea una extracción manual de cacterísticas que luego son alimentadas a un modelo de aprendizaje automático. El segundo [4] obtiene muy buenos resultados utilizando modelos pre entrenados para automatizar la extracción de características, este es el enfoque que se trabajará en el presente proyecto.

El dataset que se utilizará en el presente proyecto se introdujo en [23] con la propuesta de utilizar características lingüísticas y gestuales manualmente extraídas y un clasificador basado en árboles de decisión (random forest) para determinar si un testimonio es verdadero o falso. En este estudio se alcanzó una precisión de entre 60 y 75 %.

En [4] se plantea una aproximación que aprovecha las nuevas tendencias en análisis de imagen y video. En este artículo se plantea una aproximación multivistas al problema. Cada vista corresponde a un grupo de características específico. La extracción de características se realizó utilizando modelos de aprendizaje profundo pre-entrenados. Las características extraídas se complementaron con otros grupos de características obtenidas utilizando el esquema de codificación MUMIN[2], Facial action units (Facial Action Coding System)[9] y n-grams para características textuales.

Los modelos de deep learning para extracción de características utilizados fueron VGG-Face[21], AlexNet [14], GoogLeNet[28] y ResNet[10]. Entre estos AlexNet obtuvo mejores resultados[4].

Se comparó el desempeño de tres clasificadores: MVL (multi-view learning)[4], SVM y LMKL(Localized Multiple Kernel Learning)[4]. Curiosamente, el clasificador MLV supera a los clasificadores SVM y LMKL salvo para las características extraidas por AlexNet[4].

Los mejores resultados se obtuvieron cuando se realizó el re-entrenamiento parcial del modelo pre-entrenado utilizando el conjunto de datos. Estos resultados alcanzaron un valor de casi 100 %[4]. Durante la revisión también se encontraron artículos que tocaron temas similares al planteado en el proyecto en cuestión, estos se resumen a continuación:

En [25] se plantea una propuesta de experimento que resultaría útil para la construcción de un conjunto de datos localizado. Se describe un experimento en el que los participantes toman el rol de mentiroso o sincero en un interrogatorio sobre acciones realizadas en un experimento conducido. Como parte de los trabajos futuros se considera la construcción de un conjunto de datos representativo de un contexto de aplicación.

En [24] y se plantea el uso de redes neuronales recurrentes (LTSM) para el análisis de expresiones faciales que delatan dolor. Esta propuesta resulta bastante interesante pues se trata de un problema bastante similar al que se está atacando en el proyecto. Otro artículo que aprovecha este tipo de redes es [16]. En este se utilizan para distinguir entre expresiones faciales reales y fingidas.

3. Extracción de características

3.1. Descripción del conjunto de datos

Se utilizará un conjunto de datos puesto a disponibilidad por Rada Mihalcea del grupo ”Language and Information Technologies”de la Universidad de Michigan. Este consiste en 121 videos de testimonios reales tomados de grabaciones de juicios en Estados Unidos. De estos 60 corresponden a testimonios reales y 61 a testimonios falsos. El conjunto de datos ha sido estudiado en los artículos [23] y [4] presentados en la revisión del estado del arte.

Los videos se obtuvieron de grabaciones de juicios considerando algunas restricciones. Las capturas de la defensa o el testigo debe estar claramente identificada, el rostro debe ser visible durante la mayor parte de la duración del video y la calidad del video debe ser suficiente para identificar las expresiones faciales del sujeto[23].

La duración promedio de los videos en el conjunto de datos es de 28 segundos[23]. La longitud promedio para los testimonios falsos es de 27.7 segundos y para los reales es de 28.3 segundos[23]. Los videos corresponden a 21 mujeres y 35 hombres con edades entre los 16 y 60 años[23].

Se utilizó la librería scikit-learn [22] para separar los videos en conjuntos de validación y entrenamiento con un 33 % para el conjunto de validación. En los cuadros 5 y 6 se muestran los conjuntos obtenidos:

Cuadro 5: Conjunto de entrenamiento

Entrenamiento			
Mentira		Verdad	
trial_lie_002.mp4	trial_lie_034.mp4	trial_truth_001.mp4	trial_truth_032.mp4
trial_lie_004.mp4	trial_lie_036.mp4	trial_truth_003.mp4	trial_truth_033.mp4
trial_lie_005.mp4	trial_lie_037.mp4	trial_truth_006.mp4	trial_truth_034.mp4
trial_lie_006.mp4	trial_lie_039.mp4	trial_truth_007.mp4	trial_truth_035.mp4
trial_lie_007.mp4	trial_lie_040.mp4	trial_truth_008.mp4	trial_truth_037.mp4
trial_lie_008.mp4	trial_lie_041.mp4	trial_truth_010.mp4	trial_truth_038.mp4
trial_lie_010.mp4	trial_lie_042.mp4	trial_truth_011.mp4	trial_truth_039.mp4
trial_lie_012.mp4	trial_lie_043.mp4	trial_truth_012.mp4	trial_truth_044.mp4
trial_lie_013.mp4	trial_lie_044.mp4	trial_truth_013.mp4	trial_truth_045.mp4
trial_lie_017.mp4	trial_lie_045.mp4	trial_truth_014.mp4	trial_truth_047.mp4
trial_lie_018.mp4	trial_lie_046.mp4	trial_truth_016.mp4	trial_truth_049.mp4
trial_lie_019.mp4	trial_lie_047.mp4	trial_truth_017.mp4	trial_truth_050.mp4
trial_lie_020.mp4	trial_lie_048.mp4	trial_truth_018.mp4	trial_truth_052.mp4
trial_lie_022.mp4	trial_lie_051.mp4	trial_truth_021.mp4	trial_truth_053.mp4
trial_lie_023.mp4	trial_lie_052.mp4	trial_truth_024.mp4	trial_truth_054.mp4
trial_lie_024.mp4	trial_lie_053.mp4	trial_truth_025.mp4	trial_truth_055.mp4
trial_lie_025.mp4	trial_lie_054.mp4	trial_truth_027.mp4	trial_truth_056.mp4
trial_lie_029.mp4	trial_lie_056.mp4	trial_truth_028.mp4	trial_truth_057.mp4
trial_lie_030.mp4	trial_lie_057.mp4	trial_truth_029.mp4	trial_truth_058.mp4
trial_lie_032.mp4	trial_lie_059.mp4	trial_truth_031.mp4	trial_truth_059.mp4
	trial_lie_061.mp4		

Cuadro 6: Conjunto de validación

Validación			
Mentira		Verdad	
trial_lie_001.mp4	trial_lie_028.mp4	trial_truth_002.mp4	trial_truth_030.mp4
trial_lie_003.mp4	trial_lie_031.mp4	trial_truth_004.mp4	trial_truth_036.mp4
trial_lie_009.mp4	trial_lie_033.mp4	trial_truth_005.mp4	trial_truth_040.mp4
trial_lie_011.mp4	trial_lie_035.mp4	trial_truth_009.mp4	trial_truth_041.mp4
trial_lie_014.mp4	trial_lie_038.mp4	trial_truth_015.mp4	trial_truth_042.mp4
trial_lie_015.mp4	trial_lie_049.mp4	trial_truth_019.mp4	trial_truth_043.mp4
trial_lie_016.mp4	trial_lie_050.mp4	trial_truth_020.mp4	trial_truth_046.mp4
trial_lie_021.mp4	trial_lie_055.mp4	trial_truth_022.mp4	trial_truth_048.mp4
trial_lie_026.mp4	trial_lie_058.mp4	trial_truth_023.mp4	trial_truth_051.mp4
trial_lie_027.mp4	trial_lie_060.mp4	trial_truth_026.mp4	trial_truth_060.mp4

Para comenzar con las pruebas de extracción de características se utilizó la librería Open CV[3] para separar los videos en frames. Se calculó el total de frames por video y se tomaron muestras cada 2, 4 y 5 frames para obtener diferentes subconjuntos.

```
import cv2

cap = cv2.VideoCapture(sample_video_path)
n_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
frames = []

for i in range(n_frames):
    name = './tmp_frame_data/frame' + str(i) + '.jpg'
    frames.append(name)

frame_lenghts = {}
offsets = [1,2,4,5]
for i in offsets:
  offset = i
  m_index = "Offset " + str(offset)
  frame_lenghts[m_index] = "Frames " + str(len(frames[::offset]))
print(frame_lenghts)

```

```
Output:
{'Offset 1': 'Frames 300',
 'Offset 2': 'Frames 150',
 'Offset 4': 'Frames 75',
 'Offset 5': 'Frames 60'}
```

Código 1: Conteo de frames

El código para realizar la captura de los frames se muestra en el código 2:

```
import os

# Playing video from file:
cap = cv2.VideoCapture(sample_video_path)

try:
    if not os.path.exists(sample_frame_data_path):
        os.makedirs(sample_frame_data_path)
except OSError:
    print ('Error: Creating directory of data ' + sample_frame_data_path)

currentFrame = 0
success = True
while(success):
    # Capture frame by frame
    success, frame = cap.read()
    if (not success):
      break
    # Saves image of the current frame in jpg file
    name = sample_frame_data_path + '/frame' + str(currentFrame) + '.jpg'
    print ('Creating: ' + name)
    cv2.imwrite(name, frame)
    currentFrame += 1

cap.release()
cv2.destroyAllWindows()
```

Código 2: Extracción de frames

Se utilizó la función VideoCapture para cargar el video de muestra. Luego, se leyó uno a

uno los frames y se guardaron con el nombre numerado en un directorio.

En la imagen 10 se presenta un ejemplo de algunos de los frames obtenidos para el video muestra.

Imagen 10: Ejemplo de los frames obtenidos del video de muestra

3.2. Componente de extracción de catacterísticas

Para el componente de extracción de características se utilizó una red neuronal convolucional. Se trabajó con la estratégia de transferencia de aprendizaje. Es decir, se utilizó los pesos de una red convolucional entrenada sobre un conjunto de imágenes. Para eso se cargó varias arquitecturas de redes convolucionales con los pesos entrenados en el conjunto imagenet[7].

Se utilizó la librería keras de python para obtener los modelos pre entrenados. Los modelos utilizados fueron:

- InceptionV3[29]
- Xception[5]
- VGG16[21]
- VGG19[21]
- ResNet50[10]

En el código 3 se muestra la creación de un modelo y un ejemplo de extracción de características para un frame de muestra:

```
import numpy as np
from keras.preprocessing import image
from keras.applications.inception_v3 import InceptionV3
from keras.applications.inception_v3 import preprocess_input
from keras.applications.xception import Xception
from keras.applications.vgg16 import VGG16
from keras.applications.vgg19 import VGG19
from keras.applications.resnet50 import ResNet50
import matplotlib.pyplot as plt

 def get_base_model(model_name = 'inception_v3'):
        if model_name == 'inception_v3':
            return InceptionV3(weights='imagenet', include_top=False)
        elif model_name == 'xception':
            return Xception(weights='imagenet', include_top=False)
        elif model_name == 'vgg16':
            return VGG16(weights='imagenet', include_top=False)
        elif model_name == 'vgg19':
            return VGG19(weights='imagenet', include_top=False)
        elif model_name == 'resnet50':
            return ResNet50(weights='imagenet', include_top=False)
        else:
            raise ValueError('Cannot find base model %s' %self.model_name)

def plot_feature_maps(feature_maps):
  i=0
    j=0
    offset = 29
    height, width, depth = feature_maps.shape
    nb_plot = int(np.rint(np.sqrt(depth/offset)))
    fig = plt.figure(figsize=(20, 20))
    top = (((depth/offset)/nb_plot)*nb_plot)
    while j < depth:
        plt.subplot(nb_plot, nb_plot, i+1)
        plt.imshow(feature_maps[:,:,j], cmap='gray')
        plt.title('feature map {}'.format(i+1))
        j=j+offset
        i=i+1
        if (i > top-1): break
    plt.show()
```

```
41
42 bModel = get_base_model()
43
44 img_path = 'frame192.jpg'
45 img = image.load_img(img_path, target_size=(224, 224))
46 img_data = image.img_to_array(img)
47 img_data = np.expand_dims(img_data, axis=0)
48 img_data = preprocess_input(img_data)
49
50 mFeatures = bModel.predict(img_data)
51 print(mFeatures.shape)
52
53 # Print one feature map
54 plt.imshow(mFeatures[0, :, :, 0], cmap='gray')
55
56 # Print feature maps
57 plot_feature_maps(mFeatures[0])
```

Código 3: Modelo de extracción de características

Las imágenes se generaron a partir de las características extraidas en la capa final del modelo InceptionV3[29] entrenado en imagenet[7]. Se utilizó uno de los frames extraídos del video de muestra y se realizó la predicción del modelo. La impresión del mapa de características muestra la estructura de la última capa (1, 5, 5, 2048). Se cuenta con 2048 imágenes de 5x5 que resumen la características del frame utilizado. Las imágenes obtenidas para la impresión de uno y varios mapas de características son 11 y 12

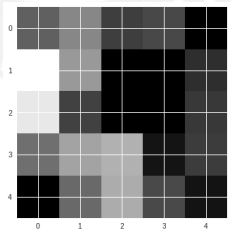

Imagen 11: Ejemplo de un mapa de características obtenidos del frame de muestra

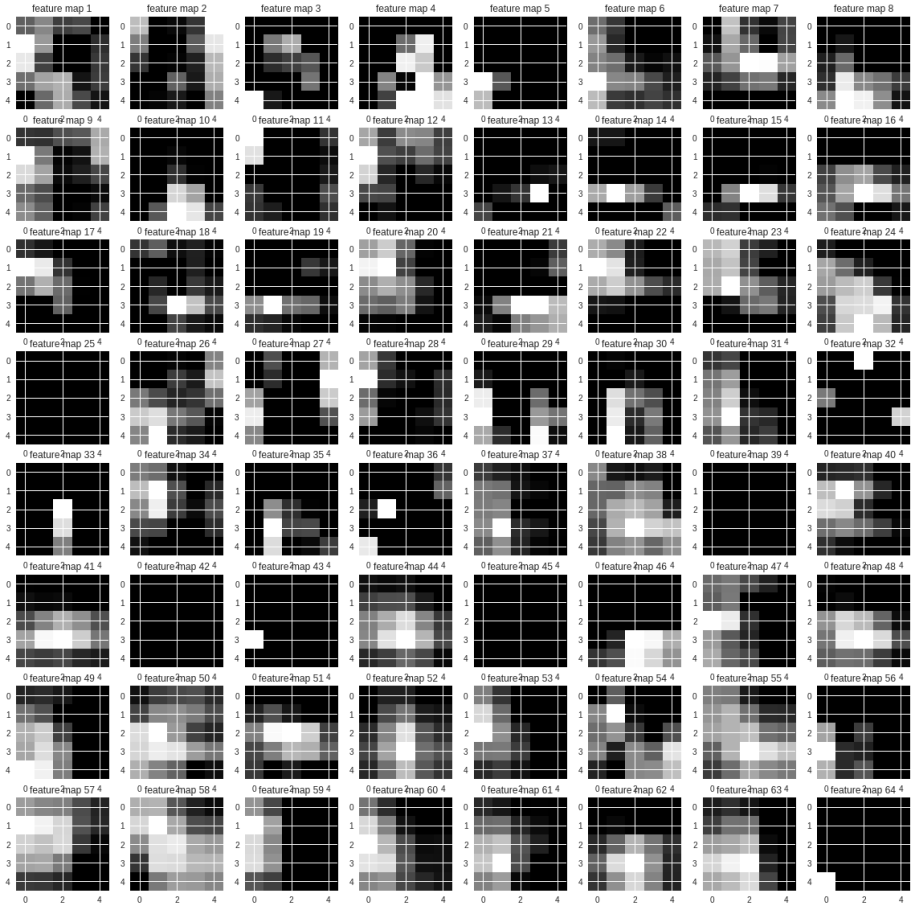

Imagen 12: Ejemplo de los mapas de características obtenidos del frame de muestra

3.3. Propuesta de Arquitectura

En la imagen 13 se presenta la propuesta de arquitectura para realizar la extracción de características y clasificación de videos.

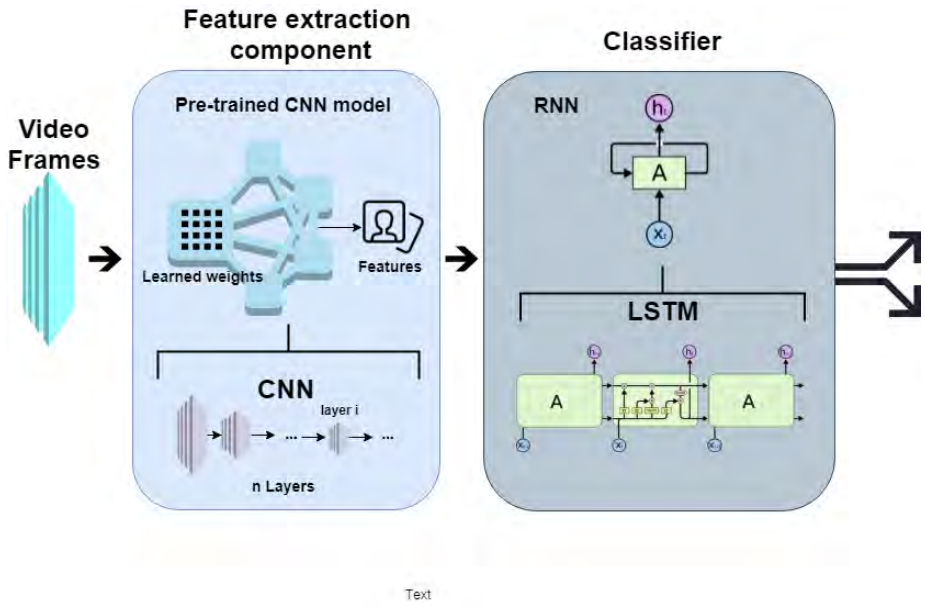

Imagen 13: Propuesta de arquitectura

Para cada video se realizará la extracción de los frames y se extraerá las características de cada frame. Para esto se utilizará un modelo pre entrenado de una red neuronal convolucional. Se aprovechará el entrenamiento realizado sobre un conjunto más completo de imágenes, imagenet. Los pesos calculados se utilizarán para extraer las características relevantes de los frames extraidos. Este experimento se repetirá utilizando cada una de las redes convolucionales revisadas previamente.

La información extraída será utilizada por un modelo clasificador para dar una respuesta. Se propone utilizar una red neuronal recurrente pues esta tiene en cuenta la temporalidad de los frames para tomar una decisión. En base a la información levantada en el estado del arte, se decidió utilizar el tipo de red LSTM. La información procesada por esta red será alimentada a una red neuronal que devuelva una probabilidad para cada posible respuesta.

3.4. Discusión de resultados

La integración a la arquitectura se realizó tomando la última capa de los modelos. Se cargó el modelo deseado como modelo base y se realizó la captura del output de la capa indicada.

```
        self.model = Model(
                inputs=base_model.input,
                outputs=base_model.get_layer('avg_pool').output
        )
```

Código 4: Obtención de la capa de salida de un modelo

Para el caso de VGG19 y VGG16 esta capa era “flatten”, para el resto era “avg_pool”. Estas son las últimas capas de procesamiento antes de llegar a las capas de predicción. Se encargan de transformar y resumir la información de la imagen en un conjunto de datos unidimensional.

Los modelos utilizados están entrenados para detección de imágenes por lo que su desempeño para el análisis no es óptimo. A pesar de esto los modelos permiten extraer características de cada frame y estas pueden utilizarse para realizar el análisis y el aprendizaje. Una propuesta de mejora es realizar pruebas con el modelo VGG Face[20] y ejecutar la extracción en base a un modelo entrenado para trabajar con rostros. Otra alternativa es realizar entrenamiento de ajuste para mejorar el resultado obtenido.

4. Aprendizaje automático

4.1. Modelo de aprendizaje automático

El modelo a utilizar es una red neuronal de memoria a corto y largo plazo (LSTM). Esta se alimentará de la información extraída de los frames de los videos. Se utilizó un generador de secuencias adaptado de [27] para alimentar la red. Este generador permite cargar las entradas desde disco evitando saturar la memoria del computador. Las secuencias de entrada generadas en base al modelo extractor se guardan en un archivo de NumPy y son servidas por el generador conforme se necesiten.

El modelo funcional se construyó utilizando un modelo secuencial compuesto por una red LSTM y dos redes densas. La red LSTM tiene 2048 unidades y dropout de 50 % y está conectada con una capa densa de 512 neuronas con activación relu y dropout de 50 %. Se realizó pruebas cambiando las unidades de la capa densa a 64, 128, 256, 512 y 1024. Finalmente, se utiliza otra capa densa con dos clases y activación softmax para obtener una probabilidad para cada clase. En el código 5 se muestra la definición básica de la red:

```
1     def lstm(self):
2         lstm_units = 2048
3         lstm_dropout = 0.5
4         dense_units = 512
5         dense_dropout = 0.5
6         model = Sequential()
7         model.add(LSTM(lstm_units, return_sequences=False,
8                        input_shape=self.input_shape,
9                        dropout=lstm_dropout))
10        model.add(Dense(dense_units, activation='relu'))
11        model.add(Dropout(dense_dropout))
12        model.add(Dense(self.nb_classes, activation='softmax'))
13        return model
```

Código 5: Modelo predictivo

4.2. Arquitectura de análisis de video

La arquitectura para el modelo de predicción construido puede separarse en tres etapas automatizadas. El preprocesamiento de datos, la extracción de características y el entrenamiento.

En la imagen 14 se muestra un diagrama del proceso de preparación de los datos. Lo primero que se hace es obtener los videos comprimidos, almacenados en google Drive. Adicionalmente se crean los directorios donde se guardarán archivos intermedios y resultados.

A continuación se utiliza el paquete scikit-learn[22] para separar los videos en conjuntos de validación y entrenamiento. Se reubican los videos en los directorios correspondientes. Se utiliza un 33 % para validación.

Finalmente, se realiza la extracción de frames de cada video utilizando el programa ffmpeg y se guarda información sobre la ubicación, el conjunto, la clase y la cantidad de frames del video en un archivo csv.

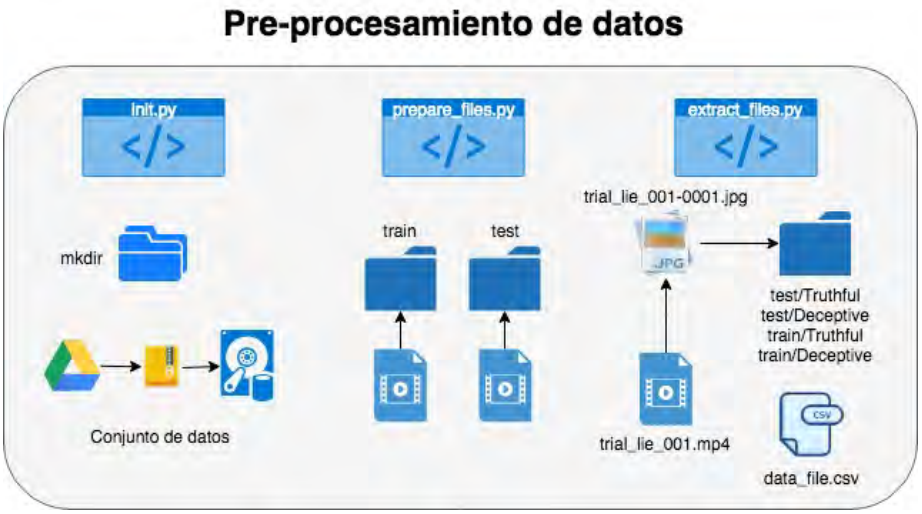

Imagen 14: Arquitectura implementada: Preprocesamiento de datos

En la imagen 15 se muestra un diagrama del proceso de extracción y entrenamiento. Se utiliza el archivo csv con el resumen de la información extraída para recuperar las capturas de un video. Estas se procesan utilizando alguno de los modelos convolucionales pre-entrenados mencionados en el capítulo 3. Se extrae la información de la capa avg_pool (flatten para VGG16 y VGG19) para generar un arreglo NumPy de características.

Durante este proceso se utiliza un parámetro llamado longitud de secuencia. Este determina cuantos frames serán utilizados para generar una secuencia de características. Se realiza una selección de frames en intervalos iguales hasta obtener la cantidad deseada. Las secuencias se guardan en un archivo NumPy.

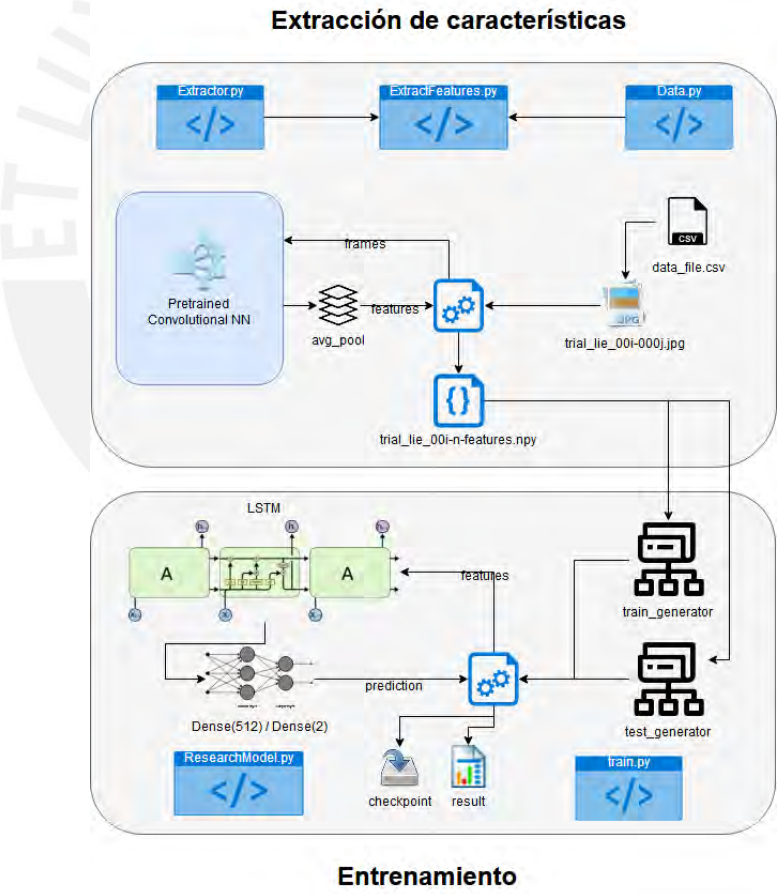

Imagen 15: Arquitectura implementada: Extracción y entrenamiento

Para realizar el entrenamiento se utilizó dos generadores que sirven secuencias desde el disco. Esto para evitar saturar la memoria del computador. Las secuencias fueron alimentadas la red LSTM y el resultado pasó por las redes densas. Se guardó los pesos del modelo al terminar cada época.

La métrica utilizada para la evaluación fue exactitud. Se obtuvo un valor de 65 %, el cual es bajo. En comparación con los resultados de 21, no se ha logrado superar ninguna de las exactitudes alcanzadas. El modelo extractor que obtuvo mejor resultado fue InceptionV3 con un 65 %. En la gráfica 16 se presenta la pérdida en los conjuntos de validación y entrenamiento para este modelo. Se puede ver que la pérdida es bastante inestable. Además, se tiene sobreajuste.

Imagen 16: Resulado base: Perdida en el conjunto de validación y entrenamiento

4.3. Discusión de resultados

Se realizaron cinco experimentos utilizando diferentes modelos convolucionales. En la tabla 7 se muestran los resultados obtenidos para cada uno de los modelos.

Cuadro 7: Resultados de la experimentación por modelo convolucional

Modelo Convolucional	Parámetros Entrenables	Número de capas	Exactitud
Resnet	23,534,592	174	0.55
Xception	20,806,952	131	0.63
InceptionV3	21,768,352	310	0.65
Vgg19	20,024,384	22	0.61
Vgg16	14,714,688	18	0.62

La métrica de prueba utilizada fue la exactitud para poder realizar la comparación con [4]. Se trabajó con entropia cruzada binaria (binary crossentropy) como perdida. Se alcanzó un 65 %, con el modelo convolucional InceptionV3, lo cual está por debajo de los resultados obtenidos en este artículo, ver imagen 21. La pérdida muestra un comportamiento muy inestable y existe sobreajuste. Además, la exactitud obtenida en validación es muy baja.

A pesar de este resultado, se logró construir un modelo funcional que realice la clasificación de los videos. Es importante resaltar la presencia de sobre ajuste, esto significa que el modelo tiene la capacidad para resolver el problema y que se puede mejorar a partir de este punto. Es necesario realizar ajustes en el modelo para obtener un mejor resultado.

5. Capítulo 5: Ajuste y evaluación

5.1. Estrategia de ajuste del modelo

En esta sección se plantean dos estratégias para mejorar los resultados obtenidos. La primera permite reducir el espacio de características. La segunda, reducir la complejidad del modelo de clasificación. Se cuenta con un modelo base que realiza la extracción automática de características de los frames de un video y utiliza una red LSTM y una capa densa para analizar la secuencia y devolver una clasificación.

Una de las primeras observaciones es que el espacio de características es muy amplio y tiene mucho ruido. La primera estratégia de ajuste apunta a reducir el ruido en el espacio de características. Se utilizará la estratégia de detección rápida de objetos planteada en [30] para detectar y extraer los rostros de cada frame. Esto ayuda a reducir el espacio de características significativamente y elimina buena parte de ruido, sin embargo, introduce otro tipo de ruido. Por la naturaleza de los videos, es posible que el rostro extraido no corresponda a la persona dando el testimonio.

Otra observación es que la red utilizada LSTM es una red recurrente con alta complejidad. Por este motivo, es más efectiva para grandes cantidades de información. El conjunto de datos actual es limitado en volúmen por lo que es posible que no se esté explotando completamente la red LSTM. Una alternativa interesante es utilizar una versión simplificada que trabaja de manera similar. Se espera que al utilizar una red GRU, Gated Recurrent Unit, que permitirá explotar de manera más efectiva el volúmen de datos con los que se está trabajando.

5.2. Ajuste y evaluación del modelo predictivo

En esta sección se presenta el procedimiento de ajuste del modelo construido, aplicando las estratégias planteadas. Para cada estratégia se presenta la implementación y los resultados.

La reducción del espacio de características se hizo usando la implementación del algoritmo presentado en [30] de Opencv[3] la cual cuenta con varios modelos para extracción

de rostros:

- haarcascade_frontalcatface
- haarcascade_frontalcatface_extended
- haarcascade_frontalface_alt
- haarcascade_frontalface_alt2
- haarcascade_frontalface_alt_tree
- haarcascade_frontalface_default

Se utilizó haarcascade_frontalcatface para detectar el rostro del individuo y reemplar cada frame con la imagen correspondiente al rostro detectado, para las imagenes que contenían más de un rostro se tomó el primer rostro encontrado en cada frame. Los frames en los que no se pudo detectar ningun rostro se descartaron. En la imagen 17 se muestran los resultados de la extracción de rostros.

Imagen 17: Resultado de la extracción de rostros

Algunos videos presentaban más de una persona hablando, estas generaron frames con ruido. En la imagen 18 se muestra un ejemplo de este problema. Se descartó los frames en los que no se detectó ningun rostro. Para los casos en los que se detectó más de un rostro se escogió uno aleatorio, en la mayoría de estos casos se cuenta con dos rostros y se requiere de mecanismos más complejos para distinguir al acusado. Se asume el riesgo de no escoger el rostro correcto y generar ruido para el modelo.

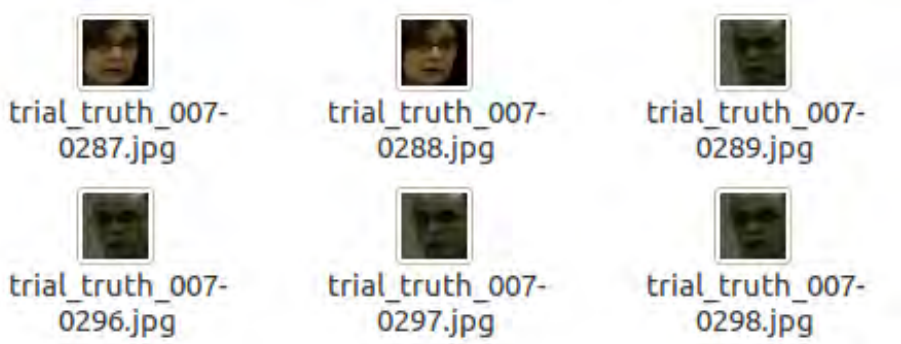

Imagen 18: Ruido en la extracción de rostros

Se repitió el proceso de extracción de secuencia y el entrenamiento. El modelo extractor de características con el mejor resultado fue InceptionV3. En la imagen 19 se presenta la pérdida en los conjuntos de validación y entrenamiento para este. Ya no está presente el sobre ajuste, reducir el espacio de características aumentó la capacidad del modelo de generalizar. Además, aumentó significativamente la exactitud a un 78.6 %.

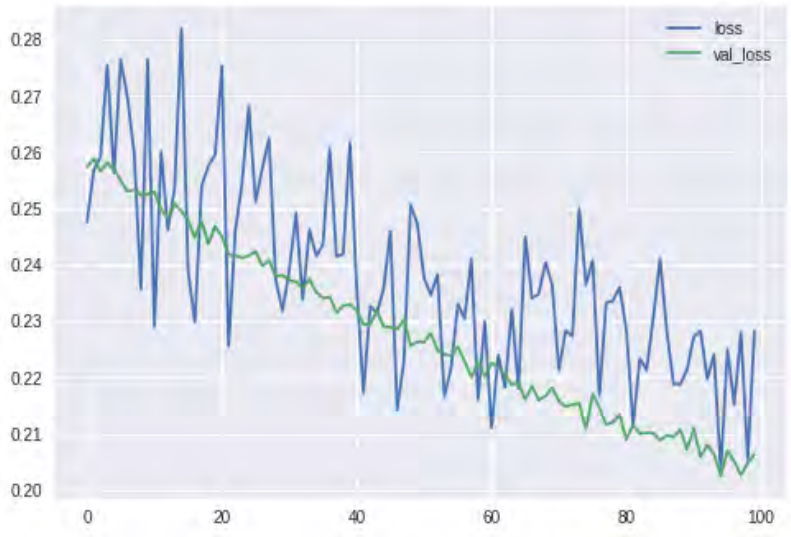

Imagen 19: Reducción del espacio de características: Perdida en el conjunto de validación y entrenamiento

El desempeño del modelo se redujo considerablemente al utilizar la red GRU. Solo se alcanzó un 56 % de exactitud. En la imagen 20 se puede ver que la perdida alcanza rápidamente un valor estable y oscila disminuyendo muy poco.

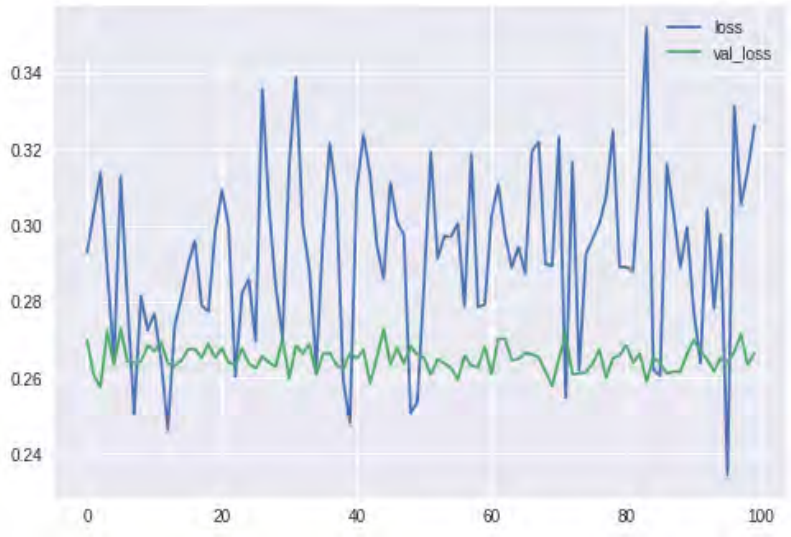

Imagen 20: Pruebas con una red GRU: Perdida en el conjunto de validación y entrenamiento

5.3. Discusión de resultados

En esta sección se discute los resultados de aplicar las dos estratégias de ajuste. En el cuadro 8 se presenta el resultado de la experimentación con cada modelo extractor y la mejora respecto a la experimentación previa.

Cuadro 8: Resultados de la segunda experimentación por modelo convolucional

Modelo Convolucional	Exactitud	Aumento en exactitud
Resnet	0.64	0.09
Xception	0.71	0.08
InceptionV3	0.79	0.14
Vgg19	0.68	0.06
Vgg16	0.76	0.14

La primera estratégia, reducir el espacio de características, se aplicó exitosamente mejorando la exactitud significativamente de 65 % a 78.6 %. Además, permitió eliminar el sobreajuste. En comparación a los resultados de [4] se logró superar varios de los modelos evaluados. En la imagen 21 se presentan estos resultados, excluyendo los que realizan afinación del modelo extractor de características. Se puede ver el resultado obtenido 78.6 % aún está por debajo de los modelos:

- Extracción manual con características adicionales y MVL.
- Extracción automática con VGGFace, características adicionales y MVL.
- Extracción automática con AlexNet, características adicionales y SVN.
- Extracción automática con AlexNet, características adicionales y LMKL.
- Extracción automática con ResNet, características adicionales y MVL.

Features	Method	Accuracy
Face-Manual & Others	SVM	0.67
Face-Manual & Others	LMKL	0.76
Face-Manual & Others	MVL	0.79
Face-AU & Others	SVM	0.63
Face-AU & Others	LMKL	0.67
Face-AU & Others	MVL	0.75
Face-VGGFace & Others	SVM	0.73
Face-VGGFace & Others	LMKL	0.76
Face-VGGFace & Others	MVL	**0.89**
Face-AlexNet & Others	SVM	0.80
Face-AlexNet & Others	LMKL	0.80
Face-AlexNet & Others	MVL	0.74
Face-ResNet & Others	SVM	0.71
Face-ResNet & Others	LMKL	0.73
Face-ResNet & Others	MVL	0.79
Face-GoogLeNet & Others	SVM	0.73
Face-GoogLeNet & Others	LMKL	0.74
Face-GoogLeNet & Others	MVL	0.78

Imagen 21: Tabla de resultados presentada en [4]

Contrario a lo esperado el desempeño del modelo no aumento al utilizar la red GRU. Esta red tiene menos complejidad que la LSTM y requiere menos procesamiento. Se partió de la asumsión de que la baja cantidad de información no contribuía al entrenamiento de una red compleja como la red LSTM y que bastaría con utilizar una red GRU. Sin embargo, los resultados indican que esta red aprende información más útil para realizar la predicción que la red GRU. Es decir, las compuertas adicionales de la red LSTM le están permitiendo mantener en memoria información más útil que la red GRU.

Los resultados para el problema de prueba utilizando son alentadores. Sin embargo, la evaluación de un modelo en producción requiere de mayor análisis. En el cuadro 9 se muestra los posibles resultados de la predicción contra la naturaleza del testimonio.

Cuadro 9: Cuadro de diagnóstico

		Naturaleza del testimonio	
		Mentira	Verdad
Predicción	Mentira	Verdadero positivo	Falso positivo
	Verdad	Falso Negativo	Verdadero Negativo

De todos los resultados el verdadero positivo y el verdadero negativo son los esperados por un buen modelo, entre estos es prioritario el verdadero positivo. Es necesario detectar la mayor cantidad de mentiras, siguiendo este criterio la primera métrica seleccionada es el recall. También resulta útil manejar el concepto de precisión, para tener una idea de cuantos de los aciertos son detecciones de mentiras.

Se realizó el cálculo de estas métricas para el mejor modelo como referencia útil para futuras comparaciones:

- Recall: 93.3 %
- Precision: 70.3 %

Los resultados obtenidos para el recall son favorables. Por otro lado, la presición indica que hay una probabilidad significativa de realizar una falsa acusación, lo cual es grave en el contexto del problema.

6. Capítulo 6: Conclusión y trabajos futuros

6.1. Conclusión

Se diseñó una arquitectura que permite obtener una solución al problema planteado. La arquitectura consiste en una capa de preprocesamiento de datos que utiliza un algoritmo de reconocimiento facial para extraer los rostros del video, limitando el espacio de características. Luego, se utiliza una red convolucional pre-entrenada para realizar la extracción de caracterísiticas. Finalmente, se utiliza una red recurrente LSTM para procesar las características y luego una red neuronal para clasificar los videos.

Es interesante resaltar que el uso de una red recurrente para el procesamiento permitió obtener resultados similares a otros modelos utilizando menos información de entrada. Solo se utilizó la información extraida de los frames de video por redes convolucionales, más no las características MUMIN, Facial Action Units y N-Grams

La primera propuesta presentó sobreajuste. A pesar de no obtener una buena exactitud permitió demostrar que el modelo podía dar solución al problema. En una segunda parte de esta investigación se plantearon estratégias para eliminar el sobre ajuste.

Además de realizar el ajuste de los hiperparámetros de la arquitectura se planteó dos estratégias. La primera permitió reducir el espacio de características, extrayendo solo las partes de interés de los cuadros del video. La extracción de rostros se realizó utilizando un método de deteción de objetos especializado en rostros[30].

Esto permitió eliminar el sobre ajuste y elevar la exactitud obtenida a un 78.6 %, valor que supera varios de los resultados obtenidos por los modelos de [4] que no aplicaron re-entrenamiento en la extracción convolucional. Se logró alcanzar este resultado reduciendo el esfuerzo en la generación de características y los recursos utilizados para el entrenamiento. . Además, se verificó la utilidad de una red recurrente en la exploración de expresiones faciales en secuencias de video.

6.2. Trabajos futuros

El resultado de esta investigación permitirá justificar la construcción de un conjunto de datos más completo. Se plantea generar videos de personas mintiendo en un interrogatorio simulado en base al experimento planteado en [25]. En este experimiento se condiciona a los participantes para mentir o decir la verdad en base a un escenario preparado previamente. Con este conjunto de datos se podrá entrenar el modelo para generalizar y se podrá proceder con la evaluación completa considerando subconjuntos de entrenamiento, validación y pruebas.

Otra propuesta interesante es utilizar otras fuentes de información, como hechos, transcripción y audio. Se deberá automatizar la extracción de características desde estas fuentes para enriquecer la entrada del modelo, manteniendo el concepto de extracción automática. Se plantea trabajar con un ensamble de modelos que combinen información de diversas fuentes, esto potencia la capacidad del modelo de generalizar la solución al problema estudiado. Se continuará realizando experimentación con las redes LSTM y GRU para determinar cómo varía su comportamiento con estas nuevas fuentes de datos.

Finalmente, se apunta a construir un producto funcional que pueda utilizarse para la detección de mentiras. Sobre un modelo extractor de características entrenado en una muestra representativa de una población se puede entrenar un modelo destinado a detectar la mentira en escenarios reales, sin embargo se deben teneren cuenta las consideraciones mencionadas en el capítulo 5.

De los errores que puede cometer el modelo, el falso positivo es el más perjudicial, esto quiere decir que se estaría acusando a una persona de mentir de manera injusta. Por otro lado, también es un error dejar pasar una mentira, pero es menos relevante que el primero. Es necesario, entonces reducir el error de tipo II. Esto entra en conflicto con el primer objetivo. Mientras más mentiras se quiera recuperar, más riesgo existe de cometer el error de tipo II. Se debe manejar un compromiso entre la necesidad de detectar mentiras y el riesgo de acusar injustamente a un interrogado. Este último problema planteado se puede mitigar utilizando el concepto de cobertura. El modelo sólo deberá emitir un juicio si la probabilidad de que este sea correcto está por encima de un límite definido.

7. Bibliografía

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[2] Allwood, J., Cerrato, L., Jokinen, K., Navarretta, C., and Paggio, P. (2007). The mumin coding scheme for the annotation of feedback, turn management and sequencing phenomena. *Language Resources and Evaluation*, 41(3-4):273–287.

[3] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[4] Carissimi, N., Beyan, C., and Murino, V. (2018). A multi-view learning approach to deception detection. In *Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on*, pages 599–606. IEEE.

[5] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *arXiv preprint*, pages 1610–02357.

[6] Chollet, F. et al. (2015). Keras. `https://keras.io`.

[7] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[8] Ekman, P. (2009). *Telling lies: Clues to deceit in the marketplace, politics, and marriage (revised edition)*. WW Norton & Company.

[9] Friesen, E. and Ekman, P. (1978). Facial action coding system: a technique for the measurement of facial movement. *Palo Alto*.

[10] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[11] Ilyas, C. M. A., Haque, M. A., Rehm, M., Nasrollahi, K., and Moeslund, T. B. (2018). Facial expression recognition for traumatic brain injured patients. In *VISIGRAPP (4: VISAPP)*, pages 522–530.

[12] Jahoda, P., Vobecky, A., Cech, J., and Matas, J. (2018). Detecting decision ambiguity from facial images. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 499–503. IEEE.

[13] Karpathy, A., Johnson, J., and Fei-Fei, L. (2015). Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

[14] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[15] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

[16] Li, L., Baltrusaitis, T., Sun, B., and Morency, L.-P. (2017). Combining sequential geometry and texture features for distinguishing genuine and deceptive emotions. In *2017 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 3147–3153. IEEE.

[17] Martínez Selva, J. M. (2005). La psicología de la mentira.

[18] Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (2013). *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.

[19] Pan, S. J., Yang, Q., et al. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[20] Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015a). Deep face recognition. In *British Machine Vision Conference*.

[21] Parkhi, O. M., Vedaldi, A., Zisserman, A., et al. (2015b). Deep face recognition. In *BMVC*, volume 1, page 6.

[22] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[23] Pérez-Rosas, V., Abouelenien, M., Mihalcea, R., and Burzo, M. (2015). Deception detection using real-life trial data. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 59–66. ACM.

[24] Rodriguez, P., Cucurull, G., Gonzalez, J., Gonfaus, J. M., Nasrollahi, K., Moeslund, T. B., and Roca, F. X. (2017). Deep pain: Exploiting long short-term memory networks for facial expression classification. *IEEE transactions on cybernetics*, -(99):1–11.

[25] Rothwell, J., Bandar, Z., O'Shea, J., and McLean, D. (2006). Silent talker: a new computer-based system for the analysis of facial cues to deception. *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, 20(6):757–777.

[26] Rumelhart, D. E., Widrow, B., and Lehr, M. A. (1994). The basic ideas in neural networks. *Communications of the ACM*, 37(3):87–93.

[27] Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402.

[28] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[29] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

[30] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE.

[31] Widrow, B., Rumelhart, D. E., and Lehr, M. A. (1994). Neural networks: applications in industry, business and science. *Communications of the ACM*, 37(3):93–106.

[32] Yun, W.-H., Lee, D., Park, C., Kim, J., and Kim, J. (2018). Automatic recognition of children engagement from facial video using convolutional neural networks. *IEEE Transactions on Affective Computing*.

[33] Zhuang, Y., Uribe, O., Mcdonald, M., Lin, I., Arteaga, D., Dalrymple, W., Worrall, B., Southerland, A., and Rohde, G. (2018). Pathological facial weakness detection using computational image analysis. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 261–264. IEEE.