Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Estimación de la veracidad de expresiones faciales utilizando aprendizaje profundo

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computaciónen Ciencias de la Computación

Gonzalo Fernández

Directora: María Elena Buemi

Buenos Aires, 2020

ESTIMACIÓN DE LA VERACIDAD DE EXPRESIONES FACIALES UTILIZANDO APRENDIZAJE PROFUNDO

En este trabajo se estudian diferentes enfoques basados en aprendizaje automático (en particular, variantes de redes neuronales artificiales) para clasificar instancias de expresiones faciales en video según su veracidad. Este problema tiene la particularidad, en comparación a la mayoría de problemas que las computadoras aprendieron a resolver utilizando inteligencia artificial, de que es una tarea que no es trivial de resolver para los seres humanos. Con ello surge la dificultad de evaluar el rendimiento de los modelos desarrollados.
Entre las múltiples aplicaciones que tiene este problema están mejorar la interacción humano-computadora, aumentar la efectividad de los robots asistentes, aportar en el tratamiento de desórdenes cognitivos crónicos, asistir investigaciones policiales, entre otros. También podría ser utilizado como herramienta para inferir qué tan bueno es un actor o para juzgar si un sospechoso dice la verdad.
Se utiliza como base de entrenamiento la SASE-FE que fue diseñada específicamente para resolver este problema en particular. Esta base contiene videos de sujetos realizando expresiones faciales, etiquetados según tipo de expresión y valor de verdad.
El principal análisis se basa en comparar redes neuronales profundas (feed-forward) con **redes neuronales recurrentes**. Este tipo particular de redes se caracteriza por su capacidad de extraer información de una secuencia y almacenarla a través del tiempo. Así, un video puede ser clasificado utilizando no sólo los atributos obtenidos en cada cuadro sino también los de sus antecesores.
Ante la escasez de datos para experimentar, se propone una nueva métrica para realizar un análisis más granular y la cual permite comparar con más detalle los resultados que arroja cada variante implementada. Los resultados sugieren que los rasgos determinantes que permiten distinguir entre una expresión sincera y una fingida están muy arraigados al sujeto que las ejecuta y, por lo tanto, desarrollar un clasificador universal (independiente del sujeto en cuestión) parece ser poco viable.
En cuanto a la comparación entre los dos tipos de redes, si bien las recurrentes no mejoraron los valores obtenidos por las profundas, sí se destaca que obtuvieron resultados similares con menor cantidad de épocas de entrenamiento.

Palabras claves: Expresiones Faciales, Landmarks, Redes Neuronales Recurrentes.

FACIAL EXPRESSIONS VERACITY ESTIMATION USING DEEP LEARNING

In this work we study many different machine learning based approaches (particulary, artificial neural networks variants) to classify instances of facial expressions on video according to it's veracity. This problem has the peculiarity, comparing to most problems that computers learned to solve using artificial intelligence, that is a task not trivial to solve to human beings. With that comes the difficulty of evaluating the performance of the developed models.
This problem has multiple aplications, like improving human-computer interaction, increasing assistant robots efectivity, helping in chronic cognitive disorders treatment, asisting policial investigations, among others. It also could be used as tool to infer an actor's performance or to judge if a suspect is telling the truth.
We use the SASE-FE dataset that was designed specifically to solve this particular problem. This dataset contains videos of subjects doing facial expresions, labeled with type of expresion and truth value.
The main analysis is based on comparing deep feed-forward neural networks with recurrent neural networks. this particular type of network is known for being capable of extract information from a sequence and keep it through time. That way, a video can be clasified using not only it's features but also the ones from it's predecessors.
Having so little data to experiment, we propose a new metric to make a more granular analysis and that allows compare with more detail the results that each implemented variant throws. Results suggest that determinants traits that allows distinguishing a genuine expresion and a faked one are too related to the subject that executes them and, then, developing an universal clasifier (independent of the subject) seems unfeasible.
Regarding the comparison between the two types of networks, although the recurrent variants couldn't overcome the values obtained by the deep variants, we can appreciate that they reach similar results but with a smaller amount of training epochs.

Keywords: Facial Expressions, Landmarks, Recurrent Neural Networks.

AGRADECIMIENTOS

A todes en el grupo de Procesamiento de Imágenes y Visión por Computadoras por animarme a terminar aunque yo ya no tuviera ganas de seguir.

A Alina Petra, quien me enseñó que vivir es mucho más que estar vivo.

Índice general

1. INTRODUCCIÓN

La estimación de la veracidad de expresiones faciales en video (determinar si el gesto ejecutado por un sujeto es sincero o fingido, en el sentido de si su reproducción se corresponde con lo que el sujeto siente) es un tema que ha comenzado a tratar la comunidad de visión por computadora. Previamente se han reportado múltiples investigaciones basadas en reconocimiento y clasificación de características extraídas de la cara: estimación de edad, género, identidad, tipo de expresión, etc.

Entre las múltiples aplicaciones que tiene este problema están mejorar la interacción humano-computadora, aumentar la efectividad de los robots asistentes, aportar en el tratamiento de desórdenes cognitivos crónicos [17], asistir investigaciones policiales [16], entre otros. También podría ser utilizado como herramienta para inferir qué tan bueno es un actor o para juzgar si un sospechoso dice la verdad [15].

La temática adquirió cierta relevancia a partir del desafío propuesto en el ICCV 2017[1], que puso a disposición un set de videos etiquetados, con más datos de los disponibles hasta el momento. Estos videos muestran sujetos realizando diferentes expresiones, siguiendo una de las tantas clasificaciones propuestas por Paul Ekman [8, 9].

El presente trabajo surgió a partir del desafío ICCV2017 en el que se propuso desarrollar un clasificador de videos utilizando un set de datos predefinido como base de entrenamiento. Todos los videos provistos para la competencia vienen acompañados de una etiqueta que indica el tipo de expresión, limitando el desafío a clasificar únicamente la veracidad de la expresión en el video.

El objetivo del presente trabajo es estudiar distintas técnicas o enfoques con el fín de diseñar un sistema basado en Aprendizaje Automático para clasificar expresiones faciales humanas en videos según su veracidad. En los últimos años, el éxito de las técnicas de aprendizaje automático (sobre todo basado en redes neuronales) ha acelerado el progreso y la investigación sobre el entendimiento de videos [19].

La propuesta está basada en utilizar distintas redes neuronales artificiales, variando hiperparámetros y comparando la performance de cada variante implementada. En particular, se plantea el uso de redes neuronales recurrentes que se utilizan cuando las instancias a clasificar son a su vez, una secuencia de subinstancias.

Otro aspecto a considerar para desarrollar el sistema es la extracción de atributos de las instancias de entrada. Se consideran varios métodos de manipulación de imágenes y, particularmente, de caras humanas. Se trabaja por ejemplo identificando los landmarks de una cara y haciendo un seguimiento de sus trayectorias a lo largo del video.

También se propone una métrica para rankear las redes implementadas en función de qué tan buenas resultaron clasificando y qué tan confiables son sus predicciones. Si bien

[1] https://competitions.codalab.org/competitions/16611

existen varias métricas estandarizadas para evaluar la performance de los clasificadores basados en inteligencia artificial (como la *accuracy*, la *precision* o el *recall*[2]), estas son poco representativas cuando el set de datos es tan reducido como en el desafío en cuestión. La métrica propuesta se utilizará para generar un clasificador final que combine lo aprendido por cada una de las redes implementadas.

Otros participantes de la competencia mencionada alcanzaron 76 % de *accuracy* sobre la base de validación y 66.7 % sobre la base de evaluación utilizando técnicas similares de Aprendizaje Automático. El presente trabajo alcanzó, en una de sus variantes iniciales, 68 % de *accuracy* sobre la base de validación. Lamentablemente, la solución no llegó a implementarse a tiempo y no se pudo submitiar antes de que cerrara el período de la competencia. Sumado a esto las etiquetas para la base de evaluación nunca se publicaron con lo cual la performance del método propuesto no pudo ser evaluada sobre dicha base.

Este trabajo se organiza de la siguiente manera. En la sección 2 se detallan las investigaciones que se fueron realizando en el marco del estudio de las expresiones faciales y su implicancia en las formas de comunicación humanas. En la sección 3 se describe el origen del problema y los detalles del desafío que motivó la realización de este trabajo. En la sección 4 se introducen los conceptos de aprendizaje automático aplicados para el desarrollo del clasificador y luego se detallan las etapas del proceso. En la sección 5 se describen los experimentos realizados para poner a prueba la eficacia del clasificador propuesto y comparar distintas variantes del mismo. En la sección 6 se muestran y analizan los resultados de los experimentos descriptos en la sección anterior.

[2] Se utilizan los nombres de las métricas en inglés para evitar ambigüedades.

2. TRABAJOS PREVIOS

2.1. Clasificación de Expresiones Faciales

Las expresiones faciales son una forma de expresar emociones a través de los músculos faciales. Las personas utilizan expresiones faciales constantemente para comunicarse y, en general, son muy buenas interpretando el significado de una expresión facial en otra persona. El psicólogo Paul Ekman realizó varios estudios intentando clasificar las expresiones faciales en distintas categorías [8, 9]. En 1972 propuso ¡la clasificación en¿que existen 6 expresiones básicas: alegría, enojo, miedo, disgusto, sorpresa y tristeza. Más adelante, en 1999 extendió esta clasificación a un total de 11, incluyendo algunas que no están codificadas con músculos faciales [10]. Si bien, no hay una clasificación oficial vigente hoy en día pero suelen utilizarse variaciones de la clasificación original de 6 expresiones básicas.

La clasificación automática de expresiones faciales en imágenes ha sido un campo de estudio activo desde fines de los años 70, pero las limitaciones en el poder de cómputo y la falta de datos hizo que se estancara [16]. Las nuevas tecnologías y la generación de nuevos sets de datos llevaron a que se reactive la investigación a principios de los años 2000s. Desde entonces se han utilizado numerosas estrategias para atacar el problema [5], tanto en la selección de atributos (análisis de la imagen, extracción manual, generación automática, análisis de microexpresiones, etc) como en la forma de clasificarlos (regresión, SVM, redes neuronales, etc).

Este problema ya ha sido ampliamente abordado y ha sido resuelto con altos niveles de precisión utilizando diferentes técnicas, siendo las redes neuronales convolucionales el modelo predominante.

Un ejemplo de ello es un equipo de la universidad de Standford [1] que entrenó múltiples redes neuronales convolucionales con distintos niveles de profundidad para clasificar un set de datos de Kaggle[1] en una de las siguientes 7 expresiones faciales: enojo, disgusto, miedo, alegría, tristeza, sorpresa o neutral. La base consiste en casi 36.000 imágenes de 48x48 píxeles en escala de grises. Para los atributos se combinaron los generados por la red con atributos de tipo HOG (Histogram of Oriented Gradients) [6]. Sin embargo, llegaron a la conclusión de que las redes profundas por sí solas pueden deducir la información provista por los atributos HOG ya que los resultados entre usarlos o usar sólo los atributos extraídos por las redes fueron considerablemente similares.

Otro trabajo, también de la universidad de Stanford [20] compara dos arquitecturas de redes neuronales convolucionales (VGG-16 y ResNet50) y un modelo SVM sobre dos sets de datos. Además de usar el set de Kaggle mencionado en el ejemplo anterior, utilizaron el set *Karolinska Directed Emotional Faces*[2] (KDEF), un set de 4900 imágenes a color de 562x762 píxeles. En ambos casos, las redes convolucionales superaron ampliamente al

[1] https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

[2] https://www.emotionlab.se/kdef/

modelo SVM.

El problema de clasificar expresiones en imágenes inevitablemente derivó en la clasificación dinámica sobre secuencias de imágenes, agregando la dimensión temporal al problema. Además de las estrategias que ya se venían utilizando para resolver el problema en imágenes, también surgieron nuevas alternativas como modelos ocultos de Markov y Redes Neuronales Recurrentes (o RNN, por sus iniciales en inglés, Recurrent Neural Networks).

2.2. Veracidad de una Expresión Facial

Cerrado ya el tema de clasificación del tipo de expresión, surge un nuevo problema. Dada una expresión, cuyo tipo ya se conoce, determinar si esta es sincera o fingida. Este problema se remonta al siglo XIX, cuando el médico e investigador Duchenne de Boulogne estudiaba los efectos de la estimulación eléctrica sobre los músculos faciales y descubrió que ciertos músculos alrededor de los ojos no pueden contraerse voluntariamente y sólo se contraen al realizar una sonrisa espontánea [7].

Este fenómeno plantea la pregunta de si toda expresión tiene algún rasgo característico que permite determinar su veracidad. Más allá del desafío computacional, esta es una característica mucho más subjetiva que el tipo de expresión y a diferencia del problema de clasificar una expresión según su tipo, es una tarea difícil incluso para humanos.

Siguiendo el análisis de otro de los equipos que participó de la competencia (NIT-OVGU [21]), los humanos tienen la capacidad de fingir expresiones faciales. La simulación de emociones es una habilidad tan poderosa que la mayoría de los observadores pueden ser fácilmente engañados [2, 3]. Sin embargo, los sistemas de visión por computadora han podido detectar expresiones fingidas en algunos casos particulares. Hoque et al. desarrolló un sistema capaz de distinguir entre una sonrisa *frustrada* y una *encantada*, una tarea que los humanos realizan mucho peor [14]. Para lograrlo, entrenó un modelo SVM a partir de un set de datos propio. En dicho set, se grabaron videos de sujetos realizando expresiones tras ser sometidos a experiencias agradables y frustrantes. El clasificador basado en SVM alcanzó 92 % de *accuracy* mientras que la clasificación realizada por humanos apenas superó el 50 %.

Por otro lado, Littlewort et al. y Bartlett et al. presentaron enfoques que clasifican expresiones de dolor según su veracidad basándose en la dinámica de las *action units*.

El primero (Littlewort et al.) desarrolló un clasificador para distinguir expresiones de dolor auténticas y fingidas. El clasificador obtiene 20 descriptores de acciones faciales a partir de un video de entrada y luego clasifica la instancia mediante un modelo SVM. El modelo fue entrenado con videos de 26 sujetos. A cada sujeto se lo filmó realizando una expresión de dolor auténtica, una expresión de dolor fingida y una expresión neutral. El modelo clasificó correctamente el 88 % de las instancias de evaluación mientras que los humanos sólo alcanzaron 49 % de *accuracy* [18].

El segundo comienza entrenando humanos para diferenciar expresiones de dolor auténticas y fingidas y la única mejora que consigue es de 50 % a 55 % de *accuracy*, mientras que

un modelo basado en aprendizaje automático llega fácil a 85 %. En este caso, el clasificador mide los movimientos faciales y obtiene las acciones faciales a lo largo del tiempo. Luego determina la veracidad en base a reconocimiento de patrones usando un modelo SVM. Los videos utilizados corresponden a 24 pares de videos (uno real y uno fingido), uno por cada sujeto [3].

En el presente trabajo se explora la capacidad de un sistema para identificar expresiones faciales verdaderas y falsas en video. Utilizar videos en lugar de imágenes como entrada también plantea una dificultad adicional. Sobre clasificación de imágenes hay mucho trabajo realizado. Las redes neuronales convolucionales se desarrollaron específicamente para clasificar imágenes, ya que analizan la estructura espacial de los atributos de entrada (es decir, los pixeles de la imagen). Se han realizado muchos trabajos utilizando este tipo de redes para clasificar imágenes de distintos dominios (Shima et al. en particular para clasificar expresiones faciales [1]), la mayoría obteniendo resultados muy alentadores.

Clasificar un video agrega el factor temporal. Xu et al. propone soluciones interesantes para la representación de video previo a la clasificación, capturando exitosamente el factor temporal [23]. Su trabajo consistió en desarrollar un modelo para detectar eventos en videos. Para ello extrajeron descriptores individuales de cada cuadro del video utilizando una red convolucional y luego hicieron pooling de los conjuntos de atributos de cada cuadro. Para este último paso utilizaron la técnica Vector of Locally Aggregated Descriptors (VLAD), basada en modelos de mixturas de gaussianas (o GMM, por sus iniciales en inglés, Gaussian Mixture Model).

En el presente trabajo se busca capturar ese factor utilizando redes neuronales recurrentes. Estas redes parecen ser indicadas para resolver este tipo de problemas debido a que pueden capturar la estructura temporal de una secuencia de datos (cada cuadro del video).

3. DESCRIPCIÓN DEL DESAFÍO DE DISCRIMINACIÓN ENTRE EXPRESIONES SINCERAS Y FINGIDAS

Este trabajo surge a partir de un desafío de la fundación ICCV[1], que propone desarrollar un sistema capaz de determinar la veracidad de una expresión facial grabada en un video. Los detalles de la competencia pueden encontrarse en la página de CodaLab[2].

3.1. Detalles de la competencia

Se parte de un dataset que contiene videos de distintas duraciones, en los que se ven a personas realizando distintas expresiones faciales. Este dataset fue generada con el propósito de evaluar la hipótesis de que un clasificador automático podría detectar expresiones falsas basándose en la detección y clasificación de microexpresiones [16].

Los videos tienen resolución espacial 1280x960 píxeles y resolución temporal 100 cuadros por segundo. Cada video tiene dos etiquetas; una correspondiente al tipo de expresión y otra que indica si la expresión es verdadera o falsa.

El tipo de expresión puede ser HAPPINESS, SADNESS, DISGUST, ANGER, CONTENTMENT o SURPRISE. La base de entrenamiento (Fig. 3.1) incluye 40 sujetos y 12 videos por sujeto, uno por cada combinación entre los 6 tipos de expresiones y los dos tipos de valor de verdad, lo que da un total de 480 videos.

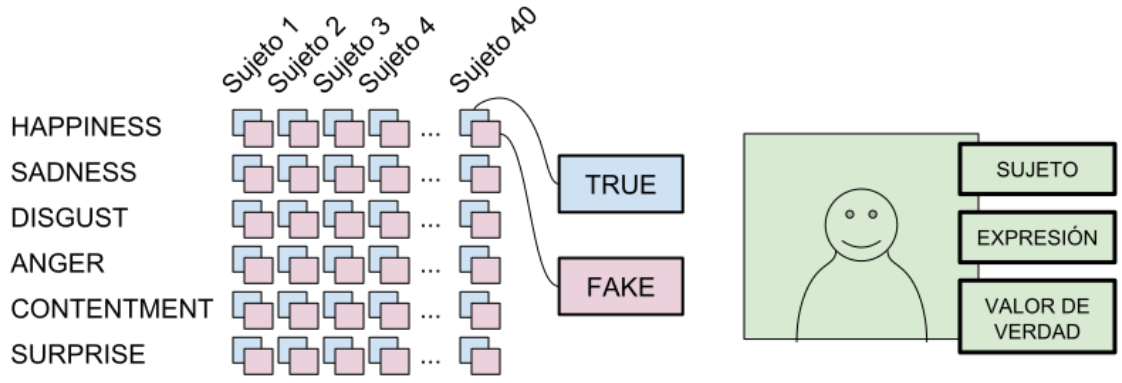

Fig. 3.1: Diagrama de la base de entrenamiento de la competencia. Por cada uno de los 40 sujetos se tienen 12 videos; uno verdadero y uno falso por cada una de las 6 posibles expresiones faciales. Cada videos viene acompañado de 3 etiquetas; un identificador de sujeto, el tipo de expresión y el valor de verdad.

Durante la etapa de aprendizaje del desafío se evaluó la *accuracy* de los competidores clasificando instancias en una base de validación. Como sucede con la base de entrenamiento, la base de validación (Fig. 3.2) consta de un video por cada combinación entre una de las 6 expresiones, uno de los dos valores de verdad y cada uno de los 5 nuevos sujetos

[1] http://iccv2017.thecvf.com/
[2] https://competitions.codalab.org/competitions/16611

que se agregan, dando un total de 60 nuevos videos. A diferencia de los videos en la base de entrenamiento, los videos en la base de validación sólo tienen incorporada la etiqueta correspondiente a la expresión.

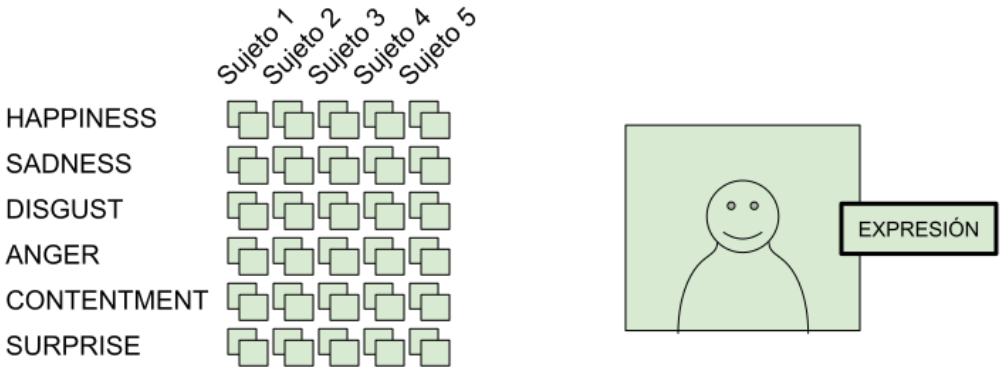

Fig. 3.2: Diagrama de las bases de validación y de evaluación de la competencia. Cada una incorporó 5 nuevos sujetos. Por cada uno de los 5 sujetos se tienen 12 videos; uno verdadero y uno falso por cada una de las 6 posibles expresiones faciales. En este caso, cada video contiene una única etiqueta que indica el tipo de expresión. Una vez finalizada la etapa de validación se publicaron las etiquetas del valor de verdad para los videos de la base de validación.

Al finalizar la etapa de aprendizaje, se publicaron las etiquetas correspondientes al valor de verdad de los videos en la base de validación y se presentó una base de evaluación, con las mismas características que inicialmente tenía la base de validación (Fig. 3.2): 60 videos correspondientes a cada combinación entre sujeto (de entre 5 nuevos sujetos), expresión y valor de verdad, etiquetados únicamente con la expresión. Las etiquetas correspondientes al valor de verdad de las instancias en la base de evaluación no están disponibles.

3.2. Resultados reportados

A continuación se detallan las estrategias adoptadas por los 3 equipos que consiguieron la calificación más alta en el desafío.

El equipo NIT-OVGU [21], conformado por el grupo de investigación Neuro-Information Technology[3] de la Universidad de Magdeburgo, Alemania[4] consiguió 76 % de *accuracy* en la etapa de validación y 66.7 % en la etapa de evaluación. El método utilizado (resumido en la Fig. 3.3) consistió en hacer un ranking de veracidad para cada par sujeto-emoción a partir del par de videos (el verdadero y el falso) de dicho sujeto para dicha expresión. Sobre cada uno de los dos videos se estiman los Action Unit Intensities (AUI) y los Facial Activity Descriptors (FAD). Luego se utilizan tales atributos para entrenar un clasificador SVM que genera un ranking de veracidad. Finalmente, el sistema puede predecir, a partir de dos videos del mismo sujeto y misma expresión, cuál de ellos es el verdadero y cuál es el falso. El código está en un repositorio público[5].

[3] http://www.iikt.ovgu.de/nit.html
[4] http://www.ovgu.de/
[5] https://github.com/fsaxen/NIT-ICCV17Challenge

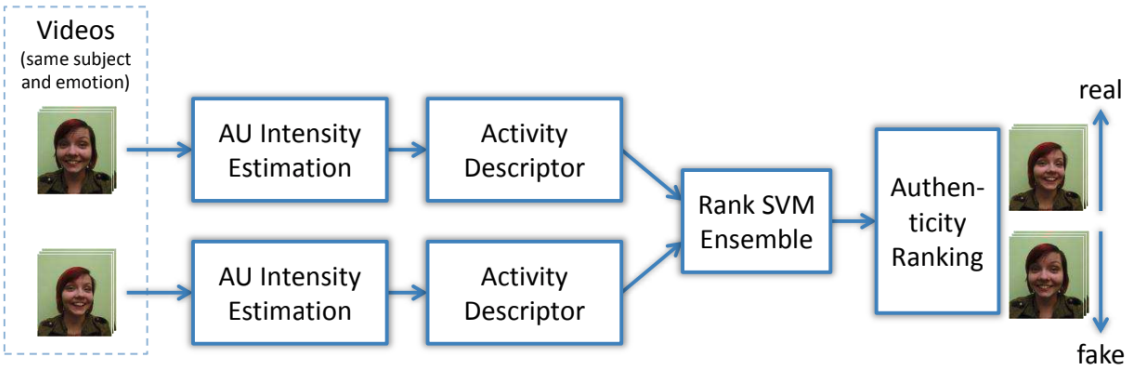

Fig. 3.3: Diagrama del proceso de clasificación del equipo NIT-OVGU. Parten del par de videos (el verdadero y el falso) para un determinado par sujeto-expresión. A cada video se le calculan los AUI y los FAD. Con esos atributos, un clasificador SVM determina cuál es el verdadero y cuál es el falso

Este equipo destacó que la performance humana para realizar la clasificación es apenas superior al azar, incluso cuando se les muestra ambos videos y se les pide elegir cuál es el verdadero y cuál es el falso.

El equipo HCI-Lab [15], conformado por el grupo de investigación Human-Computer Interaction de la Universidad Sejong en Corea del Sur[6] consiguió 71 % de *accuracy* en la etapa de validación y 66.7 % en la etapa de evaluación. El método utilizado (resumido en la Fig. 3.4) consistió en entrenar una red neuronal recurrente utilizando como atributos los landmarks faciales de cada frame. Sobre esa base, se agregó una capa de Bias Paramétrico y un discriminador Gradient Boosting.

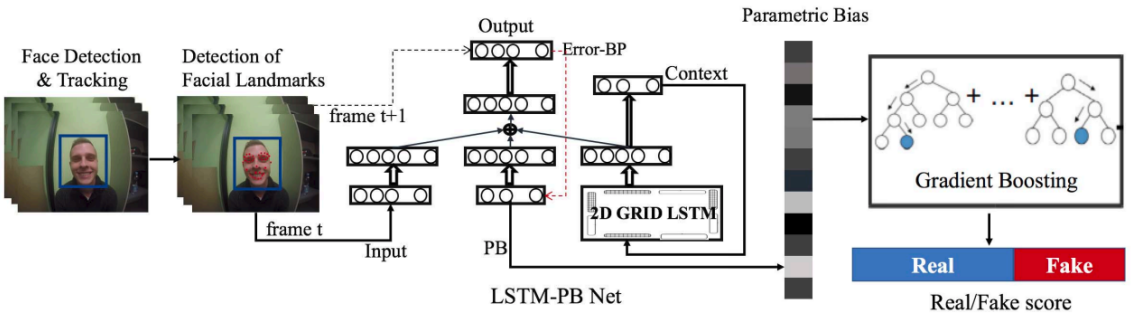

Fig. 3.4: Diagrama del proceso de clasificación del equipo HCI-Lab. Se entrena una LSTM con los landmarks de cada frame. Luego se aplica un Bias Paramétrico y un Gradient Boosting.

El equipo TUBITAK UZAY-METU [19], conformado por el grupo de investigación de procesamiento de imágenes del Instituto de Investigación de Tecnologías Espaciales TÜBİTAK[7] y por el laboratorio de datos multimedia de la Universidad Técnica de Medio Oriente[8] de Turquía, consiguió 61 % de *accuracy* en la etapa de validación y 65 % en la

[6] http://ce.sejong.ac.kr/
[7] http://uzay.tubitak.gov.tr
[8] http://www.metu.edu.tr/

etapa de evaluación. Si bien en ambas etapas quedó en tercer lugar, es destacable que fue el único cuyo valor de *accuracy* aumentó en la segunda etapa respecto a la primera. El método utilizado (resumido en la Fig. 3.5) consistió en extraer representaciones visuales de micro-emociones para cada tipo de emoción y "agregarlas" en el tiempo para construir una única representación de cada video que capture la estructura temporal a corto plazo entre los atributos a nivel frame y sus interdependencias espaciales en la representación. Para esto, se computan atributos emocionales de alto nivel (conv5) sobre cada frame. Estos atributos se normalizan restando el promedio de cada sujeto para minimizar el overfit. Luego se utiliza una red neuronal recurrente sobre la secuencia de atributos "agregados" de a 3 para generar la representación del video. Finalmente se utiliza esa representación para entrenar un clasificador SVM.

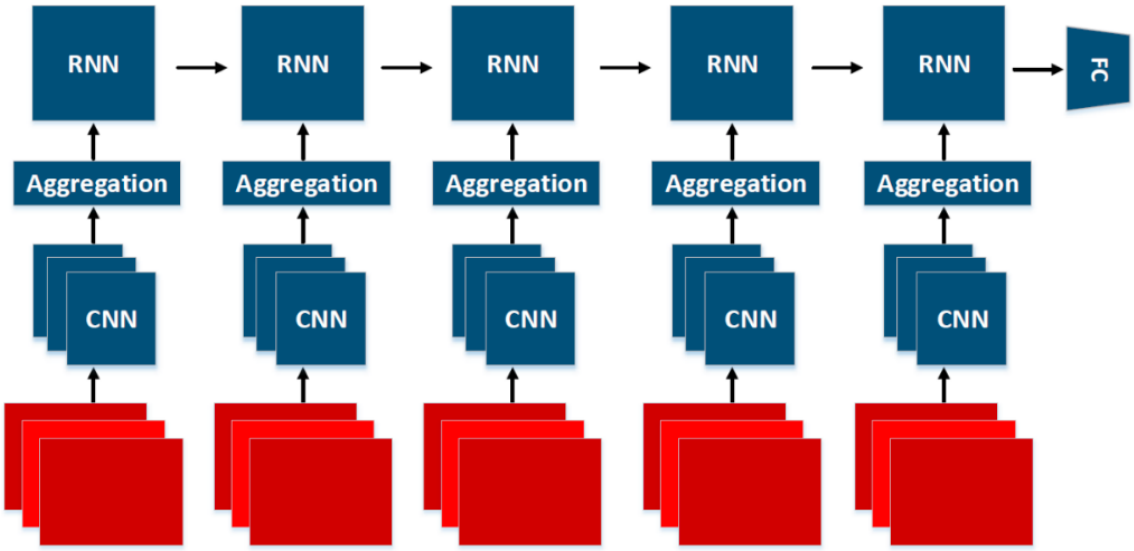

Fig. 3.5: Diagrama del proceso de clasificación del equipo TUBITAK UZAY-METU.

A continuación se muestra una tabla con el resumen de las técnicas utilizadas y los resultados obtenidos por cada equipo.

Equipo	Atributos	Clasificación	Validación	Evaluación
NIT-OVGU	AUI+FAD	SVM (por pares)	76 %	66.7 %
HCI-Lab	Landmarks	LSTM-PB	71 %	66.7 %
TUBITAK UZAY-METU	Atributos Emocionales[9]	SVM	61 %	65 %

Más detalles sobre los concursantes y sobre el desafío pueden encontrarse en el resumen de la competencia publicado por ICCV [22].

[9] Representación del video a partir de entrenar una RNN con la salida de una Conv5 sobre cada cuadro

4. METODOLOGÍA

El sistema que se busca implementar debe recibir como entrada uno de estos videos y clasificarlo según si la expresión presentada en el video es verdadera o falsa. Para lograr tal objetivo, se dividió el procedimiento de clasificación en dos etapas. La primera consiste en tomar el archivo de video en formato mp4 y generar una matriz que contenga información pertinente de cada frame del video. A esta etapa del procedimiento se la llamará Extracción de Atributos. La segunda etapa consiste en tomar la matriz generada en el paso anterior y realizar la clasificación. A esta etapa del procedimiento se la llamará Clasificación.

4.1. Aprendizaje Automático

Como se mencionó en la introducción, el enfoque elegido abarca técnicas de Aprendizaje Automático. Las técnicas de Aprendizaje Automático (o Machine Learning) están ganando cada vez más fuerza y son cada vez más utilizadas debido a su asombroso poder y facilidad de programación. En los pocos años desde que se empezó a investigar, la cantidad de tareas que pudieron resolverse satisfactoriamente con técnicas de Aprendizaje Automático ha crecido exponencialmente.

Estas técnicas engloban un nuevo paradigma de programación. En contraposición a la programación clásica, en la que el programador sabe resolver el problema y le da a la computadora instrucciones específicas para hacerlo de la misma forma, el Aprendizaje Automático consiste en programar un sistema que aprenda por sí mismo a resolver el problema. Para ello, el sistema debe ser entrenado con un conjunto de instancias etiquetadas, a partir del cual pueda encontrar patrones sobre cómo resolver el problema. Si el entrenamiento es adecuado, el sistema debería poder predecir el resultado de una nueva instancia sin etiquetar, a partir de extrapolar el conocimiento obtenido durante su entrenamiento.

4.1.1. Redes Neuronales

Las Redes Neuronales Artificiales pertenecen a la rama más reciente de Aprendizaje Automático y es también la que más rápido avanza y que mejores resultados genera. Surgen de modelar las redes neuronales naturales, como una red de clasificadores simples que, en conjunto conforman un clasificador complejo capaz de realizar tareas con precisión cercana (o a veces incluso superior) a la humana.

El componente básico es el perceptrón (Fig. 4.1), que modela una neurona. Este componente toma un valor como entrada y le aplica una transformación para generar la salida. Esta transformación no es más que alguna función matemática con determinados coeficientes. El entrenamiento de un perceptrón consiste en ajustar dichos coeficientes. Para ello, se toman distintos valores de entrada cuyos resultados son conocidos y se compara la salida generada por el perceptrón con el valor que debería generar. En base a la diferencia entre dichos valores, se modifican los coeficientes de la función del perceptrón para ajustar la función y que el resultado generado por el perceptrón para esa entrada sea más parecido al valor real.

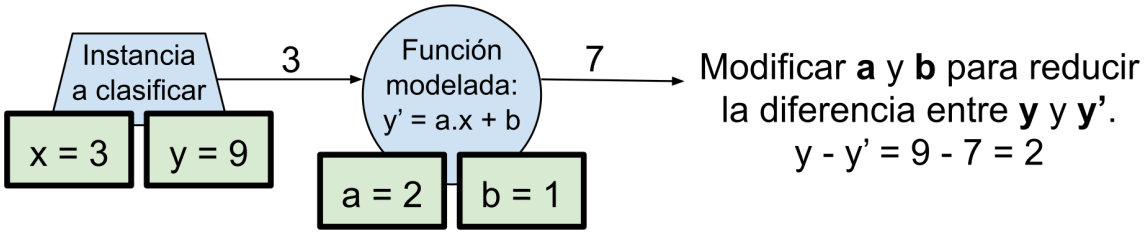

Fig. 4.1: Ejemplo (simplificación) de funcionamiento de un perceptrón. Este modela una determinada función con ciertos parámetros variables. Una instancia a clasificar se representa como un vector (en el ejemplo, de tamaño 1) y la salida del perceptrón es simplemente aplicarle la función a la entrada. Una instancia puede venir etiquetada. En ese caso, el error se calcula como la diferencia entre la etiqueta ($y = 9$) y el resultado del perceptrón ($y' = 7$). Entrenar un perceptrón consiste en ir modificando los valores de sus parámetros (a y b) para ajustar mejor a la mayor cantidad de entradas posibles.

Un perceptrón simple no puede aprender más que una función lineal. Sin embargo, al combinar cientos o hasta miles de perceptrones simples, la red neuronal resultante puede aprender funciones tan complejas como el problema requiera.

La Red Neuronal Artificial (Fig. 4.2) se divide en capas. Cada capa contiene alguna cantidad de perceptrones. En general, cada perceptrón se conecta a cada uno de los perceptrones de la capa siguiente. La red neuronal más simple requiere al menos dos capas, una de entrada (input layer) y una de salida (output layer). Para modelar un problema de clasificación, se ingresa cada atributo de una instancia como entrada a un perceptrón en la capa de entrada. El tamaño de la capa de entrada determina el tamaño de las instancias que la red debe clasificar (la cantidad de atributos). La clasificación consiste en asignarle un peso a cada clase posible utilizando los valores generados por la capa de salida. Este peso representa qué tan segura está la red que la instancia ingresada debe ser asignada a dicha clase, así que para determinar la clase a la que pertenece una instancia se toma la clase con mayor peso resultante. El tamaño de la capa de salida determina la cantidad de clases a la que una instancia puede ser asignada.

A cualquier capa entre las capas de entrada y de salida se le llama capa oculta (hidden layer). Una red puede tener cualquier cantidad de capas ocultas. Al incrementar la cantidad y el tamaño (en cantidad de perceptrones) de las capas ocultas, también incrementa la capacidad de aprendizaje de la red. Para realizar una clasificación, la información se propaga a través de la red desde los perceptrones en la capa de entrada hasta los perceptrones en la capa de salida. Cada conexión entre dos perceptrones tiene asignado un peso que determina qué tan importante es esa porción de información para la función que ese perceptrón en particular debe aprender. También tienen asignado un umbral de activación (bias) que determina si la información recibida debe ser propagada a los perceptrones de la siguiente capa. Para que una red realice correctamente la tarea para la cual fue programada, debe ser entrenada. En el caso de las redes neuronales, el entrenamiento consiste en determinar los valores de los pesos y umbrales de activación de cada conexión. Por lo tanto, una red neuronal puede representarse con dos matrices W y b, conteniendo los valores de los pesos y umbrales de activación, respectivamente. Entrenar la red consiste

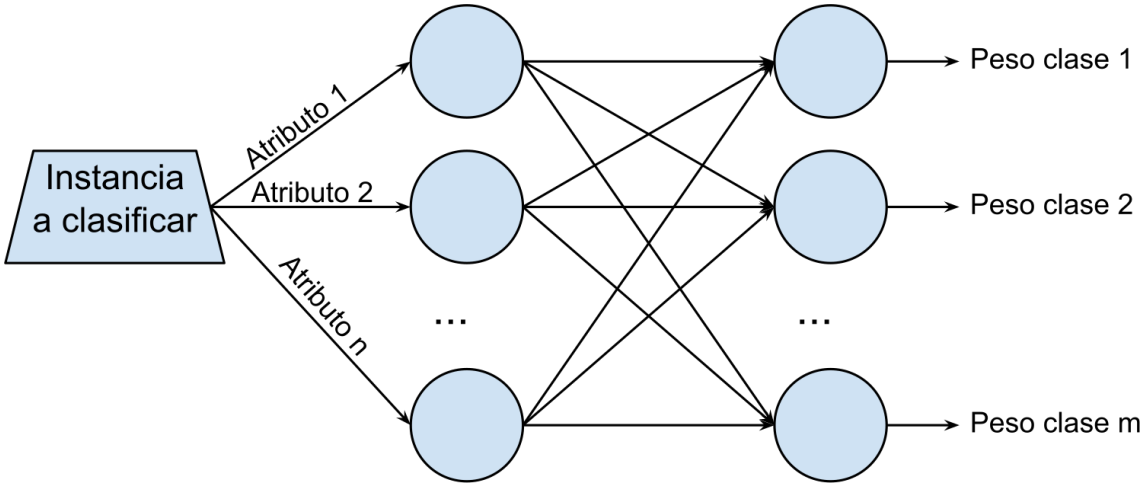

Fig. 4.2: Esquema de una Red Neuronal Clásica (Feed Forward). Los perceptrones se agrupan en capas. La salida de cada perceptrón de una capa i se usa como entrada para cada perceptrón de la capa i+1. La primera capa se llama *capa de entrada* y tiene un perceptrón por cada atributo de una instancia. La última capa se llama *capa de salida* y tiene un perceptrón por cada clase modelada. El resto de las capas se llaman *capas ocultas*.

en someter instancias clasificadas e ir alterando los valores de las matrices para obtener el resultado deseado.

4.1.2. Redes Neuronales Recurrentes

Las Redes Neuronales Recurrentes (Fig. 4.3) son un tipo particular de Redes Neuronales orientadas a clasificar, ya no instancias, sino secuencias de instancias. Fueron introducidas por John J. Hopfield en 1982 [13]. Estas redes vienen a suplir el problema de cómo representar el paso del tiempo al realizar una clasificación con una red neuronal [11].

Lo que distingue a las redes neuronales recurrentes de las redes neuronales tradicionales (feed-forward) es el hecho de que utilizan el resultado de la clasificación anterior como parte de la entrada para clasificar al siguiente elemento en la secuencia. Si la estructura de las redes tradicionales se representa como una secuencia de capas a través de la cual la información se propaga, entonces las redes recurrentes se representan con un una única capa cuya salida está a su vez conectada a su entrada.

A nivel implementación, una Red Neuronal Recurrente requiere, además de las matrices W y b, una tercera matriz que representa el estado de la red. Este estado se modifica tras cada elemento clasificado en la secuencia. De esta forma, el estado de una red recurrente viene a representar una suerte de memoria que le permite a la red utilizar información de clasificaciones pasadas como entrada para clasificar un nuevo elemento en la secuencia.

4.1.3. Redes Neuronales Recurrentes LSTM (Long Short-Term Memory)

Las redes neuronales recurrentes, si bien demostraron captar correctamente el factor temporal de una secuencia de instancias, tienen un problema fundamental que se evidencia al intentar clasificar secuencias largas. Al ir avanzando en la secuencia, la información

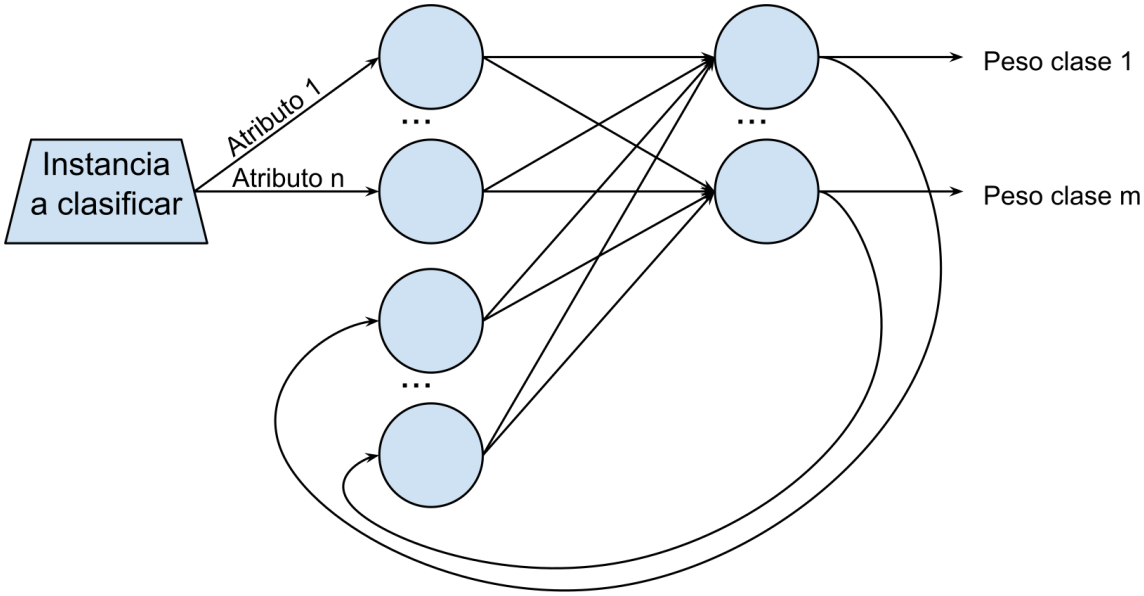

Fig. 4.3: Representación de una Red Neuronal Recurrente. La información generada por la capa de salida en cada paso, es utilizada también por la capa de entrada en el paso siguiente.

provista por las primeras subinstancias empieza a perder relevancia y su influencia sobre el resultado de la siguiente clasificación cae drásticamente. A este problema se le conoce como Vanishing Gradient Problem [4]. Es por esto que se dice que las redes recurrentes tienen memoria de corto plazo. De ahí es que surgen las redes LSTM (Fig. 4.4). Estas redes reemplazan a los perceptrones de las redes recurrentes agregando las unidades denominadas *input gate* y *output gate* con el objetivo de propagar la memoria a más largo plazo. El input gate regula la influencia de la memoria del estado anterior sobre el actual. El output gate regula la influencia de la memoria del estado actual sobre el estado siguiente.

4.2. Preprocesamiento

El primer paso para clasificar un video consiste en discretizarlo como una secuencia de imágenes. El sampleo se realizó a 25 cuadros por segundo. Como los videos no son todos de la misma duración, la cantidad de imágenes obtenidas a partir de cada video no es siempre la misma. Este primer paso convierte cada video en una secuencia de $25*d$ imágenes, siendo d la cantidad de segundos que dura el video. Luego, a cada una de estas imágenes se la recortó alrededor de la cara del sujeto (provocando que las dimensiones de las imágenes también difieran entre videos) y se la pasó a escala de grises.

4.3. Extracción de Atributos

La base de un buen clasificador de Aprendizaje Automático es la extracción de atributos. Los atributos de una instancia son aquellos valores que el clasificador evalúa para determinar a qué clase debe pertenecer una instancia. A diferencia de otros clasificadores de Aprendizaje Automático, para los cuales el programador debe decidir cuáles son los atributos pertinentes para la clasificación, los clasificadores basados en Redes Neurona-

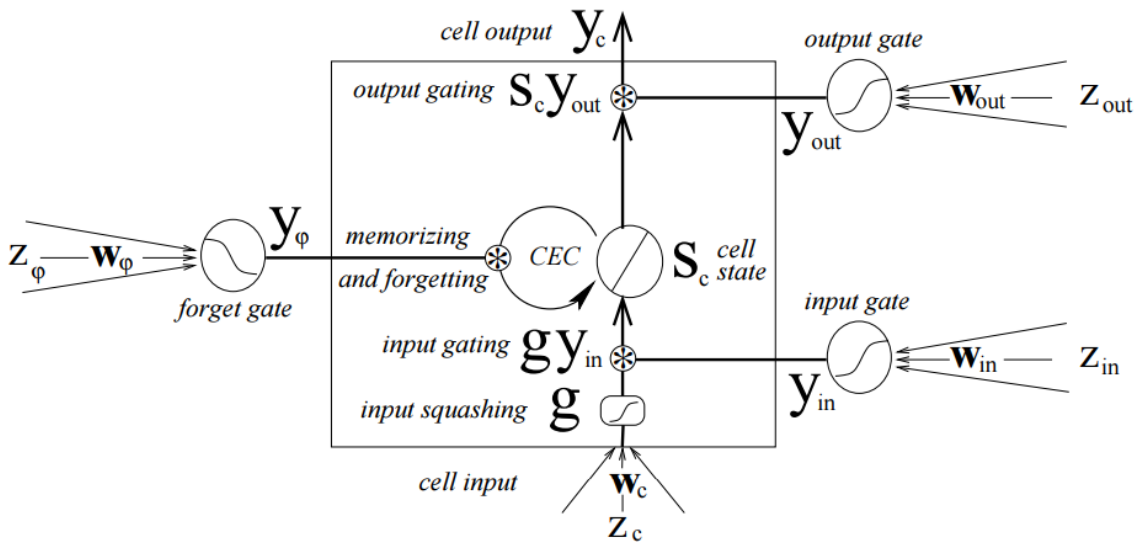

Fig. 4.4: Imagen tomada del paper de Gers et al. en el que se utiliza una LSTM (e incluso introducen una modificación) para resolver un problema en el que era necesario recordar información a lo largo de varias subinstancias de la entrada [12].

les aprenden a extraer automáticamente atributos a partir de las instancias mientras se entrenan (Fig. 4.5). Esta Extracción es en realidad una transformación que se aplica a medida que la información se propaga a través de las capas ocultas de la red. Una vez entrenadas, realizan la misma transformación de una nueva instancia para clasificarla. La capa de salida en una red entrenada utiliza la información generada por la última capa oculta para clasificar la instancia. Esa información (la salida de la última capa oculta) puede pensarse como el vector de atributos pertinentes para la clasificación, dado que una instancia requiere una única capa (la capa de salida) para ser clasificada a partir de estos atributos.

Este procedimiento suele usarse con redes convolucionales para clasificar imágenes, utilizando como atributos originales (los que serán entrada a la red clasificadora) el valor de cada píxel y generando nuevos atributos que, en función de lo que esté clasificando la red, representan cosas más abstractas, generalmente, difíciles de extraer a mano. Estos pueden variar desde la proporción de luz y sombra hasta la cantidad de personas en la imagen.

Para la extracción de atributos se utilizó esta idea (Fig. 4.6). Se utilizó una red neuronal para generar los atributos de cada imagen con el procedimiento descrito anteriormente. Esta red es denominada Red Generadora de Atributos ya que, si bien su propósito es clasificar una imagen en base a los atributos inherentes a la imagen, en este trabajo se utilizará para generar los atributos que luego serán la entrada de la red que clasificará el valor de verdad de un video.

En definitiva, la etapa de Extracción de Atributos consiste en tomar la secuencia de $25*d$ imágenes de cada video y a cada imagen someterla a la red generadora de atributos. De esta forma, cada video de d segundos de duración se convierte en una matriz de $25*d$ filas y A columnas, siendo A la cantidad de atributos que genera la red, es decir, el tamaño

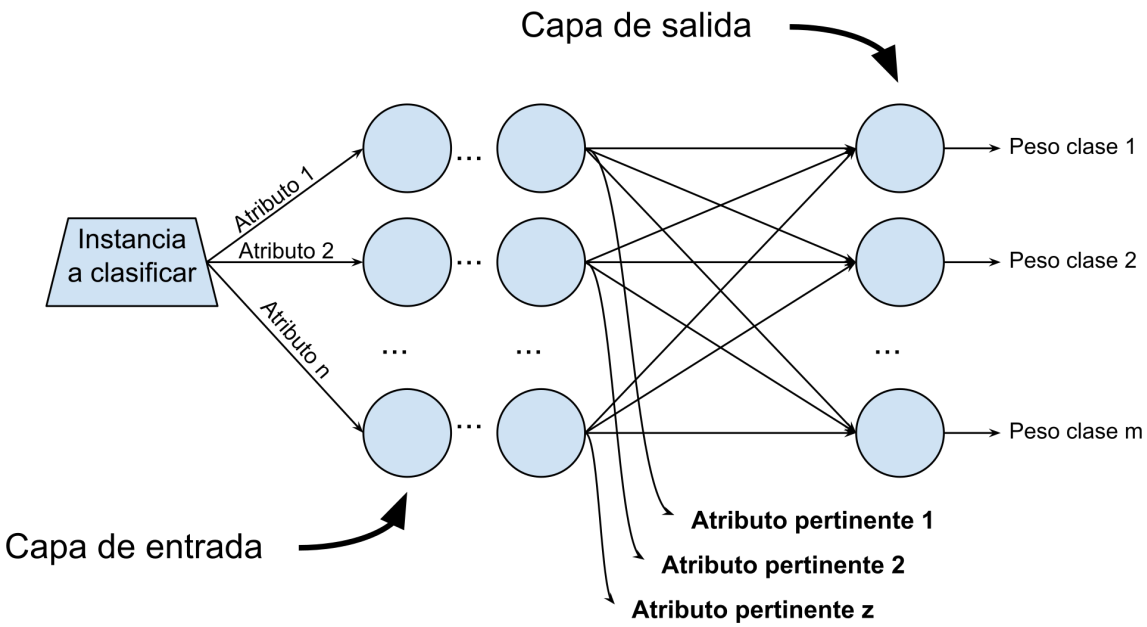

Fig. 4.5: Procedimiento de generación de atributos a partir de una red neuronal. Se parte de una red ya entrenada y se clasifican las instancias de las cuales se quieren extraer atributos pertinentes. El resultado de la clasificación se ignora pero se toman como atributos para representar a la instancia de entrada, los valores de salida de los perceptrones de la última capa oculta.

de la última capa oculta de dicha red.

En particular, se experimentó con distintas redes generadoras de atributos.

La primera es una red neuronal convolucional entrenada para clasificar sujetos de una base de datos según su identidad. La última capa oculta de esta red es una capa de tamaño 4096. Entonces, al quitar la capa de salida en esta red, se obtiene un programa que dada una imagen genera un vector de 4096 atributos optimizados, en principio para determinar la identidad del sujeto en la imagen. Al set de atributos generado por esta red se le asigna el código **ID**.

A continuación, se utilizaron otras 3 redes, cada una de ellas entrenada para clasificar el tipo de expresión en una imagen, esperando que los atributos generados por estas redes sean más relevantes a la hora de determinar si una expresión es verdadera o falsa, de lo que serían los atributos generados por la red anterior.

La primera de estas 3 redes se construyó basándose en la estructura descrita en el paper [1]. La red consiste en 4 capas convolucionales de tamaños 64, 128, 512 y 512, seguidas de dos capas densas de tamaños 256 y 512. A cada capa convolucional se le aplica dropout de 0.25, batch normalization y max pooling. A cada capa densa se le aplica batch normalization y dropout de 0.5. La red se entrenó por 35 épocas con la base fer2013 de Kaggle que contiene casi 29000 imágenes de 48x48 en escala de grises clasificadas entre 7 tipos de expresiones. Una vez obtenido el clasificador entrenado, se lo utilizó para clasificar las imágenes de las bases de CodaLab, para lo cual fue necesario escalar dichas imágenes

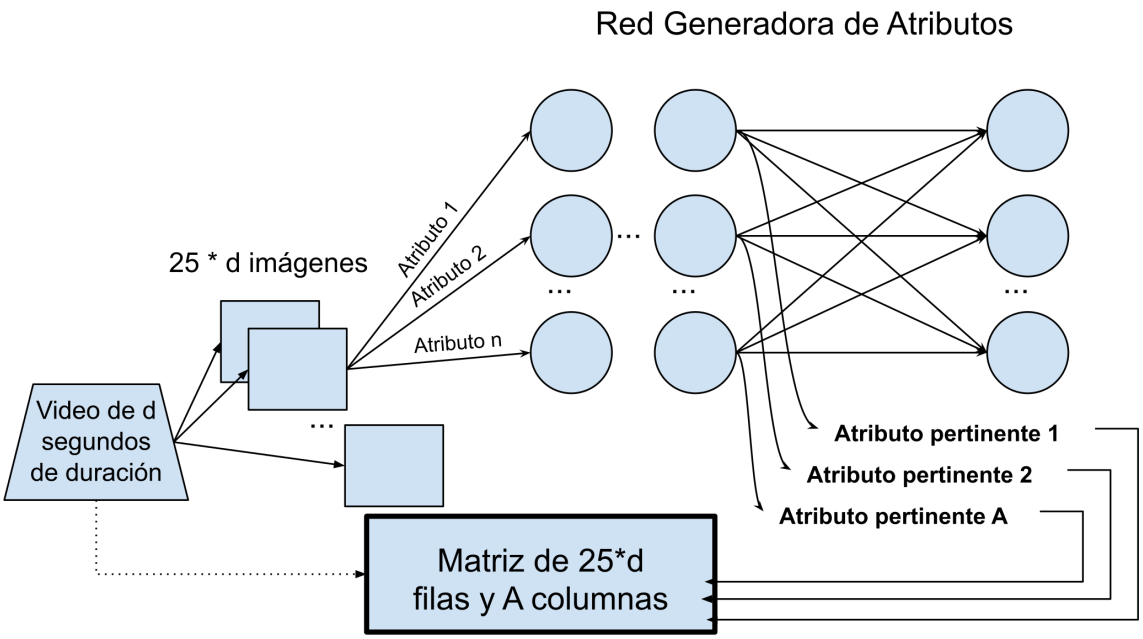

Fig. 4.6: Resumen del procedimiento para generar atributos en el presente trabajo. Primero se separa cada video en una secuencia de imágenes, discretizando a 25 cuadros por segundo. A cada una de estas imáganes se la clasifica con una red generadora de atributos. Los A atributos generados constituyen un vector que representa el cuadro en cuestión. La representación final de cada video consiste en una matriz de 25*d filas y A columnas.

a 48x48. Al set de atributos generado por esta red se le asigna el código **005**.

Como segundo generador de atributos se entrenó otra red convolucional en dos partes. Primero se la entrenó por 10 épocas con una parte del set Cohn Kanade. Esta base consiste en videos en escala de grises de expresiones faciales, con la particularidad de que cada video finaliza en el appex de la expresión correspondiente. Para la primera etapa del entrenamiento de esta red, se utilizaron los últimos 3 frames de cada video que estuviera etiquetado con alguna de las 6 expresiones presentes en la base de CodaLab (es decir, se descartaron las etiquetadas con Fear). Todas las imágenes se recortaron alrededor de la cara y luego se adaptaron al tamaño 256x256. Además, para aumentar la base, se agregó cada imagen blurreada con factor 1 y 2, espejada y rotada 3 grados en cada dirección. Para la segunda etapa se entrenó la red por otras 10 épocas usando los frames de los videos de CodaLab. En este caso, al no conocer de antemano el frame correspondiente al appex de la expresión, hubo que utilizarlos todos. Las imágenes ya estaban en escala de grises y recortadas alrededor de la cara pero eran todas de distintos tamaños, así que hubo que escalarlas a 256x256. La estructura de la red se tomó como la red convolucional más simple, la propuesta en el tutorial de Keras. Esta consiste en 2 bloques convolucionales de dos capas de tamaño 32 cada uno y una capa densa de tamaño 256. A cada capa convolucional se le aplica dropout de 0.25, batch normalization y max pooling. A la capa densa se le aplica dropout de 0.5 y batch normalization. A los sets de atributos generados por esta red se les asignan los códigos **KCK** (al entrenarse sólo con la Cohn Kanade) y **KCKCL** (al entrenarse primero con la Cohn Canade y luego con la CodaLab).

La tercera es una red neuronal que, en lugar de utilizar los valores de los píxeles como entrada, como todas las anteriores, utiliza como entrada las posiciones de los landmarks de Cohn Kanade. Esto requirió obtener los landmarks de cada cuadro. Para tal fin, se utilizó el detector de caras de la biblioteca dlib que, a partir de una imagen, calcula 68 landmarks faciales sobre ella. Como las imágenes son de distintos tamaños y relaciones, se normalizaron las posiciones de los landmarks como la proporción relativa respecto al recuadro de la cara. De esta forma, cada imagen se convierte en un vector de 136 atributos correspondientes a las posiciones relativas x e y de cada landmark. Este vector se utilizó como entrada de una nueva red compuesta por 5 capas densas de tamaños 64, 128, 256, 128 y 64. A cada capa se le aplica dropout de 0.5. Al set de atributos generado por esta red se le asigna el código **LM**.

Como último experimento se utilizó como vector de atributos la concatenación de los atributos generados por las redes **005** y **LM**. A este set de atributos se le asigna el código **LM5**. Se tienen entonces un total de 6 sets de atributos a ser seleccionados para utilizar como entrada en la etapa de clasificación.

4.4. Clasificación

Hasta ahora se describió la primera etapa del procedimiento general: La extracción de atributos. De forma resumida, lo que se hizo fue tomar las instancias de entrada (archivos de video) y convertirlas a matrices de atributos donde cada fila representa un cuadro del video original (Fig. 4.7). Tras haber obtenido las representaciones de estas, se procede a clasificarlas mediante un modelo que tome como entrada dicha representación (en forma de matriz) y prediga la etiqueta más probable para asignarle como valor de verdad (TRUE o FAKE).

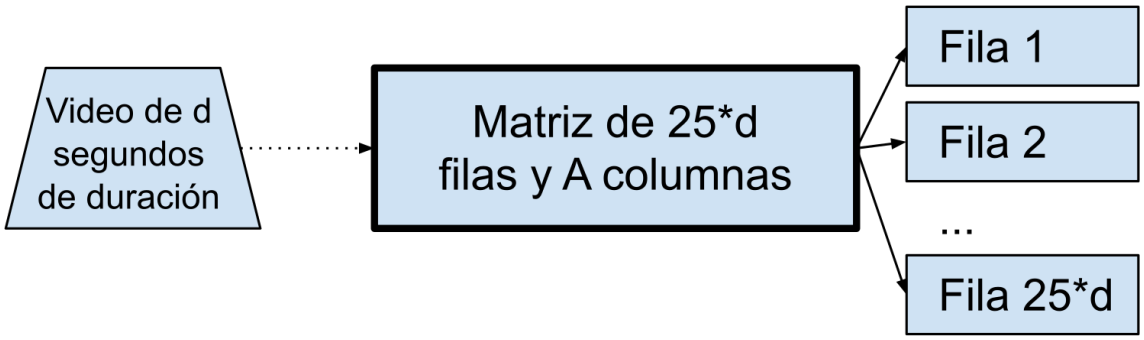

Fig. 4.7: Dada una representación de video como una matriz, la secuencia de subinstancias que se utilizan como entrada para las redes recurrentes consiste en la secuencia generada a partir de tomar cada una de las 25*d filas

Para la segunda etapa del procedimiento se implementaron distintas redes neuronales utilizando la biblioteca Keras de Python. Todas las redes toman como entrada un archivo que contiene la matriz de atributos de cada instancia y devuelven un valor de verdad por cada instancia. Como los videos en la base de validación incluyen la etiqueta del tipo de expresión, cada red se entrenó una vez por cada tipo de expresión para clasificar únicamente los videos correspondientes a ese tipo.

Red Neuronal Recurrente Simple

La primera red implementada es una Red Neuronal Recurrente con dos capas LSTM de tamaño 32. Para entrenar esta red se toma cada matriz y se le pasa a la capa de entrada cada una de las filas de la misma. Como esta red no permite cantidad variable de subinstancias, hubo que normalizar los videos de forma que todas las matrices tengan la misma cantidad de filas. Esta cantidad se convirtió en un hiperparámertro de la red. Al encontrar una matriz con más filas que las requeridas, se descartan las sobrantes de forma que las que queden estén equidistantes. Al encontrar una matriz con menos filas que las requeridas, se replican las que se tienen en igual proporción (dentro de lo posible) hasta que se alcance la cantidad requerida. A esta red se le asigna el código **RS**.

Redes Neuronales Profundas

Con el objetivo de comparar los resultados obtenidos por la red recurrente contra los resultados que se podrían obtener por redes profundas, se implementaron dos Redes Neuronales Profundas. La primera con 5 capas ocultas de tamaños 16, 64, 128, 64 y 16. La segunda con 6 capas ocultas de tamaños 64, 64, 128, 128, 64 y 64. Ambas con Batch Normalization y regularizadores L1 y L2. En estas redes, se tomó como entrada la matriz completa como un vector de atributos de tamaño D^*A, siendo D el mismo hiperparámetro utilizado en la primera red para normalizar la cantidad de filas de cada matriz y A la cantidad de atributos de cada imagen. A estas dos redes se les asignan los códigos **P1** y **P2** respectivamente.

Redes Neuronales Recurrentes Delta

El siguiente paso fue implementar otras dos redes neuronales recurrentes, con la misma estructura que la primera pero incorporando una variación consistente en la diferencia calculada como la resta entre una fila y la anterior, denominada Delta. En lugar de utilizar los valores de cada fila, se utiliza la diferencia de cada fila respecto a la fila anterior. Esto es equivalente a, en lugar de tomar los atributos de cada frame, tomar la diferencia de cada atributo de un frame, respecto al mismo atributo en el frame anterior. La primera de estas dos redes utiliza como atributo la diferencia respecto al anterior. La segunda utiliza como atributo un valor entero en el rango [-1, 1]. Si la diferencia es menor al 5 %, el atributo vale cero. Si no, vale 1 o menos 1 según si la diferencia es positiva o negativa respectivamente. A estas dos redes se les asignan los códigos **RD** y **RDD** respectivamente.

4.5. Implementación

Para la implementación se utilizó python 2 y las bibliotecas `keras`, `tensorflow` y `theano`. El archivo principal es *tesis.py*. Ejecutarlo permite replicar todos los experimentos mencionados anteriormente. Todos los resultados intermedios se guardan en archivos para no tener que ser procesados nuevamente. A continuación se detallan los parámetros disponibles (en todos los casos, respetando mayúsculas).

- **Objetivo**: permite definir qué se va a hacer. Los valores que puede tomar son los siguientes:

  - **MOD**: Crea un modelo de clasificador en base al resto de los parámetros y lo guarda en un archivo.
  - **LOO**: Ejecuta una validación Leave-One-Out para un modelo.
  - **VAL**: Evalúa la clasificación para un modelo a partir de varias métricas.
  - **TF**: Realiza una clasificación con un modelo.

- **Atributos**: código del set de atributos a utilizar para entrenar al modelo. Los valores que puede tomar son los códigos de los sets de atributos definido en la sección 4.3: **ID**, **005**, **KCK**, **KCKCL**, **LM** y **LM5**.

- **Base para entrenar**: base de datos utilizada para entrenar el modelo que se va a utilizar (sólo si el objetivo no es LOO). Los valores que puede tomar son **TRAIN** para la base de entrenamiento, **VALID** para la base de validación, **V501** para la primera mitad de la base de validación y **V502** para la segunda mitad de la base de validación. Notar que no es posible entrenar con la base de evaluación ya que las etiquetas no están disponibles.

- **Base para validar**: base de datos utilizada para validar la clasificación realizada por el modelo que se va a utilizar (sólo si el objetivo es VAL). Los valores que puede tomar son **TRAIN** para la base de entrenamiento, **VALID** para la base de validación, **V501** para la primera mitad de la base de validación y **V502** para la segunda mitad de la base de validación. Notar que no es posible entrenar con la base de evaluación ya que las etiquetas no están disponibles.

- **Base para clasificar**: base de datos que se quiere clasificar con el modelo que se va a utilizar (sólo si el objetivo es TF). Los valores que puede tomar son **TRAIN** para la base de entrenamiento, **VALID** para la base de validación, **V501** para la primera mitad de la base de validación, **V502** para la segunda mitad de la base de validación y **TEST** para la base de evaluación.

- **Modelo del clasificador**: código de la red neuronal que se va a entrenar. Los valores que puede tomar son los códigos de las redes definidas en la sección 4.4: **RS**, **P1**, **P2**, **RD** y **RDD**.

- **ts**: Cantidad de cuadros para normalizar. En los casos de las redes recurrentes (RS, RD y RDD) este parámetro puede valer 0 para no utilizar ninguna normalización de cuadros.

- **epoch**: Cantidad de épocas a entrenar.

Tras parsear los parámetros, se llama a la función correspondiente en base al objetivo. La función *nuevo_clasificador* en el archivo *clasificadores.py* crea un clasificador, lo entrena y lo guarda en un archivo. La función *cargar_matriz_de_atributos* en el archivo *atributos.py* genera la matriz de atributos x y el vector de etiquetas y de una base. La función *clasificacion* en el archivo *clasificacion.py* obtiene un clasificador entrenado llamando a *nuevo_clasificador* y lo utiliza para predecir las etiquetas de otro conjunto de instancias cargadas en una matriz llamando a *cargar_matriz_de_atributos*. La función *validacion* en el archivo *validacion.py* llama a *clasificacion* para clasificar un conjunto de instancias y después compara las predicciones con las etiquetas reales. Los modelos entrenados se

guardan en la carpeta *ModelosEntrenados*. Las clasificaciones realizadas se guardan en la carpeta *Predicciones*. Los resultados de las validaciones se guardan en la carpeta *Resultados*. El archivo *tiempos.json* mantiene la cantidad de tiempo en segundos que le tomó a cada clasificador entrenarse.

5. EXPERIMENTOS

A continuación se presentan los métodos utilizados para evaluar el rendimiento de cada una de las redes implementadas. Los experimentos se ejecutaron en una máquina con 12 procesadores Intel Xeon E5-1650 de 3.60GHz con Ubuntu SMP de 64 bits.

5.1. Leave One Out

Un primer método utilizado para evaluar el rendimiento de los clasificadores es la estrategia llamada leave-one-out. Esta técnica consiste en entrenar el clasificador con todas las instancias de la base de entrenamiento, excepto las correspondientes a un sujeto particular. Luego se utiliza el clasificador entrenado para clasificar las instancias de dicho sujeto. El objetivo es verificar si el clasificador abstrae la información requerida más allá de los sujetos particulares. El procedimiento para evaluar cada una de las redes implementadas, utilizando la estrategia leave-one-out consistió en organizar los videos de la base de entrenamiento según la expresión y el sujeto. Por cada una de las seis expresiones y por cada uno de los cuarenta sujetos se entrenó una red utilizando los videos de la base de entrenamiento correspondientes a la misma expresión pero a distinto sujeto. Por lo tanto, cada red se entrenó con 78 instancias y luego se validó con dos instancias. Por esta razón, el porcentaje de *accuracy* en cada etapa de la validación sólo puede ser 0 %, 50 % o 100 %.

5.2. Validación

Una vez que estuvo disponible la base de validación del desafío, se pudo proceder con el siguiente método de evaluación de los clasificadores. Este consiste en entrenar cada red con todas las instancias de la base de entrenamiento y clasificar los videos de la base de validación. Esta nueva base cuenta con sesenta nuevos videos (diez por cada expresión), así que cada red se entrenó con ochenta instancias y se validó con diez instancias.

5.3. Validación en base a confianza

Al tener tantos hiperparámetros con los cuales experimentar, se pudo generar una gran cantidad de clasificadores. Al observar las mediciones de *accuracy* de cada uno de estos clasificadores sobre la base de validación no se detectaron resultados interesantes. Sin embargo, al discriminar estas mediciones según el tipo de expresión se pudo apreciar que ciertos clasificadores parecían ser considerablemente mejores para clasificar alguna expresión en particular.

Surge entonces la idea de generar un nuevo clasificador que, en base al tipo de expresión del video de entrada, derive la responsabilidad de realizar la clasificación a aquel modelo que haya obtenido el mejor resultado para dicha expresión. Esto requiere que los videos ya tengan la etiqueta del tipo de expresión. En el set de la competencia esto está asegurado pero si no fuese así, podría agregarse una etapa previa para clasificar el video según la expresión.

La hipótesis se basa en que si, por ejemplo, el clasificador A obtuvo 80 % de *accuracy* sobre la expresión 1 y 20 % sobre la expresión 2, pero el clasificador B obtuvo 20 % sobre la expresión 1 y 80 % sobre la expresión 2, ambos clasificadores obtendrían sólo 50 % de *accuracy*. Sin embargo, un nuevo clasificador que utilice al clasificador A para clasificar expresiones de tipo 1 y al B para expresiones de tipo 2 debería poder alcanzar 80 % de *accuracy*.

Esta idea trajo consigo 2 problemas. El primero, si se utiliza la base de validación para obtener la *accuracy* de cada clasificador, entonces serían necesarios nuevos datos para evaluar que la performance de este último método sea mejor. Se esperaría poder utilizar para ello la base de evaluación, pero esto no es posible, ya que las etiquetas correspondientes al valor de verdad de cada instancia (True o Fake) de la base de evaluación nunca fueron publicadas.

El segundo problema consiste en que, al haber sólo 10 instancias de cada expresión en la base de validación, muchas redes empatan en el primer puesto al clasificar dicha base (los valores de *accuracy* posibles son 0.0, 0.1, 0.2, ..., 0.9 y 1.0).

Para solucionar el primer problema se dividió la base de validación en dos bases. La primera se usará para determinar los mejores clasificadores y la segunda para evaluar si la elección fue correcta.

Para solucionar el segundo problema se desarrolló otra métrica basada en la confianza. Un clasificador de k clases en realidad no devuelve la clase a la que corresponde la instancia clasificada, sino que devuelve, para cada una de las k clases, qué tan segura está la red de que dicha instancia pertenece a tal clase.

Por lo tanto, si se tienen 2 redes R1 y R2 que clasificaron correctamente la instancia I1 como "True", inicialmente podría pensar que ambas son igual de buenas clasificando. Sin embargo, al mirar los valores que devuelven las redes, estos podrían ser True 4, Fake 1 para R1 y True 5, Fake 4.9 para R2. Si bien ambas redes clasificaron correctamente a I1, podemos decir que R1 nos da mayor confianza que R2. Para R1, I1 es "muy True" y "muy poco Fake". Si bien para R2, I1 es "más True" que para R1, también es cierto que R2 estuvo muy cerca de confundir a I1 con Fake, ya que para R2 I1 también es "muy Fake". Con este criterio se busca generar un ranking que dificulte el empate entre redes para cada expresión.

Dicho ranking está basado en la seguridad que tiene una red cuando clasifica correctamente una instancia. Llamaremos confianza de una red R para un conjunto de instancias, a la sumatoria de las diferencias entre los niveles de activación para cada clase, para cada instancia y la notaremos C(R). Formalmente, se define de la siguiente manera:

$$C(R) = \sum_{i \in TRUE} (\Phi_R(TRUE, i) - \Phi_R(FAKE, i)) + \sum_{i \in FAKE} (\Phi_R(FAKE, i) - \Phi_R(TRUE, i))$$

Donde TRUE es el conjunto de instancias de la clase TRUE, FAKE es el conjunto de instancias de la clase FAKE y Φ_R(c, i) es el nivel de activación de la red R para la clase c

y la instancia i.

Para cada instancia de TRUE, el nivel de confianza de una red aumenta según qué tan segura está esa red de que la instancia pertenece a la clase TRUE y disminuye según qué tan segura está la red de que la instancia pertenece a la clase FAKE. Si una instancia de TRUE es clasificada como FAKE, el valor de confianza va a ser negativo, ya que el nivel de activación para la clase FAKE va a ser mayor que el de la clase TRUE. El mismo razonamiento aplica a la inversa para las instancias de FAKE.

6. RESULTADOS

En esta sección se discuten los resultados obtenidos por cada clasificador implementado, según cada método de evaluación utilizado.

6.1. Leave One Out

En general, los resultados obtenidos a través del método leave-one-out no fueron muy prometedores (Fig. 6.1). Viendo los valores de *accuracy* obtenidos al estimar la veracidad en los videos de un sujeto de la base de entrenamiento, tras haber entrenado al modelo con el resto de los videos pareciera que no es buena idea generalizar lo aprendido sobre un grupo de sujetos al resto de la población.

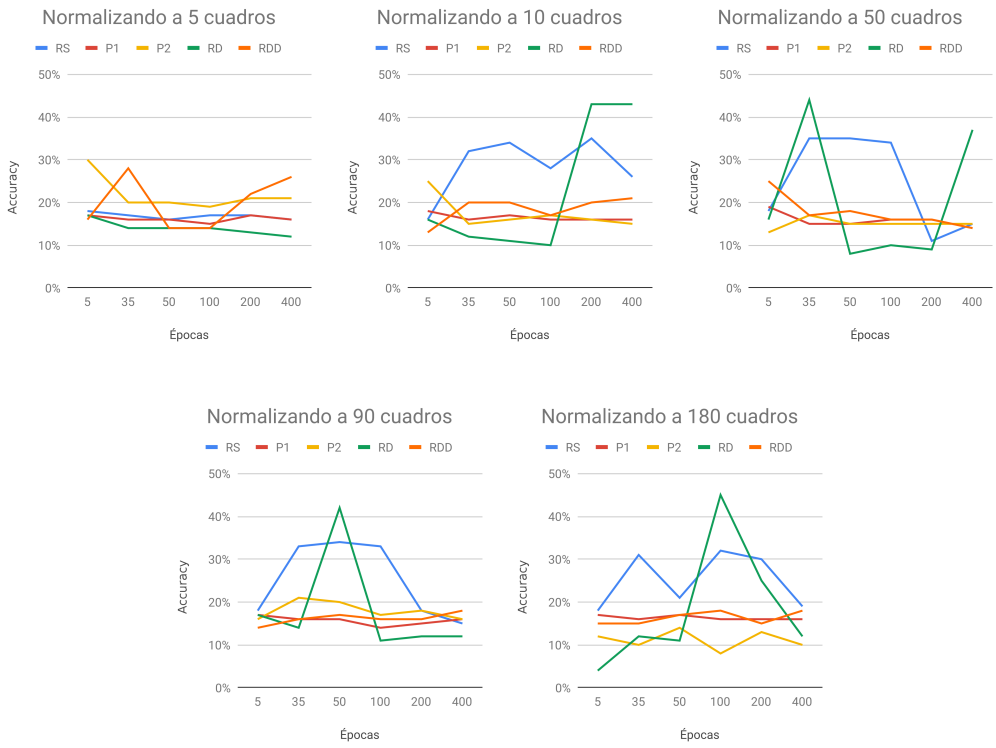

Fig. 6.1: Resultados del experimento Leave One Out. Se muestra la *accuracy* obtenida por cada una de las redes clasificadoras en función de la cantidad de épocas de entrenamiento. En todos los casos, los atributos utilizados fueron los generados con la red ID.

El primer diseño experimental consistió en tomar los atributos de identidad y validar la predicción de cada uno de los sujetos de la base habiendo entrenado al modelo con el resto de los sujetos. Se normalizó la cantidad de cuadros del video de entrada usando 5 valores distintos (5, 10, 50, 90 y 180). La cantidad de épocas se varió entre 5, 35, 50, 100, 200 y 400. En ninguno de los casos se alcanzó 50 % de *accuracy*. Dado los pésimos

resultados y los altos costos de entrenamiento del método Leave-one-out, se decidió no seguir experimentando con los demás tipos de atributos.

6.2. Validación

Al validar los clasificadores contra la base de validación se obtuvieron resultados muy variados en función del tipo de red y de los hiperparámetros utilizados. Incluso, un mismo clasificador entrenado para una determinada expresión obtuvo rendimiento muy distinto cuando se lo entrenó para clasificar otra expresión.

Los equipos ganadores de la competencia, NIT-OVGU y HCILab obtuvieron 76 % y 71 % de *accuracy* respectivamente, en esta etapa. En la etapa final, ambos alcanzaron *accuracy* de 66 %.

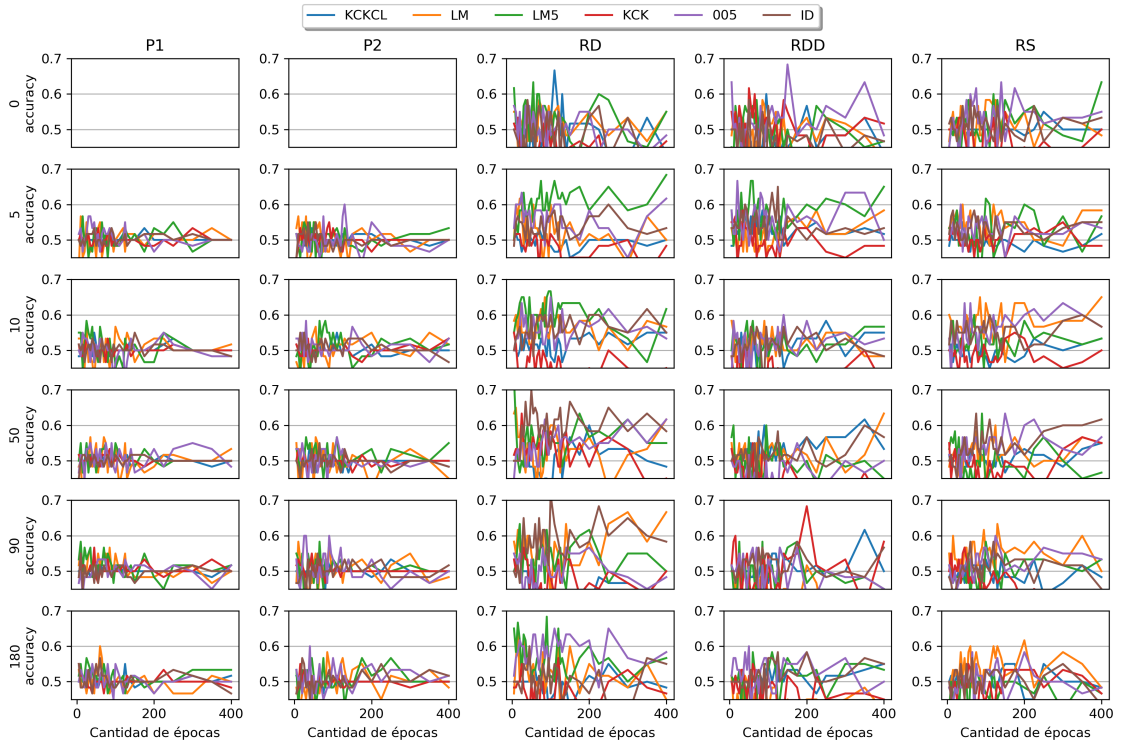

Fig. 6.2: Resultados de la validación (*accuracy* en función de cantidad de épocas de entrenamiento) para cada combinación entre red clasificadora y conjunto de atributos.

En la Fig. 6.2 se muestran los resultados obtenidos para todas las combinaciones entre tipos de atributos y tipos de redes. Además, se compara el resultado en función de la cantidad de cuadros usados para normalizar. Las redes profundas (P1 y P2) no pueden tomar una cantidad variable de cuadros (cuando el parámetro de normalización vale 0) así que esos gráficos están vacíos. A simple vista se observa que las redes profundas tienen un margen mucho más restricto y pocas veces alcanzan el 60 % de *accuracy*. Además, se ve una clara tendencia a acercarse al 50 % al incrementar suficiente la cantidad de épocas de entrenamiento. Las redes recurrentes en cambio, muestran mucha más varianza, y si

bien, también parecen sobreajustar al aumentar la cantidad de épocas de entrenamiento, se observan varias mediciones cercanas al 70 %. La cantidad de cuadros del video tampoco parece ser un factor determinante para mejorar la *accuracy* de un modelo. Esto pareciera contradecir la hipótesis inicial de que una buena resolución temporal permitiría detectar microexpresiones sutiles que definen la veracidad de la expresión en cuestión. La *accuracy* máxima alcanzada fue 73 %. La red que consiguió esa *accuracy* fue la Recurrente Delta (RD) usando los atributos LM5 normalizando 50 cuadros y entrenando 5 épocas.

Todos estos gráficos muestran los resultados de cada red sobre toda la base de validación. Sin embargo, al separar por el tipo de expresión se encontró que una misma red podía performar muy diferente entre dos expresiones distintas.

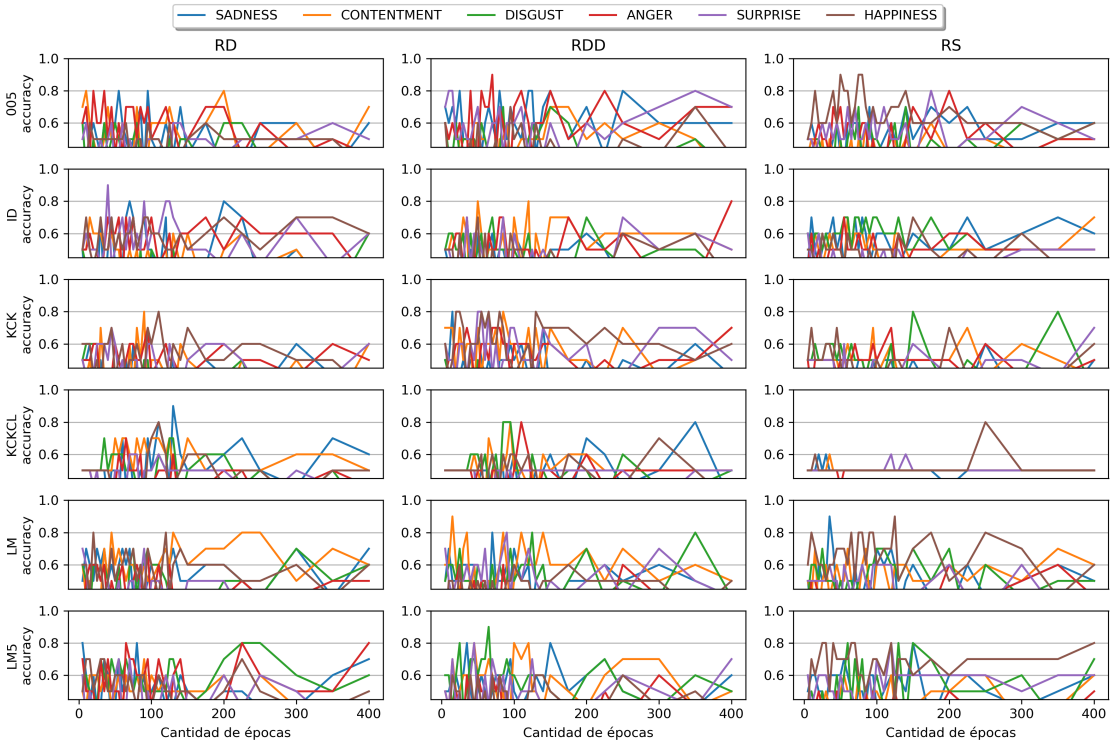

Fig. 6.3: Resultados de la validación (*accuracy* en función de cantidad de épocas de entrenamiento) para cada combinación entre red clasificadora y conjunto de atributos, desagregado por tipo de expresión, sin normalizar.

En la Fig. 6.3 se comparan los resultados sobre cada expresión independientemente, variando el tipo de red y el tipo de atributos y sin normalizar. Aquí se observa que una misma red puede generar resultados muy diferentes para distintas expresiones. Esta vez, se alcanzan resultados de cerca del 90 % de *accuracy*. Recordar que, como todos los videos traen la etiqueta correspondiente a la expresión, cada modelo se entrena para cada expresión independientemente, por lo tanto, cada curva proviene de un modelo distinto.

Surge entonces la pregunta de si se podrá obtener un nuevo clasificador que utilice, según el tipo de expresión, una red específica para clasificarla y así obtener *accuracy*

mucho más elevada. Para cada una de las 6 expresiones, hay un clasificador que alcanza al menos 80 % (no todos están en el gráfico anterior). Entonces, al combinarlos de esta manera, el clasificador final debería obtener más de 80 % de *accuracy* sobre toda la base de validación. El problema para poner a prueba esta teoría es que los datos de evaluación no tienen la etiqueta de veracidad, por lo cual no pueden ser usados para contrastar los resultados. En resumen, se necesitarían 3 bases. La primera para entrenar las redes, la segunda para determinar el mejor clasificador para cada expresión y la tercera para verificar los resultados. Como no se tiene una tercera base, se propone dividir la base de validación a la mitad y usar la primera mitad (V501) para seleccionar los mejores clasificadores y la segunda mitad (V502) para validar los resultados.

6.3. Validación en base a Confianza

Partiendo de lo dicho en la sección anterior, se buscó rankear los distintos clasificadores para cada una de las seis expresiones. Para esto, se utilizaron las predicciones sobre la mitad de la base de validación (V501).

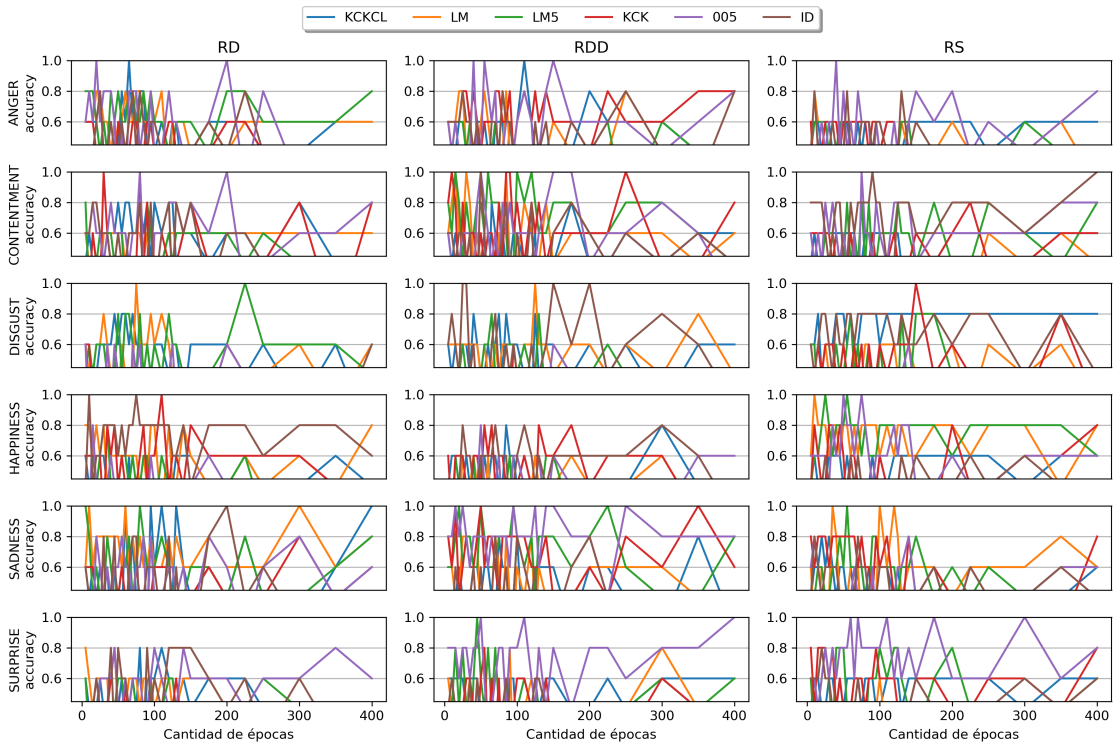

Fig. 6.4: Resultados de la validación (*accuracy* en función de cantidad de épocas de entrenamiento) sobre la primera mitad de la base de validación, para cada combinación entre red clasificadora y conjunto de atributos, desagregado por tipo de expresión, sin normalizar.

El problema es que, al ser tan pequeña la base de validación, muchas redes quedan empatadas en primer lugar con 100 % de *accuracy*. En la Fig. 6.4 se muestra la *accuracy* por expresión alcanzada para cada combinación entre tipo de red y tipo de atributos. Una vez más, sólo se muestran las redes recurrentes sin normalización. Al tener sólo 5 sujetos

la *accuracy* sólo puede ser 0 %, 20 %, 40 %, 60 %, 80 % y 100 %. Debido a ello, es difícil determinar cuál es la mejor combinación de hiperparámetros para cada expresión.

Para solucionar este problema se pensó una métrica distinta que permita rankear a los clasificadores. Hasta ahora se consideró que el resultado de la evaluación de una red podía tomar uno de dos posibles valores, correspondientes a los dos posibles valores de verdad. Sin embargo, el resultado de la evaluación consiste en realidad en dos números que representan qué tanta confianza tiene la red de que la instancia clasificada pertenece a cada clase. Para determinar la clase asignada por la red se toma aquella para la cual este valor de confianza es más alto. Entonces, si una red A le asigna a la instancia X 0.3 de valor TRUE y 0.2 de valor FAKE, quedará empatada con una red B que le asigne a la misma instancia X 1.5 de valor TRUE y 0.1 de valor FAKE. Sin embargo, debe considerarse que la red B es "mejor" clasificando la instancia X (asumiendo que X efectivamente es sincera), ya que tiene más confianza en clasificación que realizó. Utilizando la medida de confianza descrita en la sección 5.3, se logra un ranking sin empates.

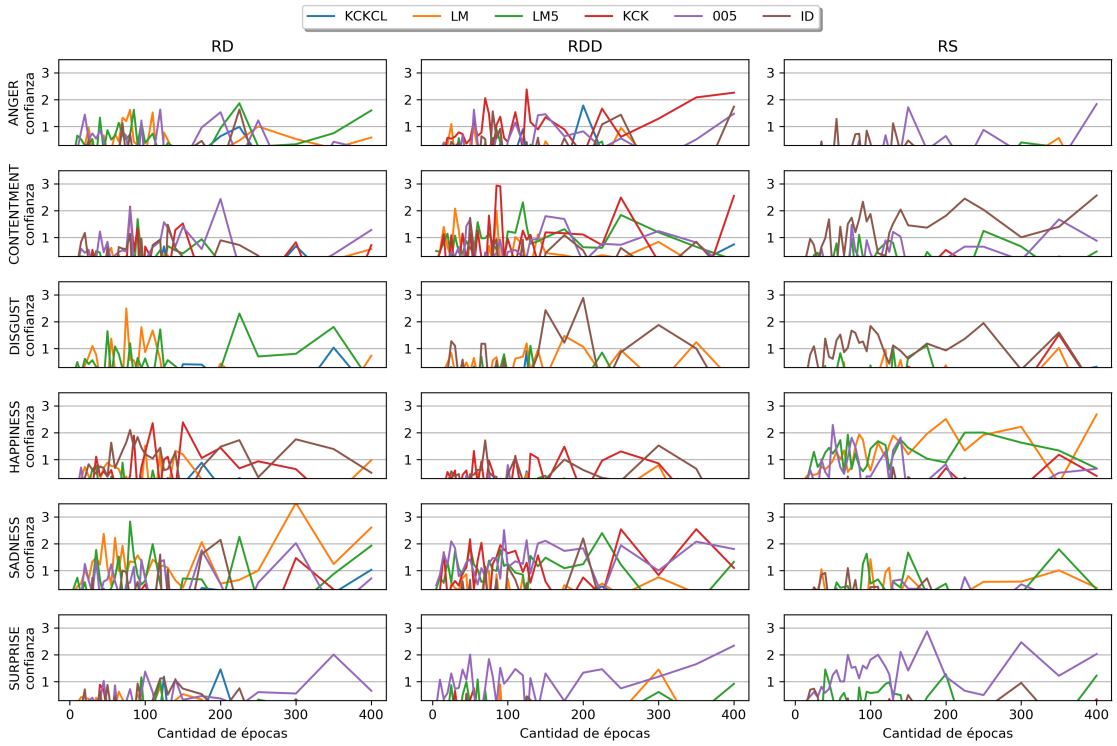

Fig. 6.5: Resultados de la validación (confianza en función de cantidad de épocas de entrenamiento) sobre la primera mitad de la base de validación, para cada combinación entre red clasificadora y conjunto de atributos, desagregado por tipo de expresión, sin normalizar.

La Fig. 6.5 vuelve a mostrar los resultados de evaluar la primera mitad de la base de validación (V501) pero en lugar de medir la *accuracy*, se mide la confianza. Esta métrica permite diferenciar mejor la performance de cada red y así rankearlas mejor en función de qué tan buenas son clasificando. Habiendo elegido la mejor red para cada expresión (aquella con mayor nivel de confianza para la expresión correspondiente) se generó un

nuevo clasificador cuyo funcionamiento consiste simplemente en derivar la clasificación a la red seleccionada para el tipo de expresión del video de entrada. La evaluación de este clasificador se realizó clasificando la segunda mitad de la base de validación (V502).

El resultado final de este clasificador no arrojó buenos resultados. La tabla de la Fig. 6.6 muestra la *accuracy* alcanzada sobre cada expresión sobre la base V502 al usar los modelos que mejor clasificaron (con mayor confianza) cada expresión en la base V501.

HAPPINESS	SADNESS	CONTENTMENT	DISGUST	ANGER	SURPRISE	TOTAL
40 %	60 %	40 %	0 %	40 %	20 %	33 %

Fig. 6.6: Resultados de la validación (porcentaje de *accuracy*) sobre la segunda mitad de la base de validación (V502, utilizada como base de evaluación en este último experimento), utilizando para cada expresión la red con mayor nivel de confianza al clasificar la primera mitad de la base de validación (V501), desagregado por tipo de expresión y total.

Una vez más, se aprecia que la generalización no funciona bien. Se utilizaron 5 sujetos para determinar los mejores clasificadores y dichos clasificadores se utilizaron luego para clasificar otros 5 sujetos. Al observar los modelos seleccionados para realizar la clasificación se ve que casi todos ellos son redes profundas, aunque al tomar los primeros 20 (Fig. 6.7), son las variantes recurrentes las que predominan. En cuanto a los tipos de atributos (Fig. 6.8), LM y LM5 resaltan bastante en general, aunque en los casos particulares de CONTENTMENT y DISGUST, los tipos predominantes parecen ser ID, KCK y KCKCL.

	HAP	SAD	CON	DIS	ANG	SUR
#1	P1	P1	P1	P1	RD	P1
#2	RD	P1	P2	P1	P1	RD
#3	RD	RS	RS	P2	RDD	RD
#4	RD	RS	RD	P1	RD	RD
#5	RD	RS	RDD	RDD	RDD	RD
#6	RD	RDD	RD	RD	RD	RD
#7	RD	RS	RDD	RDD	RD	RD
#8	RD	P2	P2	RD	RD	RD
#9	RD	RD	RS	P2	RDD	RD
#10	RD	RD	RS	P1	RDD	RD
#11	RD	RS	RDD	RDD	RD	RD
#12	RD	RS	RDD	RDD	RDD	RD
#13	RD	RD	RDD	RDD	RD	RD
#14	RD	P2	RDD	RDD	RDD	RD
#15	RD	RD	RDD	P2	RD	RD
#16	RD	RD	RDD	RDD	RD	RD
#17	RD	RD	RDD	P1	RDD	RD
#18	RD	RS	RDD	RD	RD	RD
#19	RS	RD	RDD	P1	RDD	RS
#20	RD	RS	RS	RS	RS	RD

Fig. 6.7: Red de clasificación utilizada en cada uno de los 20 clasificadores con mayor nivel en la medida de confianza.

	HAP	SAD	CON	DIS	ANG	SUR
#1	005	LM5	KCKCL	ID	LM	005
#2	LM5	LM	ID	005	LM	LM5
#3	LM5	LM	ID	ID	005	LM5
#4	005	LM	ID	KCKCL	LM	005
#5	LM5	LM	ID	ID	005	LM5
#6	LM5	ID	ID	LM	LM	LM5
#7	LM5	LM	KCK	ID	ID	LM5
#8	LM5	LM5	005	LM5	ID	LM5
#9	LM5	LM5	ID	KCKCL	005	LM5
#10	LM5	005	005	LM	LM5	LM5
#11	LM5	LM	KCK	LM5	LM	LM5
#12	LM5	LM	KCK	LM	005	LM5
#13	LM5	LM	ID	ID	005	LM5
#14	LM	LM5	KCK	LM	005	LM
#15	LM5	LM5	ID	ID	005	LM5
#16	LM	005	KCK	ID	005	LM
#17	LM5	LM5	ID	KCKCL	LM5	LM5
#18	005	LM	KCK	ID	ID	005
#19	005	LM5	ID	KCKCL	005	005
#20	LM5	LM	ID	LM	LM	LM5

Fig. 6.8: Conjunto de atributos utilizado en cada uno de los 20 clasificadores con mayor nivel en la medida de confianza.

7. CONCLUSIONES

Los experimentos con LOO mostraron que los rasgos que permiten identificar la veracidad de una expresión son muy relativos a cada sujeto. Utilizando tan pocos datos las redes obtuvieron resultados muy poco precisos, haciendo evidente su falta de capacidad para generalizar. Esto lleva a pensar que no existen rasgos genéricos que permitan distinguir una expresión sincera de una fingida en general (como pasaba con la sonrisa, según los experimentos de Duchenne de Bologne), o que sí existen pero son muy variados entre los sujetos y por lo tanto, se requieren muchos más datos para poder aprenderlos todos.

Las distintas variantes de redes recurrentes se comportaron en promedio mejor que las dos variables de redes profundas, tendiendo estas últimas a sobreajustar mucho más rápido. Si bien no se puede concluir que en general las redes recurrentes son mejores (en cuanto a su capacidad de predicción) para clasificar videos, ya que las diferencias en los resultados son mínimos, sí se puede remarcar que alcanzaron niveles de *accuracy* similares en mucho menos tiempo y con arquitecturas mucho más simples (en cuanto a cantidad y tamaño de capas). Se concluye entonces que sí es recomendable utilizar redes neuronales recurrentes, en lugar de redes *feed-forward* profundas, para problemas de clasificación de videos. Los resultados obtenidos no necesariamente serán mejores pero sí se alcanzarán más rápido y utilizando menos recursos (en particular, espacio de almacenamientoy memoria ram para guardar los modelos entrenados).

Variar la cantidad de cuadros del video no permitió concluir que existan pequeños rasgos que sean determinantes al momento de clasificar la veracidad de una expresión. La propuesta del desafío sugería que detectar microexpresiones (pequeñas en movimiento y cortas en el tiempo) podía facilitar la tarea de clasificación. En base a dicha hipótesis es que desarrollaron la base con alta definición temporal (100 cuadros por segundo) y espacial (resolución de 1280x960 píxeles). Sin embargo, al variar únicamente la cantidad de cuadros tomados para realizar la clasificación, no se observa ninguna relación entre esta y la *accuracy* obtenida.

La escasez de datos es un problema importante. Las estrategias de Aprendizaje Automático se basan en analizar grandes volúmenes de datos para extraer patrones que permitan generalizar clasificaciones o tomas de decisiones. El problema de la estimación de veracidad en una expresión facial en video recién ha comenzado a ser explorada y la reducida cantidad de datos no permite avanzar mucho más. Una posibilidad para seguir profundizando es cambiar el enfoque y utilizar como entrada imágenes en lugar de videos. Al generar una nueva base de imágenes extrayendo los cuadros de cada video se puede obtener una base mucho más amplia sobre la cual experimentar. Sin embargo, tal tarea requiere filtrar varios cuadros de ruido, sobre todo los primeros y los últimos ya que seguramente aporten poca información o hasta información incorrecta.

Bibliografía

[1] Shima Alizadeh and Azar Fazel. Convolutional neural networks for facial expression recognition. *CoRR*, abs/1704.06756, 2017.

[2] D. A. Kashy M. M. Wyer B. M. DePaulo, S. E. Kirkendol and J. A. Epstein. Lying in everyday life. journal of personality and social psychology, 1996.

[3] Marian Stewart Bartlett, G. C. Littlewort, Mark G. Frank, and Kang Jun Lee. Automatic decoding of facial movements reveals deceptive pain expressions. *Current Biology*, 24:738–743, 2014.

[4] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

[5] C. A. Corneanu, M. O. Simón, J. F. Cohn, and S. E. Guerrero. Survey on rgb, 3d, thermal, and multimodal approaches for facial expression recognition: History, trends, and affect-related applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1548–1568, Aug 2016.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[7] G. Duchenne de Bologne. The mechanism of human facial expression, 1862.

[8] Paul Ekman. Are there basic emotions? *Psychological review*, 99 3:550–3, 1992.

[9] Paul Ekman. An argument for basic emotions, 1992.

[10] Paul Ekman. Handbook of cognition and emotion, chapter 3: Basic emotions, 1999.

[11] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[12] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *J. Mach. Learn. Res.*, 3:115–143, March 2003.

[13] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities, 1982.

[14] M. E. Hoque, D. J. McDuff, and R. W. Picard. Exploring temporal patterns in classifying frustrated and delighted smiles. *IEEE Transactions on Affective Computing*, 3(3):323–334, July 2012.

[15] Y. Kim and X. Huynh. Discrimination between genuine versus fake emotion using long-short term memory with parametric bias and facial landmarks. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 3065–3072, Oct 2017.

[16] K. Kulkarni, C. Corneanu, I. Ofodile, S. Escalera, X. Baro, S. Hyniewska, J. Allik, and G. Anbarjafari. Automatic recognition of facial displays of unfelt emotions. *IEEE Transactions on Affective Computing*, pages 1–1, 2018.

[17] Gwen Littlewort, Marian Stewart Bartlett, and Kang Lee. Faces of pain: automated measurement of spontaneousallfacial expressions of genuine and posed pain, 2007.

[18] Gwen C. Littlewort, Marian Stewart Bartlett, and Kang Lee. Automatic coding of facial expressions displayed during posed and genuine pain. *Image Vision Comput.*, 27(12):1797–1803, November 2009.

[19] S. Ozkan and G. B. Akar. Relaxed spatio-temporal deep feature aggregation for real-fake expression prediction. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 3094–3100, Oct 2017.

[20] Alexandru Savoiu and J. Wong. Recognizing facial expressions using deep learning.

[21] F. Saxen, P. Werner, and A. Al-Hamadi. Real vs. fake emotion challenge: Learning to rank authenticity from facial activity descriptors. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 3073–3078, Oct 2017.

[22] J. Wan, S. Escalera, G. Anbarjafari, H. J. Escalante, X. Baro, I. Guyon, M. Madadi, J. Allik, J. Gorbova, C. Lin, and Y. Xie. Results and analysis of chalearn lap multi-modal isolated and continuous gesture recognition, and real versus fake expressed emotions challenges. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 3189–3197, Oct 2017.

[23] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1798–1807, June 2015.