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

Estudio de redes profundas livianas con aprendizaje basado en distribuciones aplicadas al reconocimiento facial

Tesis de Licenciatura en Ciencias de la Computación

Nicolás Mastropasqua

Director: Daniel Acevedo

Buenos Aires, 2023

ESTUDIO DE REDES PROFUNDAS LIVIANAS CON APRENDIZAJE BASADO EN DISTRIBUCIONES APLICADAS AL RECONOCIMIENTO FACIAL

Hoy en día, la búsqueda de soluciones lightweight que logren resultados comparables a modelos de Deep learning robustos ha recibido particular atención debido a su implementación factible en dispositivos móviles. Uno de los problemas que podrían aprovechar esta cualidad es el de Facial Expression Recognition (FER). Considerando que una gran cantidad de datasets de expresiones faciales suelen estar anotados con emociones categóricas cuando en realidad la mayoría de las expresiones exhibidas en escenarios in-the wild ocurren como combinaciones o composición de emociones básicas, se puede hacer uso de Label Distibution Learning (LDL) como estrategia para el entrenamiento. En este trabajo se abordará el problema de FER a través de redes neuronales livianas entrenadas con LDL. Bajo el supuesto de que las imágenes de expresiones faciales deberían tener una distribución de emoción similar a la de su vecindad en un espacio de etiquetas auxiliares adecuado, como aquel determinado por la tarea de Action unit recognition, se puede aprovechar la información de las distribuciones e incorporarla como parte la función de pérdida. Concretamente, se estudiarán en profundidad dos arquitecturas lightweight del Estado del arte, EfficientFace y CERN, y se analizará el impacto de distintos acercamientos para implementar LDL considerando datasets in the wild como RAF-DB, CAER-S, FER+ y AffectNet.

Palabras claves: Facial expression recognition, Label distribution learning, Lightweight, Convolutional Neuronal Network, Action unit recognition

ESTUDIO DE REDES PROFUNDAS LIVIANAS CON APRENDIZAJE BASADO EN DISTRIBUCIONES APLICADAS AL RECONOCIMIENTO FACIAL

Nowadays, the search for lightweight solutions that achieve comparable results to those of heavy deep learning models has received increasing attention due to a feasible implementation on mobile devices. One of the problems that might benefit from this approach is the task of Facial Expression Recognition (FER). Considering the fact that datasets usually come with categoric labeling but most emotions occur as combinations, mixtures, or compounds of the basic emotions, we make use of label distribution learning (LDL) as a training strategy. In this thesis we deal with the FER problem using lightweight neuronal networks and LDL. We further assume that facial images should have similar emotion distributions to their neighbors when the right auxiliary task is considered, like the Action Unit Recognition problem. This neighbors' distribution information is captured in the loss function to help the LDL training process. Specifically, we conduct a study of two state-of-the-art architectures, EfficientFace and CERN, to then analyze the impact of using different approaches to LDL on a variety of in-the-wild datasets: RAF-DB, CAER-S, FER+ and AffectNet.

Palabras claves: Facial expression recognition, Label distribution learning, Lightweight, Convolutional Neuronal Network, Action unit recognition

Índice general

1. INTRODUCCIÓN

1.1. El problema del reconocimiento de expresiones faciales (FER)

El reconocimiento de expresiones faciales o **F**acial **E**xpression **R**ecognition (de ahora en más FER por su sigla en inglés) es la tarea de clasificar expresiones en un conjunto de emociones básicas a partir de imágenes de rostros. Está presente en numerosos campos de interés que presentan desafíos como la asistencia de la conducción a través de la detección de fatiga según expresiones faciales [24], el reconocimiento de estados emocionales en robots sociales [34] [9] y la clasificación del trastorno del espectro autista [29] entre tantos otros.

El estudio de las expresiones faciales tiene una larga historia y una de sus perspectivas más influyentes, que tiene sus orígenes en el siglo XIX con Charles Darwin, sostiene que son acciones residuales a respuestas más complejas de adaptación al medio [9]. Modelos más recientes enfatizan el poder de expresividad que tienen los gestos faciales para comunicar información vital para la supervivencia [44]. Por lo tanto, la habilidad de exagerar la expresión original y la de interpretación de las mismas habría formado parte del proceso de selección. Existe evidencia [43] [4] que muestra que los músculos esenciales para representar ciertas emociones son universales, incluso hay numerosos estudios que sostienen la hipótesis que las expresiones faciales son interpretadas a niveles similares tanto en culturas occidentales como orientales [44].

Dentro del mundo de la detección automática es necesario definir algún tipo de codificación para describir las expresiones faciales. Una de las prácticas más extendidas, probablemente por su simpleza, es marcadamente subjetiva ya que clasifica a las expresiones en las emociones subyacentes que se supone indujeron dichos gestos. La elección de las categorías tiene sus origines en el trabajo de Paul Ekman [11], donde se distinguen seis emociones básicas (felicidad, tristeza, miedo, enojo, asco y sorpresa). Otra corriente popular, también con un acercamiento notablemente subjetivo, es mapear una expresión a un espacio bidimensional basado en *valence* (que tan agradable es el sentimiento) y *arousal* (el grado de activación) que en principio admite mayor riqueza y la posibilidad de captar emociones más sutiles.

Por el contrario, existen alternativas de codificación descriptivas que se enfocan en el tipo de movimiento que puede hacer el rostro humano. Una de las más influyentes es el Facial Action Coding System (FACS) desarrollado por Ekman y Friesen [12] que codifica cada posible movimiento involucrado en una expresión en acciones de músculos faciales denominadas Action Units (AUs). A partir de esto, se puede definir una codificación híbrida mezclando ambas ideas como el caso de EMFACS [40] que determina emociones a partir de acciones de músculos específicas (AUs) en lugar de la creencia sobre la emoción subyacente. Tanto este acercamiento híbrido como el categórico asumen una correspondencia uno a uno entre la acción retratada en la cara y la emoción subyacente, pero esto podría quedarse corto ya que una emoción puede dar como resultados múltiples expresiones.

Para poder entrenar modelos de detección automática en el contexto de FER, se emplean datasets que, dependiendo de las condiciones en las que fueron generados, pueden impactar fuertemente en la complejidad de la tarea de reconocimiento. En primer lugar puede que la toma de las muestras haya sido a partir de una única fotografía o bien a partir de una secuencia de video que muestra el desarrollo de la emoción. En este último

caso se suele alimentar a los modelos a partir de fotogramas extraídos de momentos claves de la secuencia que muestra la emoción, la cual suele separarse en tres etapas: onset, apex y offset. Por otro lado, las condiciones ambientales en las cuales se toman las imágenes están sujetas a variables como las condiciones de iluminación, los ángulos de cámara e inclinación de la cabeza, el método de obtención de las imágenes (posado o espontáneo, a veces también llamado in-the-wild), la complejidad de la escena que rodea al sujeto y la presencia de oclusiones.

Por ejemplo, Cohn-Kanade (CK) [23] fue uno de los primeros datasets de dominio público utilizado para FER, en donde 123 sujetos realizaron expresiones que reflejaban las emociones de manera posada en distintas secuencias de video y en condiciones controladas de laboratorio como puede verse en la Figura 1.1. Distinto a lo anterior, existen actualmente múltiples datasets para FER donde se obtienen las imágenes directamente de Internet y luego se las etiqueta con crowd sourcing, como en FER+ [2] o bien con anotadores entrenados como en Affectnet [38]. Aunque este último acercamiento insuma más tiempo y requiera de cierta habilidad específica, podría dar lugar a anotaciones menos inconsistentes.

Fig. 1.1: Ejemplos de muestras de datasets de FER. Las imágenes de la fila superior se corresponden al dataset Cohn Kanade. Las secuencias de video fueron registradas a partir de un expresión Neutral (onset) hasta llegar al momento de máxima expresión en el último frame (apex), siendo este el obtenido como representante. En la fila inferior se muestran algunos ejemplos del dataset FER+. Ambas filas se ordenan por emoción: Enojo, Desprecio, Asco, Miedo, Felicidad, Sorpresa, Tristeza, Neutral.

Dentro de las etapas iniciales de los frameworks para el reconocimiento de expresiones se encuentran las de *face detection* y *face registration*. Para la primera, ciertos modelos de deep learning [50] muestran robustez a condiciones desafiantes como las que representan los datasets in-the-wild. Estos métodos delimitan la región del rostro mediante una bounding box que luego puede usarse para ajustar un recorte. Para el caso de face registration en 2D, los modelos suelen detectar una serie de features geométricos en el espacio del rostro (como los *fiducial points* o *landmarks*). Debido a que la orientación relativa de la cabeza con respecto a la imagen puede no estar alineada, se suelen utilizar estos features para la frontalización (proceso por el cual se alinea la cara a una posición central).

1.2. FER con Deep Learning

Desde el resurgimiento de las redes neuronales profundas, las arquitecturas convolucionales demostraron ser suficientemente robustas a las transformaciones afines que surgen en problemas como Image Recognition y FER en particular. Sin duda la aparición de AlexNet [26] marcó un hito en el área de Computer Vision, siendo una de las primeras

redes convolucionales profundas en aprovechar la potencia de cómputo de las GPUs para el entrenamiento y siendo la ganadora de la competencia ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) por un amplio margen de 10.8 %. Tras su introducción, le siguió una ola de arquitecturas de redes convolucionales profundas que fueron consiguiendo muy buenos resultados en este tipo de problemas como VGG [45], GoogLeNet [47], ResNet [17] y DenseNet [21] entre otras. Sin embargo, mantener un buen desempeño en escenarios in-the-wild aún continúa siendo un desafío.

Por otro lado, debido a la gran cantidad de capas empleadas, el consumo de memoria y el costo en términos de operaciones de coma flotante (o FLOPs del inglés floating point operations) ha ido en aumento. Es así que el incremento en la complejidad de las redes introduce un trade-off entre costo computacional y la capacidad de predicción del modelo cuando la plataforma objetivo tiene recursos acotados (drones, móviles y autos entre otros). Esto motivó el diseño de arquitecturas lightweight que intenten ajustarse a dicho trade-off, como Xception [7], MobileNet [19], ShuffleNet [33] por nombrar algunas que, entre otras estrategias, explotan el uso de depthwise separable convolutions y group convolutions para reducir la cantidad de parámetros y FLOPs. Alternativamente, surgieron otras arquitecturas [27] como resultado del método *Neuronal Architecture Search* que automatiza parcialmente la optimización de modelos dadas restricciones particulares sobre la complejidad y tipos de arquitectura objetivo. Sin embargo, todas estas redes también se ven perjudicadas cuando se las evalúa en escenarios in-the-wild.

Otro de los problemas con los cuales se tiene que lidiar al momento de diseñar modelos para FER, y que afecta particularmente al caso de Deep Learning, es la inconsistencia en el etiquetado de grandes volúmenes de datos. En este caso, las inconsistencias pueden venir de dos fuentes: la ambigüedad intrínseca que surge en la tarea de clasificar expresiones faciales en emociones y el sesgo de anotación que depende del tipo de entrenamiento, del criterio e incluso del contexto cultural de los anotadores. Más aún, este sesgo es evidente entre datasets como se ilustra en la Figura 1.2, donde dos grupos de instancias de los datasets AffectNet y RAF-DB [30] que podrían ser percibidas como muy similares, tienen etiquetas distintas (fear y disgust respectivamente). Esto puede inducir un sesgo de clasificación cuando los modelos entrenados predicen nuevas instancias y disminuir el nivel de generalización que se puede alcanzar. De hecho, el entrenamiento en conjunto de varios datasets se torna más difícil, ya que el modelo debería poder tolerar y aprender a pesar de este tipo de inconsistencias.

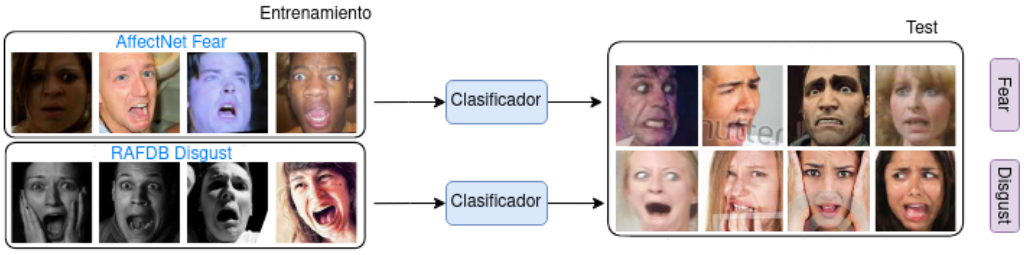


Fig. 1.2: Casos de sesgo de anotación entre datasets. El grupo de instancias de la izquierda tienen expresiones muy similares. Sin embargo, las de la fila superior son etiquetadas como fear en AffectNet y las de la fila inferior como disgust en RAF-DB. Esta situación produce un sesgo en la clasificación de los modelos ante instancias no vistas y dificulta el nivel de generalización cross-dataset.

Según Gera et al. [15] hay tres componentes que vienen siendo exitosos para FER in-the-wild: mecanismos de atención que enfatizan regiones relevantes para el problema, mecanismos de captura de features locales y globales y transfer learning [56] desde el dominio del reconocimiento facial. Basados en estos principios, su arquitectura lightweight CERN se muestra más robusta bajo situaciones de oclusión y variación de pose que otras del estado del arte.

La arquitectura EfficientFace propuesta por Zhao et al. [52] surgió a partir de adaptaciones de arquitecturas lightweight al contexto de FER in the wild, y logró resultados alentadores en datasets como RAF-DB [30] y CAER-S [28]. Para ello, su arquitectura aprovecha el diseño de ShuffleNet v2 [33] como red base a la que se le agregan algunos bloques convolucionales de atención inspirados en BAM [41] y bloques para combinar features locales-globales para ser más robusta ante cambios en las poses y oclusiones pero sin dejar de lado la restricción de complejidad. Además, para intentar compensar la posible pérdida de rendimiento por la elección de un diseño menos complejo, EfficientFace usa Label Distribution Learning (LDL) [54] [14], una estrategia alternativa al uso de etiquetas únicas que intenta hacer el entrenamiento más robusto teniendo en cuenta que la mayoría de las expresiones faciales pueden entenderse como con una combinación de emociones de distintas intensidades [42]. Relacionado con esto, en Chen et al. [5] se utilizó esta misma estrategia pero incorporando información directamente de la topología de un espacio de etiquetas descriptivas auxiliares, como Action Units [12] o Landmarks, para tratar de abordar el problema del entrenamiento con anotaciones inconsistentes.

1.3. Objetivos del trabajo

Nuestro trabajo se centra en el análisis de EfficientFace y CERN como arquitecturas lightweight para el problema de FER in-the-wild. La primera hipótesis que se tiene es que ambas pueden alcanzar desempeño equiparable a redes más complejas sobre múltiples datasets. Para ello, como objetivo inicial intentamos replicar los resultados obtenidos en los trabajos originales [15] [52]. Consideramos los datasets utilizados como benchmark en cada trabajo y agregamos otros a efectos de una comparación más exhaustiva del desempeño de estos modelos: RAF-DB [30], CAER-S [28], FER+ [2] y AffectNet [38].

Luego se estudia el uso Label Distribution Learning (LDL) [54] [14] como técnica alternativa de aprendizaje en redes profundas livianas en el contexto in-the-wild. Como hipótesis, sostenemos que LDL debería mejorar el desempeño de ambas arquitecturas y además hacer el entrenamiento más robusto al ruido en las anotaciones. Para el caso de EfficientFace buscamos conseguir resultados similares a los publicados en el trabajo original donde se exhiben mejoras entre 1 % y 2 % sobre RAF-DB Y CAER-S. En CERN, en cambio, se introduce esta técnica esperando ver una tendencia similar. A su vez, basándonos en el trabajo de Chen et al. [5], buscamos mejorar Label Distribution Learning guiando el aprendizaje mediante la topología del espacio de etiquetas auxiliares definida por Action Units. Pensamos que, además, este agregado debería hacer la técnica aún más robusta a inconsistencias.

Finalmente, aunque la comparación del desempeño de los modelos se hace teniendo en cuenta al accuracy rate como medida principal, también resulta de interés evaluar la complejidad a través de una medida más directa que los FLOPs como lo es la velocidad de inferencia.

El resto del informe se estructura como sigue: En la Sec. 2 se hace una breve revisión

de los métodos más relevantes para esta tesis: redes neuronales convolucionales lightweight y acercamientos para Label Distribution Learning; En la Sec. 3 se desarrollan los métodos sobre los cuales se basa este trabajo, haciendo énfasis en sus aplicación al problema de FER in-the-wild. En la Sec. 4 se discuten los experimentos ensayados junto con sus resultados. Finalmente en la Sec. 5 se exponen las conclusiones del trabajo junto con posibles líneas a futuro.

2. TRABAJOS RELACIONADOS

En el corazón de todos los modelos de redes neuronales profundas que se estudian en este trabajo para el problema de FER se encuentra la operación de convolución. En realidad, aunque coloquialmente se haga uso de este término, la operación que técnicamente se implementa recibe el nombre cross-correlation.

Considerando el caso donde el input de la operación es de dos dimensiones, por ejemplo una imagen en escala de grises, la transformación consiste en superponer un kernel (básicamente una matriz de pesos) sobre la imagen y deslizarlo espacialmente. Cada vez que se superpone sobre una región de la imagen distinta, combina linealmente la información de los píxeles para obtener un nuevo valor. Al desplazarlo por toda la imagen, se obtiene una nueva imagen filtrada como se aprecia en el ejemplo de la Figura 2.1.

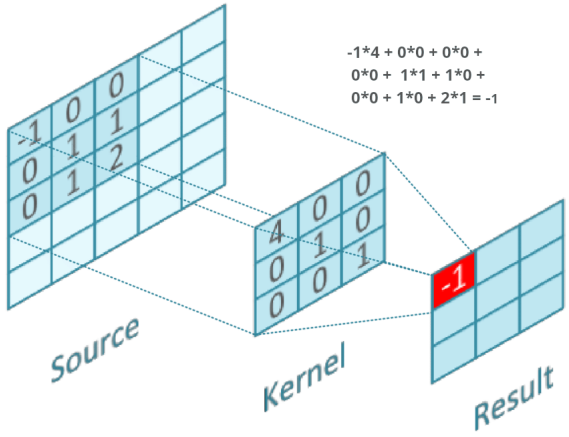


Fig. 2.1: Ejemplo para el caso $2D$ de una operación de convolución. En cada paso de deslizamiento, se hace la multiplicación lugar a lugar de la submatriz en donde está superpuesto el kernel y se suman los términos. El resultado determina el nuevo pixel en una nueva imagen. Para este caso, el procedimiento comienza haciendo coincidir el pixel de la esquina superior izquierda del kernel con el respectivo pixel de la primera fila de la imagen. Para que no se indefina la operación, las últimas dos columnas y las últimas dos filas no deberían ser visitadas. Por esta razón el resultado que se observa es de menor dimensión, aunque hay distintas variantes de convoluciones para preservar el tamaño. Técnicamente la operación descrita es la cross-correlation y la diferencia con una convolución es que el filtro se superpone habiéndolo rotado 180°.

La elección de los pesos del kernel es fundamental para determinar el efecto del filtro sobre la imagen original. Por ejemplo, el filtro Sobel con kernel $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ dejan pasar la información de los bordes como se ve en la Figura 2.2

A través de los algoritmos de optimización para el entrenamiento de redes neuronales, como Stochastic Gradient Descent, se pueden aprender los pesos adecuados de un kernel para destacar distintas features de una entrada. Como todas las regiones que visita el kernel comparten los mismos pesos, y la aplicación del kernel debe ser útil globalmente en

Fig. 2.2: Resultado de aplicar el filtro Sobel para detectar los bordes de la imagen de la izquierda.

todo el espacio, el aprendizaje debe ser lo suficientemente general.

La operación de convolución es una transformación lineal y por lo tanto puede representarse como una capa más dentro de una arquitectura de redes neuronales. Además por ser transformación lineal, se puede obtener una matriz asociada e implementarse, en su forma naive, como en la Figura 2.3. Notar que con inputs de grandes tamaños la matriz asociada se vuelve rala, lo cual podría tener implicaciones relevantes en términos de eficiencia y consumo de memoria.

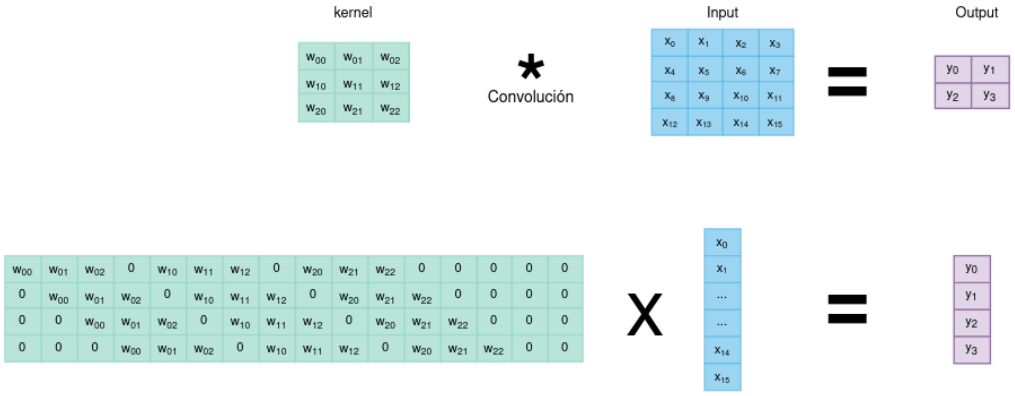


Fig. 2.3: Desarrollo de la operación de convolución como multiplicación matriz-vector. Se puede ver que las filas de la matriz de pesos se pueden obtener con un shift a derecha.

De forma más general, la convolución puede ser aplicada para inputs con más de un canal, como podría ser el caso de imágenes RGB. En este caso el filtro ahora se aplica en un espacio $3D$ con dos dimensiones espaciales (alto y ancho) y otra para los canales, logrando así el mapeo de correlaciones cruzadas a nivel canal y espacial en un nuevo resultado. Además si se procesa el mismo input a partir de un conjunto de filtros, cada uno de los cuales genera un nuevo resultado que concatenado con el resto a lo largo de la dimensión de canales determina un nuevo feature map 3D como se ilustra en la Figura 2.4. Este esquema puede generalizarse y abstraerse en una capa convolucional, en la cual también se suele aplicar una transformación no lineal al resultado que se conoce como función de activación (ejemplos de estas son ReLU, Leaky ReLU, Sigmoide, Tanh, etc).

Las arquitecturas convolucionales más simples son esencialmente aplicaciones encadenadas de estas capas. Se puede pensar que a lo largo de ellas, las convoluciones actúan como una forma de extracción de features cuyos niveles de abstracción crecen a medida

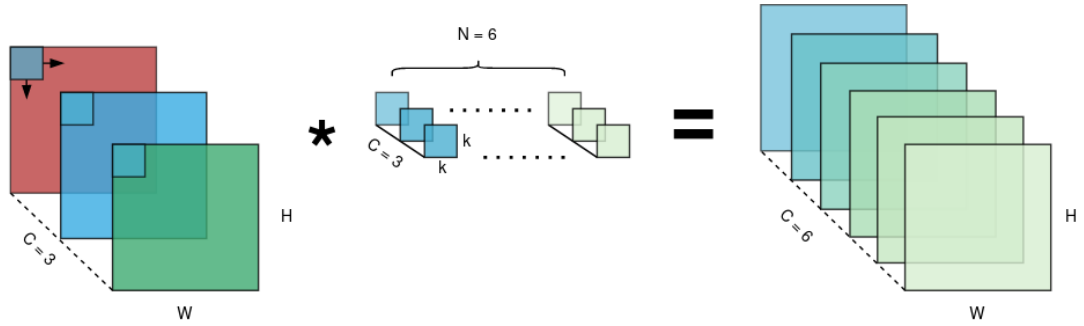


Fig. 2.4: Ejemplo de convolución $3D$ a partir de un input de 3 canales. Cada filtro de tamaño $k \times k \times 3$ ahora se aplica a nivel espacial y entre-canales sobre el input. Cada sección del volumen determinado por el filtro es un kernel de $k \times k$ que se aplica sobre uno de los canales del input como en la convolución 2D. Esto generaría 3 matrices resultado, pero las mismas se combinan en una sola sumándose entre ellas. Luego, se repite este proceso con los filtros restantes generando así un feature map resultante de tamaño $W \times H \times 6$.

que consideramos capas más lejanas de la entrada original. Comparadas con las capas fully-connected, en las capas convolucionales las conexiones ralas entre estas pueden ayudar a disminuir el problema de overfitting a la vez que, junto con el aprovechamiento de las correlaciones cruzadas espaciales locales y la robustez a la invarianza traslacional, resultan adecuadas para tareas de reconocimiento de imágenes.

De ahora en más, se hará un abuso de notación al definir FLOP como la cantidad de operaciones multiply-add ($acum \leftarrow acum + (a \times b)$), similar a como se define en los distintos trabajos revisados [33]. Esta forma de contar las operaciones está relacionada con las instrucciones FMA que aprovechan las arquitecturas modernas para agilizar el cómputo.

Dado un input I de tamaño $D_w \times D_h \times M$, siendo D_w , D_h las dimensiones espaciales (ancho y alto) y M la cantidad de canales, si se aplica una operación de convolución $3D$ con filtro de tamaño $k \times k \times M$ se obtiene una salida de tamaño $D'_w \times D'_h$. Por lo tanto, aplicando N filtros como los anteriores da como resultado un feature map I' de tamaño $D'_w \times D'_h \times N$.

Entonces, para conseguir una sección de I' con tamaño $D'_w \times D'_h$ se necesitan $k^2 D'_w D'_h M$ FLOPs. Al tener que repetir la operación por cada uno de los N filtros, la cantidad de FLOPs, denotados por la variable F_1, que se necesitan para la operación de convolución $3D$ es

$$F_1 = k^2 D'_w D'_h M N \tag{2.1}$$

y la cantidad de parámetros entrenables, denotados por la variable P, serán para cada uno de los filtros $k^2 M$, resultando en $P = k^2 M N$.

2.1. Arquitecturas convolucionales lightweight

Desde la aparición de AlexNet [26] en 2012 comenzó la tendencia de hacer redes convolucionales cada vez más profundas y más anchas convergiendo en arquitecturas como VGG 2014 [45] con aproximadamente 138M parámetros.

Estos acercamientos significaron una forma directa de mejorar el desempeño general de estas arquitecturas pero a costa de aumentar su complejidad considerablemente. De hecho,

el crecimiento desmesurado de la cantidad de parámetros entrenables alentó situaciones indeseables, como overfitting, en este tipo de redes. Intuitivamente, al tener demasiadas "variables libres" el modelo tiene más grados de libertad y corre el riesgo de adaptarse excesivamente al ruido y a las singularidades propias de un dataset (situación que se puede agravar si la cantidad de instancias disponibles es limitada). En este sentido, la introducción de capas de dropout [46] o batch normalization [22] y otras técnicas de regularización permitieron mejorar esta situación.

Alternativamente podría pensarse que una salida viable para los problemas descriptos es utilizar arquitecturas con conexiones más ralas, incluso dentro de las convolucionales. Aunque se reduzca significativamente la cantidad de operaciones, el overhead provocado por cache misses y lookups en las arquitecturas actuales junto con la disponibilidad de bibliotecas optimizadas específicamente para operaciones entre matrices y vectores densos, como Pytorch, hacen que la elección de representaciones de matrices ralas sea, en ocasiones, poco ventajoso al menos en términos de velocidad [47]. Sin embargo, el aprovechamiento de este tipo de representaciones ralas sigue recibiendo atención y encontrando mejoras, más aun teniendo en cuenta el avance en el soporte[1] de las arquitecturas de GPU actuales.

Nuestro interés está enfocado en trabajos que propusieron arquitecturas diseñadas con la idea de reducir el costo computacional. En general las técnicas empleadas buscan favorecer aún más las conexiones ralas o bien implementar factorizaciones de las operaciones de las capas convolucionales en una serie de operaciones más chicas que trabajen de manera independiente sobre las correlaciones cruzadas entre canales y las espaciales.

2.1.1. Inception, Xception y Depth-wise separable convolutions

En 2014 la arquitectura Inception [47] propuso el uso de convoluciones 1×1 basándose en la idea de Network in Network [31]. Este caso particular de convoluciones 3D, donde cada filtro es de $1 \times 1 \times M$ siendo M la cantidad de canales de entrada, recibe su nombre por el tamaño de kernel utilizado y se usa principalmente como capas intermedias de reducción de dimensionalidad. La hipótesis fundamental de Inception es que las correlaciones cruzadas entre-canales y las espaciales están lo suficientemente desacopladas que es preferible no mapearlas en conjunto (sic [47]).

El módulo de Inception esencial primero captura las correlaciones entre canales a través de un conjunto de convoluciones 1×1. Cada uno de estos resultados determinan embeddings de menor tamaño sobre los cuales luego se pueden aplicar distintas convoluciones de kernels más grandes (potencialmente, con distintos tamaños de kernel), para capturan las correlaciones cruzadas a nivel espacial, y finalmente concatenar cada uno de los resultados producidos de manera independiente.

Globalmente, la arquitectura es un stack de estos módulos con aplicaciones intermedias de operaciones max-pool con stride para reducir la dimensión espacial a la mitad a medida que aumenta el ancho de los canales.

Siguiendo sobre esta idea, en 2017 en Chollet [7] se introdujo la arquitectura Xception y se observó que un módulo Inception es equivalente a aplicar primero una única convolución 1×1 que genere un único embedding que luego puede partirse en segmentos disjuntos sobre los cuales se aplican las convoluciones de kernels más grandes. Esa idea llevada al extremo resulta en la aplicación de tantas convoluciones de kernels grandes sobre tantos segmentos

[1] https://developer.nvidia.com/blog/accelerating-matrix-multiplication-with-block-sparse-format-and-nvidia-tensor-cores/

como sea posible subdividir al embedding (ver la Figura 2.5). En otras palabras, lo que se tiene ahora son aplicaciones de convoluciones 2D que solo evalúan una sección del embedding, considerando así únicamente las dimensiones espaciales. Notar que esto juega con una hipótesis más fuerte que la original ya que asume que las correlaciones cruzadas entre-canales y las espaciales pueden ser mapeadas completamente separadas.

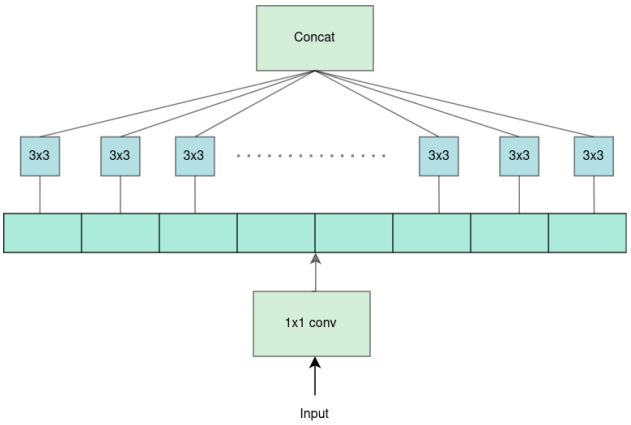


Fig. 2.5: Módulo Xception [47].

De esta forma, el módulo Xception termina siendo similar a lo que se conoce como depth-wise separable convolution. La diferencia está en el orden de las operaciones: éstas aplican primero las convoluciones por canal espacial y luego las 1×1 para reducir dimensionalidad. En [7] argumentan que está diferencia termina siendo poco relevante.

En particular, lo interesante de las depth-wise separable convolutions es que empíricamente demostraron rendimientos equivalentes a las tradicionales [19] pero ganando ampliamente en la complejidad expresada en FLOPs.

En concreto, dado un input I de tamaño $D_h \times D_w \times M$ al aplicar primero las convoluciones 2D con filtros de $k \times k \times 1$ a cada uno de los M canales por separado, se obtiene una salida intermedia S de tamaño $D'_h \times D'_w \times M$ como se puede ver en la Figura 2.6. Por lo anterior, se necesitan $D'_h D'_w M k^2$ FLOPs.

Suponiendo que se quiere llegar a un feature map I' con N canales como en 2.1, se aplicarían luego esa cantidad de convoluciones 1×1 (notar que cada uno de estos filtros tiene tamaño $1 \times 1 \times M$ pues se aplican sobre S como convolución $3D$). Por lo tanto serán necesarios otros $D'_h D'_w MN$ FLOPs. Finalmente, para computar una depth-wise separable convolution se necesitan:

$$F_2 = D'_h D'_w M(k^2 + N) \tag{2.2}$$

La cantidad de parámetros serán $k^2 M + MN$, el primer término considerando los de la primera etapa de convoluciones y el segundo la de las convoluciones 1×1.

Además, de 2.1 y 2.2 se puede ver que

$$\frac{F_2}{F_1} = \frac{1}{N} + \frac{1}{k^2} \tag{2.3}$$

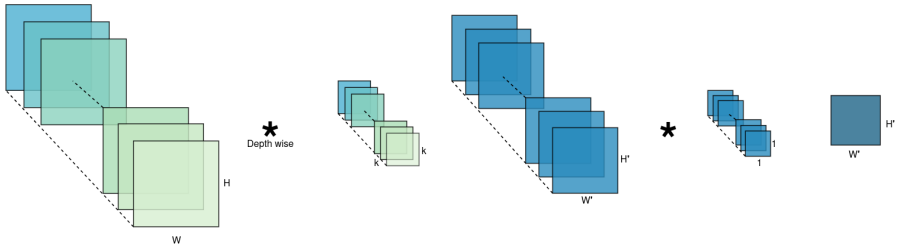


Fig. 2.6: Depth-wise separable convolution simplificada que obtiene un feature map de salida con un único canal.[3]

2.1.2. ResNet: Skip connections y bloques bottleneck

Aunque este tipo de arquitecturas no sea particularmente del tipo lightweight, tienen su lugar en este contexto por dos razones: los módulos skip connection y los bloques bottleneck.

Previo a la introducción de las arquitecturas ResNet [17] se empezó a observar una situación particular en la que en algunos modelos, al aumentar la complejidad a través de la cantidad de capas, mostraban un error de entrenamiento más alto (contrario al esperado overfitting). Intuitivamente, la noción esperada era que el incremento en la complejidad pueda mantener al menos el mismo desempeño que la versión más pequeña. Por lo tanto, debería poder facilitarse una forma en la que la red pueda "saltear" capas que no estén aportando mejoras durante el entrenamiento. Esta idea se ve ilustrada en la Figura 2.7.

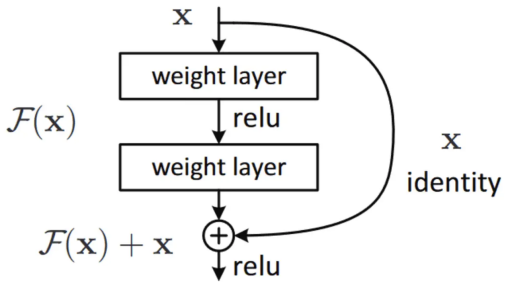


Fig. 2.7: Módulo Skip Connection [17]. Si $\mathcal{F}$ es el mapeo que aprende un stack de, por ejemplo, dos capas entonces a través de la Skip Connection debería ser más fácil que la red aprenda a atenuar lo suficiente el mapeo $\mathcal{F}$ y pueda obtenerse como salida una aproximación de la función identidad sobre el input original.

Otro acercamiento relevante que introdujeron estas arquitecturas para reducir la complejidad son los bloques bottleneck que dado un input, utilizan las convoluciones 1×1 para proyectar a una dimensión más chica y luego aplicar las convoluciones tradicionales para extraer features relevantes pero sobre el embedding. Luego, se vuelve a aplicar una convolución 1×1 para llegar a la dimensión original como se ve en la Figura 2.8.

2.1.3. ShuffleNet

La operación que mayor FLOPs insume en las depth-wise separable convolutions son las convoluciones 1×1. En la arquitectura ShuffleNet V1 [51] se hace especial énfasis en

[3] Fuente: An Efficient Multi-Level CNN Approach for White Blood Cells Classification.

Fig. 2.8: Comparación de módulos Skip connection [17]. El de la derecha con bottleneck, implementado con una convoluciones 1×1 para proyectar los feature maps a una dimensión más baja donde luego aplicar la convolución tradicional.

esta cuestión y se propone un diseño que tiene como premisa el aprovechamiento de las group convolutions.

Group Convolutions

Las group convolutions fueron introducidas en la arquitectura AlexNet para paralelizar el computo en múltiples GPUs a partir de la separación en grupos. En las convoluciones 3D tradicionales cada filtro tiene que tener la misma profundidad de canales que el input, sin embargo en este caso la idea es separarlos en grupos que toman al input como la unión de volúmenes disjuntos (ver la Figura 2.9 como referencia).

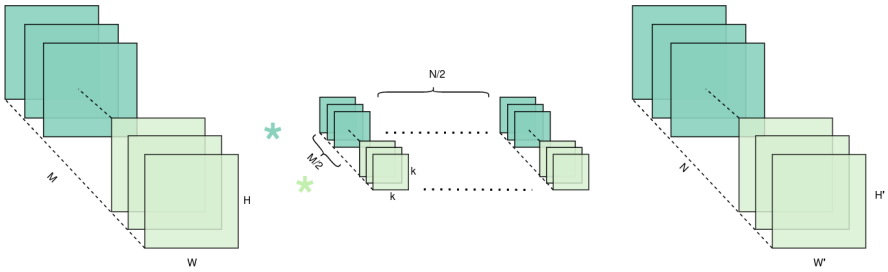


Fig. 2.9: Group convolutions con g = 2.[4]

Supongamos que se tiene un input I de tamaño $D_w \times D_h \times M$ y se quiere obtener un output I' de tamaño $D'_w \times D'_h \times N$ utilizando group convolutions con g grupos. Luego, cada filtro tiene tamaño $k \times k \times \frac{M}{g}$. Si por cada grupo tenemos $\frac{N}{g}$ filtros, el resultado de aplicar el conjunto de filtros de cada grupo a volúmenes disjuntos de I serán g feature maps de tamaño $D'_w \times D'_h \times \frac{N}{g}$. Luego al concatenar cada uno de estos grupos, se obtiene un único feature map de tamaño $D'_w \times D'_h \times N$. Es interesante notar que si $g = M$, se tiene una aplicación de convoluciones por canal como la primera etapa de una depth-wise convolution. En resumen, la cantidad de parámetros y FLOPs de las group convolutions se reducen por el factor g.

ShuffleNet V1

La premisa de esta arquitectura es la de reemplazar las convoluciones 1×1 por su versión agrupada, alivianando así el costo computacional. Para favorecer el flujo de información entre grupos, se introdujo una operación de channel shuffle para particionar la salida en sub-bloques.

Para definir un bloque ShuffleNet simple, se tomó un bottleneck, tal como se describe en la Sec. 2.1.2, y se reemplazaron las convoluciones de kernels grandes por su equivalente depth-wise separable. Luego, utilizando la idea descrita previamente, se reemplazaron las operaciones de convolución 1×1 por su versión agrupada como se ve en la Figura 2.10.

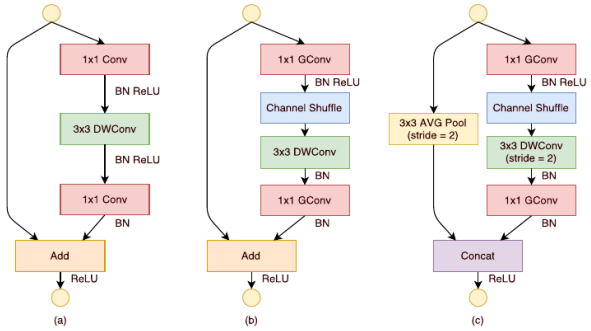


Fig. 2.10: ShuffleNet Units [51]. A izquierda, un bloque bottleneck simplificado al que se reemplaza su convolución tradicional con kernel 3x3 por su equivalente en versión depth-wise. En el centro, un bloque ShuffleNet simplificado que copia la idea anterior pero reemplazando las convoluciones 1×1 en su versión por grupos. Notar que la primera de ellas es utilizada para proyectar la entrada a una dimensión más chica y la última se utiliza para adaptarla al tamaño original. Además, a diferencia de [17] y según lo sugerido en [7] no se aplican no-linealidades como ReLU luego de las depth-wise. Finalmente, a derecha, una variante más completa del bloque ShuffleNet.

ShuffleNet V2

En el trabajo de Ma et al. [33] se delimitaron una serie de buenos principios para el diseño de arquitecturas lightweight y se aplicaron luego para intentar mejorar el el diseño de ShuffleNet V1. Por un lado, se exhibió una cota inferior teórica para el costo de acceso a memoria (MAC) de las convoluciones 1×1, la operación más costosa dentro de los módulos estilo bottleneck de las ShuffleNet units. Esta cota se alcanza cuando la cantidad de canales de entrada y salida son iguales, asumiendo una memoria cache lo suficientemente grande. Experimentalmente corroboraron que a medida que la proporción entre los canales de entrada y salida sea cercaba a uno, el MAC se vuelve más chico.

Por otro lado, como se vio en esta sección, las group convolutions mejoran la complejidad bajando la cantidad de FLOPs respecto a las convoluciones tradicionales por un factor determinado por la cantidad de grupos. En comparación, y dadas restricciones de FLOPs, esto permite lograr mayor profundidad en la red. Sin embargo, el incremento en la cantidad de capas trae aparejado un incremento en MAC (ver ecuación 2 en [33]).

Por último, otros de los puntos que se remarcaron tienen que ver con las operaciones pointwise (por ejemplo RELU, AddTensor, entre otras) que insumen una baja cantidad de FLOPs pero a cambio de un alto MAC.

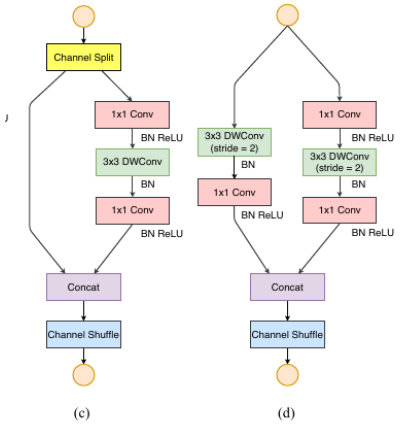


Fig. 2.11: ShuffleNet Unit V2 (izq) y ShuffleNet Unit V2 con spatial downsampling (der). [33]

Motivado por todo lo anterior, introdujeron cambios importantes sobre ShuffleNet V1: Implementaron la operación de channel split que parte el input en dos grupos como se ve en la Figura 2.11. De las dos ramas obtenidas, una permanece como identidad y a la otra se le aplica la misma idea que antes pero reemplazando las group convolution 1×1 para intentar minimizar la cantidad de grupos. Además, se abandonó la noción de bloque bottleneck para favorecer la premisa de tener tamaños de inputs y outputs iguales, modificando todas las convoluciones de la rama de esta forma. Al igual que antes, se mantuvo la operación de channel shuffle al final luego de concatenar los resultados de ambas ramas. Para implementar spatial down-sampling, se modificó el bloque como en el esquema (d) de la Figura 2.11, que a diferencia del bloque común, no hace una separación de canales inicial de manera de poder obtener el doble de canales al concatenar.

La arquitectura final ShuffleNet V2 (ver la Figura 2.12) está formada por etapas que involucran stacks de las nuevas ShuffleNet units. Estos stacks comienzan con la unidad de spatial down-sampling que reduce la dimensión espacial a la mitad y duplica canales. Luego, dependiendo de la etapa, se encadenan aplicaciones de las unidades comunes.

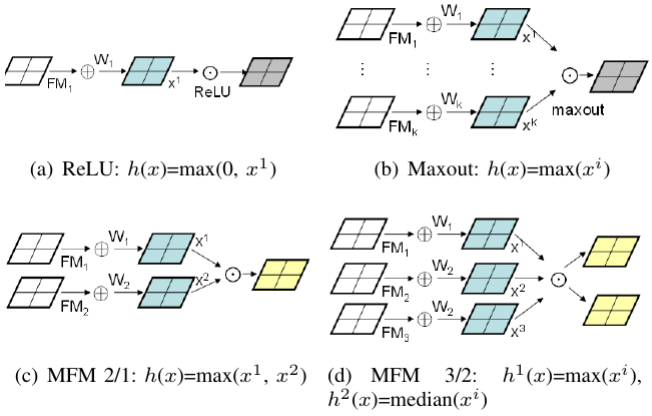
Layer	Output size	KSize	Stride	Repeat	Output channels			
					0.5×	1×	1.5×	2×
Image	224×224				3	3	3	3
Conv1	112×112	3×3	2	1	24	24	24	24
MaxPool	56×56	3×3	2					
Stage2	28×28		2	1	48	116	176	244
	28×28		1	3				
Stage3	14×14		2	1	96	232	352	488
	14×14		1	7				
Stage4	7×7		2	1	192	464	704	976
	7×7		1	3				
Conv5	7×7	1×1	1	1	1024	1024	1024	2048
GlobalPool	1×1	7×7						
FC					1000	1000	1000	1000
FLOPs					41M	146M	299M	591M
# of Weights					1.4M	2.3M	3.5M	7.4M

Fig. 2.12: Descripción de la arquitectura ShuffleNet V2 según las distintas variantes de tamaños (0.5, 1x,1.5x,2x) [33].

2.1.4. LightCNN

En el trabajo de Xiang et al. [49] se optó por una capa de activación que imite la inhibición lateral, un comportamiento estudiado en neurociencia. Dado un estímulo, la neurona más excitada puede mandar una señal de inhibición a sus vecinas para agudizar la señal resultante que va hacia la corteza cerebral, permitiendo además separar señales informativas de otras con mayor ruido.

La operación en cuestión es MFM un caso particular de Maxout (ver la Figura 2.13) que además no requiere de parámetros entrenables, lo cual favorece su aplicación en arquitecturas lightweight. En particular la versión MFM2/1 combina dos feature maps seleccionado el máximo entre las activaciones lugar a lugar.



(a) ReLU: $h(x)$=max(0, x^1) (b) Maxout: $h(x)$=max(x^i)

(c) MFM 2/1: $h(x)$=max(x^1, x^2) (d) MFM 3/2: $h^1(x)$=max(x^i), $h^2(x)$=median(x^i)

Fig. 2.13: Una comparación de diferentes tipos de funciones de activación. (a) ReLU suprime una neurona al aplicar un umbral a las respuestas de magnitud. (b) Maxout con suficientes unidades ocultas hace una aproximación lineal por partes a una función convexa arbitraria. (c) MFM 2/1 suprime una neurona por una relación competitiva. Es el caso más simple de activaciones Maxout. (d) MFM 3/2 activa dos neuronas y suprime una [49].

Por un lado, su objetivo se parece al de la función RELU que inhibe señales que no superen el umbral de cero. Sin embargo, esto puede dar lugar a pérdida de cierta información valiosa en especial en las primeras capas donde tanto valores positivos como negativos deberían ser respetados [49].

El bloque MFM actúa como una forma de selección de features de manera local. Además, como el gradiente de la operación respecto de la neurona que fue suprimida resulta nulo, al hacer backpropagation, Stochastic Gradient Descent solo afecta a las neuronas que fueron excitadas.

Las arquitecturas tipo LightCNN se construyeron con esta operación como pieza fundamental a lo largo de las distintas etapas. Se incorporó la idea de los bloques bottleneck pero utilizando MFM2/1 en lugar de convoluciones 1×1 para reducir dimensionalidad y eliminan las capas de batch-norm. Además, al comienzo de cada etapas se adoptaron convoluciones 1×1 que generan los embeddings duplicando la cantidad de canales. En la Figura 2.14 se puede ver la descripción completa del modelo LightCNN-29, la más compleja de estas arquitecturas.

Type	Filter Size /Stride, Pad	Output Size	#Params
Conv1	$5 \times 5/1, 2$	$128 \times 128 \times 96$	2.4K
MFM1	-	$128 \times 128 \times 48$	-
Pool1	$2 \times 2/2$	$64 \times 64 \times 48$	-
Conv2_x	$\begin{bmatrix} 3 \times 3/1, 1 \\ 3 \times 3/1, 1 \end{bmatrix} \times 1$	$64 \times 64 \times 48$	82K
Conv2a	$1 \times 1/1$	$64 \times 64 \times 96$	4.6K
MFM2a	-	$64 \times 64 \times 48$	-
Conv2	$3 \times 3/1, 1$	$64 \times 64 \times 192$	165K
MFM2	-	$64 \times 64 \times 96$	-
Pool2	$2 \times 2/2$	$32 \times 32 \times 96$	-
Conv3_x	$\begin{bmatrix} 3 \times 3/1, 1 \\ 3 \times 3/1, 1 \end{bmatrix} \times 2$	$32 \times 32 \times 96$	662K
Conv3a	$1 \times 1/1$	$32 \times 32 \times 192$	18K
MFM3a	-	$32 \times 32 \times 96$	-
Conv3	$3 \times 3/1, 1$	$32 \times 32 \times 384$	331K
MFM3	-	$32 \times 32 \times 192$	-
Pool3	$2 \times 2/2$	$16 \times 16 \times 192$	-
Conv4_x	$\begin{bmatrix} 3 \times 3/1, 1 \\ 3 \times 3/1, 1 \end{bmatrix} \times 3$	$16 \times 16 \times 192$	3981K
Conv4a	$1 \times 1/1$	$16 \times 16 \times 384$	73K
MFM4a	-	$16 \times 16 \times 192$	-
Conv4	$3 \times 3/1, 1$	$16 \times 16 \times 256$	442K
MFM4	-	$16 \times 16 \times 128$	-
Conv5_x	$\begin{bmatrix} 3 \times 3/1, 1 \\ 3 \times 3/1, 1 \end{bmatrix} \times 4$	$16 \times 16 \times 128$	2356K
Conv5a	$1 \times 1/1$	$16 \times 16 \times 256$	32K
MFM5a	-	$16 \times 16 \times 128$	-
Conv5	$3 \times 3/1, 1$	$16 \times 16 \times 256$	294K
MFM5	-	$16 \times 16 \times 128$	-
Pool4	$2 \times 2/2$	$8 \times 8 \times 128$	-
fc1	-	512	4,194K
MFM_fc1	-	256	-
Total	-	-	12,637K

Fig. 2.14: Esquema general de la arquitectura LightCNN-29. [49]

2.2. Mecanismos de atención

La incorporación de los mecanismos de atención a las redes convolucionales recibió particular interés a partir de trabajos como el de Hu et al. [20] donde se introdujo la arquitectura SENet que hace uso de los módulos Squeeze and Excitation como bloque fundamental para aprender eficientemente a destacar o suprimir los canales de un feature map. Una vez más, la aparición de variantes proliferó y potenció el uso de la atención en este tipo de redes y particularmente en el problema de FER in-the-wild donde se tiene que lidiar con distintos tipos de condiciones ambientales y de oclusión, siendo sumamente importante que el modelo pueda detectar regiones sobresalientes. Para nuestro trabajo particular, se estudiaron algunas que no introducen un overhead de complejidad excesivo.

2.2.1. Squeeze and Excitation modules

Los módulos Squeeze and Excitation [20] fueron uno de los primeros en introducir mecanismos de atención en redes convolucionales con resultados alentadores. En esencia, esta técnica permite que las redes puedan aprovechar la información de las interdependencias entre canales para aprender a acentuar o suprimir las features de los canales según su relevancia.

Para ello, el módulo se estructuró tomando como base un bloque bottleneck (ver la Figura 2.15). Dado un input feature map I de tamaño $D_w \times D_h \times M$, se aplica una capa de Global Average Pooling, etapa de squeeze, que agrega la información por canal obteniendo un vector de tamaño $1 \times 1 \times C$. Luego le siguen dos capas totalmente conectadas que determinan los pesos por canal, etapa de excitation. Para reducir el costo, estas capas

también actúan reduciendo la dimensionalidad por un factor r dado. Finalmente se aplica una función sigmoidea al vector de pesos resultante que se utiliza para escalar los pesos de I.

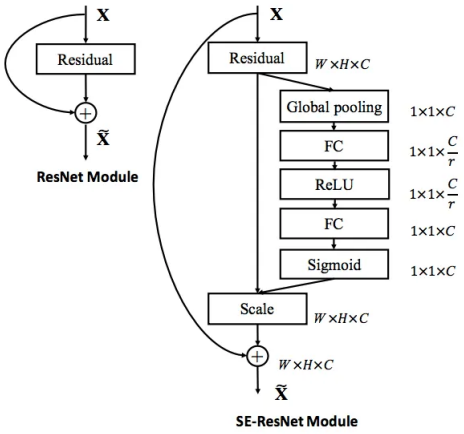


Fig. 2.15: El esquema del módulo Residual original (izq.) y el módulo SEResNet (derecha) que solo agrega $2\frac{M^2}{r}$ parámetros. [20].

2.2.2. Bottleneck Attention Module (BAM)

Los **B**ottleneck **A**ttention **M**odule (BAM) propuestos en el trabajo de Hu et al. [41] sirven como mecanismos de atención generales que pueden ser adaptados fácilmente a distintos tipos de arquitecturas con un overhead de complejidad relativamente bajo.

Una de las debilidades señaladas [41] sobre los módulos Squeeze and Excitation tiene que ver con la falta de extracción de información relevante a la dimensión espacial.

En BAM, para un feature map I de tamaño $D_w \times D_h \times M$ el modulo de atención genera un attention map $M(I)$ de tamaño $D_w \times D_h \times M$ que luego se combina con I. Finalmente el feature map I' resultante se computa como $I' = I + I \otimes M(I)$, donde '$\otimes$' es la multiplicación punto a punto.

Como se describe en la Figura 2.16, el módulo descompone el input en dos ramas, una que resalta features a nivel espacial y otra entre-canales. La rama entre-canales genera un vector de pesos de tamaño $M \times 1 \times 1$ que estima como debería ponderarse la atención por canal. Esto lo hace aplicando primero un Global Average Pool para agregar la información del input feature map, codificando la información global de cada uno. Luego para estimar la atención sobre los canales se utiliza una capa totalmente conectada que reduce la dimensión la cual es finalmente restablecida en la capa de salida luego de aplicar batch norm. La operación puede expresarse como $M_c(I) = BN(FC(GlobalAvgPool(I)))$ siendo FC la aplicación de la capa totalmente conectada y BN la función de batch norm.

Por otro lado, la rama espacial sigue en su diseño la idea de los bloques bottleneck. En primer lugar, se proyecta el input feature map a un espacio de menor dimensión (disminuido por un factor r dado) utilizando convoluciones 1×1 para después aplicar dos convoluciones encadenadas con kernel de 3x3 para agregar información contextual del embedding obtenido. Finalmente se vuelve a reducir la dimensión por un factor de r con otra convolución 1×1 de manera de obtener una salida de tamaño $1 \times D_w \times D_h$ que contiene la información de qué regiones espaciales son las más salientes.

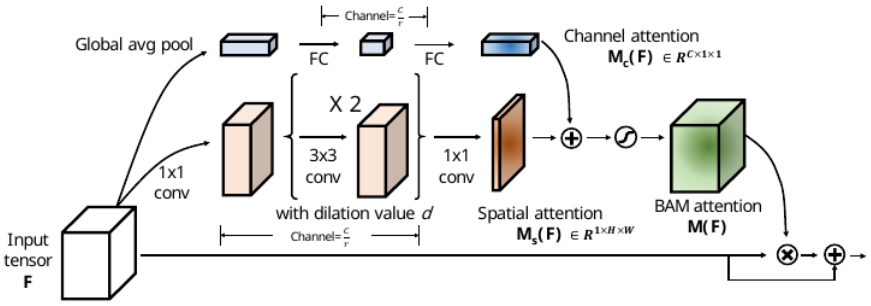


Fig. 2.16: Dado el feature map intermedio F, el módulo BAM calcula el mapa de atención correspondiente $M(F)$ a través de las dos ramas de atención separadas: el canal M_c y el espacial M_s. El valor de dilatación (d) y relación de reducción (r) son los dos hiperparámetros para el módulo [41].

Como los tamaños de ambas ramas no coinciden, para obtener el attention map final se expanden las dimensiones de ambos a $D_w \times D_h \times M$ y se los combina sumándolos y luego aplicando una función sigmoidea.

Finalmente, los resultados de [41] sugieren que el posicionamiento más efectivo de estos módulos es en los bottlenecks de las arquitecturas convolucionales. Sin embargo, cada módulo introduce un overhead de complejidad, medida como cantidad de parámetros, superior al de un módulo Squeeze and Excitation ya que únicamente considerando la rama entre-canales que tiene dos capas totalmente conectadas se llega a $2\frac{M^2}{r}$ parámetros.

2.2.3. Efficient Channel Attention Module (ECA)

En el trabajo propuesto por Wang et al. [48] se hace una observación clave sobre los módulos Squeeze And Excitation: la idea de computar los pesos de los canales proyectados sobre un feature map de menor dimensión que después vuelve a mapearse al espacio original hace que la correspondencia de los pesos y los canales termine siendo indirecta. Sin embargo, esta reducción es necesaria si se quiere mantener la restricción de complejidad.

La alternativa ensayada buscó considerar la atención de manera local aprovechando group convolutions (revisadas en la Sec. 2.1.3). Sin embargo, por lo visto en la Sec. 2.1.3, usar demasiados grupos podría incrementar el MAC.

Con esta idea detrás, los módulos ECA utilizan una convolución 1D (no confundir con 1×1) con un kernel de tamaño k elegido de manera adaptativa. Esta capa es utilizada para reemplazar a las capas totalmente conectadas de los módulos Squeeze and Excitation.

Como se puede ver en la Figura 2.17, dado un input feature map I, se sigue con la misma aplicación de GAP que tenían los módulos Squeeze and Excitation obteniendo un vector de tamaño $1 \times 1 \times M$. Luego, con la convolución 1D se desliza el kernel por la dimensión de los canales y se aplica una función sigmoidea. Estas últimas dos operación preservan la dimensión, por lo tanto el resultado puede utilizarse directamente sobre I.

Según los resultados del trabajo [48], el desempeño con respecto los módulos Squeeze and Excitation es notablemente mejor y más aún, el costo de la versión más simple de ECA solo involucra k parámetros entrenables.

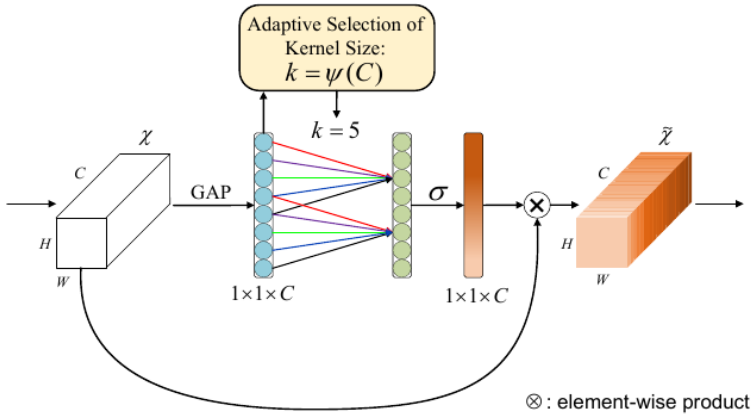


Fig. 2.17: Diagrama de ECA. Dadas los features obtenidos por Global Average Pooling (GAP), ECA genera pesos de canal realizando una convolución 1D rápida de tamaño k, donde k se determina de manera adaptativa a través de un mapeo $\psi(\cdot)$ de la dimensión C del canal. [48]

2.3. Inconsistencias de etiquetas en FER

Como se mencionó en la Sec. 1.2, uno de los desafíos de entrenar modelos con grandes volúmenes de datos tiene que ver con la presencia de inconsistencias en las etiquetas, es decir incertezas en las etiquetas del ground-truth de un dataset, ya que puede sesgar el aprendizaje, potencialmente disminuyendo el rendimiento final. Esto representa un problema relevante en métodos de Deep Learning supervisados ya que la necesidad de tener disponible grandes cantidades de datos etiquetados precisamente es muy costoso.

En particular para el caso de FER con anotaciones de categorías dadas por las emociones básicas, se juzga cual es la emoción subyacente que generó la expresión facial retratada en el rostro. En este caso, una de las principales fuentes de inconsistencias tiene que ver con la naturaleza ambigua de las expresiones faciales en sí. Según Shariff et al. [42], solo existen una cantidad reducida de emociones básicas y el resto ocurren como combinaciones de éstas con cierta intensidad. Por esta razón, utilizar una única etiqueta de emoción (SLL, Single Label Learning) puede incorporar un grado no menor de ambigüedad ya que no permite capturar toda esta expresividad, más aún en un contexto in-the-wild.

A diferencia de lo anterior, donde las inconsistencias pueden surgir por la ambigüedad intrínseca de una expresión, existe un segundo tipo de inconsistencia que afecta a la categorización de emociones que está relacionada con el sesgo de anotación dado por el origen, contexto cultural y habilidad de los anotadores. Esto es aún más delicado cuando las etiquetas son anotadas a partir de resultados de motores de búsqueda, crowd-sourcing entre otros (ver la Figura 2.18).

Para poder afrontar este problema existen técnicas más allá de la revisión manual de anotaciones que resulta en muchas ocasiones prohibitiva por el costo de hacerlo con volúmenes gigantes de datos. Algunos acercamientos [13] consisten en adaptar el aprendizaje del clasificador para ser más tolerante al ruido, por ejemplo a través una función de pérdida más robusta. Otros se basan en mejorar la calidad de los datos a través de algún tipo de filtrado como etapa de preprocesamiento. Si bien este tipo de técnicas no suelen ser intensivas en costo computacional, se corre el riesgo de dejar de lado información relevante al dataset.

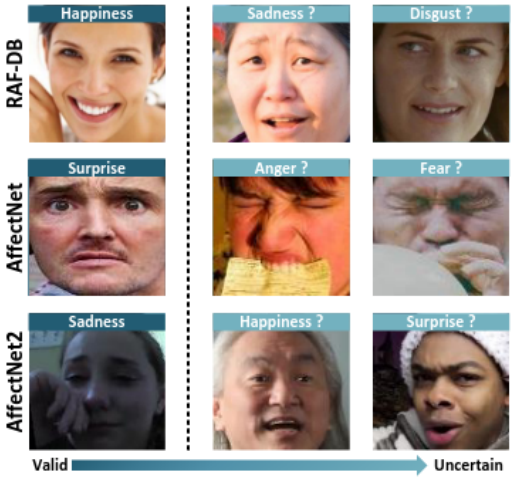


Fig. 2.18: Distintos datasets contienen expresiones con diferentes grado de ambigüedad que pueden generar inconsistencias según como sean percibidas las expresiones por los anotadores.

2.3.1. Bootstrapping para etiquetas inconsistentes

La arquitectura LightCNN [49] tiene como otro de sus objetivo de diseño ser robusta ante datasets con etiquetas inconsistentes.

El acercamiento empleado utiliza una técnica de Bootstrapping, o self-learning, que puede resumirse de la siguiente manera: (1) Inicialmente se entrena la red sobre el dataset CASIA-WebFace y luego se hace finetunning con el dataset MS-CELEB que contiene etiquetas ruidosas. Dado una instancia del rostro de una persona, la red aprende a estimar la identidad entre una gran cantidad de sujetos. (2) Se aplica el clasificador sobre el dataset de MS-CELEB, solo aquellas instancias cuya identidad coincida con el ground-truth y lo hagan con una probabilidad mayor a un umbral de certeza serán mantenidas en el dataset. De esta manera se define un nuevo dataset MS-CELEB-R que contiene solo las instancias con más confianza. (3) Se reentrena el modelo desde cero utilizando el dataset MS-CELEB-R que debería ser menos ruidoso. (4) Se aplica el nuevo modelo sobre el dataset MS-CELEB aceptando aquellas instancias que coincidan en el ground-truth o bien no lo hagan pero la estimación sea superior a un umbral de confianza. De esta manera, se obtiene un nuevo dataset MS-CELEB limpio. Finalmente, el modelo final vuelve a entrenarse desde cero utilizando el dataset limpio, y según [49] los resultados para este escenario superan claramente al entrenamiento con etiquetas ruidosas.

2.3.2. Label Distribution Learning (LDL)

Para poder afrontar ambigüedades intrínsecas a la naturaleza de las emociones, el uso de Single Label Learning no permite más que la asociación a una única emoción categórica.

Si consideramos a cada emoción básica como una etiqueta, Multi Label Learning permite describir a cada imagen a partir de un conjunto de etiquetas definidas como relevantes. El problema es que se pierde noción de la intensidad con la que aparece cada una.

Utilizando la distribución de emociones, presentada en Zhou et al. [55], se puede asociar a cada imagen un vector como etiqueta, donde cada componente es la intensidad que

aporta una determinada emoción básica a la expresión facial subyacente. Normalizando cada intensidad al rango $[0, 1]$ y haciendo que la suma de las componentes sea 1, se obtiene la denominada distribución de la emoción para la imagen dada. Esta técnica, denominada Label Distribution Learning [54], debería, entonces, ser más robusta a las inconsistencias que surgen en FER. De hecho, se consiguieron mejores resultados para el dataset JAFFE [55]. Este dataset, obtenido en condiciones controladas de laboratorio, además cuenta con votos por categoría de emoción, permitiendo estimar una distribución de emociones y utilizarlas como para el entrenamiento.

El problema de este acercamiento es que, justamente, tener disponible la distribución de emociones no es algo habitual en los datasets usados en FER. Una alternativa estudiada en Gato et al. [14] es modelar la distribución de emociones como una distribución normal basada en la información provista por el label categórico.

Otro enfoque interesante es el de construir un modelo que pueda aprender a estimar las distribuciones de emociones. En el trabajo de Zhao et al. [52], se propone la arquitectura EfficientFace, un clasificador que se entrena con la ayuda de un modelo generado auxiliar que aprende previamente a estimar las distribuciones de emociones. Este componente es entrenado independientemente y solo actúa como una suerte de generador de ground-truth al momento de entrenar el modelo lightweight. En otras palabras, para computar la función de pérdida del clasificador lightweight se utiliza el error cometido entre su estimación y la salida del generador (soft labels). Este acercamiento puede enmarcarse como una estrategia de knowledge distilattion [18], donde se utiliza una red de mayor complejidad que puede aprender mejores representaciones para luego entrenar otra red más chica.

Con un enfoque similar En Chen et al. [5] se propone entrenar un clasificador guiado por la topología del espacio de etiquetas de una tarea auxiliar como Landmark detection y Action Unit Recognition. A partir de esto, propone computar una función de perdida auxiliar para que dada una instancia del espacio de imágenes de expresiones faciales, se tenga en cuenta la distribución de emociones estimada de los vecinos sobre el espacio de etiquetas de la tarea auxiliar. A su vez la distribución de emoción de cada vecino es pesada en función de la distancia a la instancia original en el espacio auxiliar. Esta técnica, entonces, propone aprovechar la topología de etiquetas que, en principio, deberían ser más descriptivas y estar sujetas a menos ambigüedades. Sin embargo, una vez más, la disponibilidad de etiquetas como Action Units en datasets de FER es limitada por lo que se debe recaer en el uso de métodos Ad hoc para conseguir estas anotaciones.

3. METODOLOGÍA

3.1. El framework propuesto

Habiendo repasado la teoría detrás de algunos diseños ejemplares de arquitecturas lightweihgt y habiendo considerado las distintas técnicas que suelen acompañar el entrenamiento de las mismas, este trabajo considera dos frameworks del estado del arte para su estudio: EfficientFace [52] y CERN [15] cuyas arquitecturas parten de las ya mencionadas ShuffleNet v2 (Sec. 2.1.3) y LightCNN29 (Sec. 2.1.4).

En el fondo, ambas tienen en común varias de las ideas centrales que se discutieron en la revisión de este trabajo: El uso de arquitecturas diseñadas eficientemente y la incorporación de algún mecanismo de atención para enfatizar regiones relevantes para FER in-the-wild. También comparten estrategias para capturar información del contexto local y global e incorporarla de alguna manera a los feature maps.

Además ambas utilizan la técnica de Transfer Learning [56], cuyo objetivo general es extraer el conocimiento aprendido por el modelo sobre una tarea y dominio original y aplicarlo en una tarea y dominio objetivo. En este caso particular, la tarea de origen es la del reconocimiento facial a partir del dataset MS-CELEB [16] como dominio ya que en esencia, los feature maps de las primeras capas deberían estar relacionados con rasgos elementales del rostro humano y por lo tanto, es un punto común con FER.

En el caso de EfficientFace simplemente se inicializan los pesos a partir del modelo entrenado sobre la tarea original, exceptuando los de la última capa totalmente conectada ya que estos estarán íntimamente relacionados con la tarea objetivo. Notar que esta técnica también puede utilizarse congelando las capas iniciales que deberían estar asociadas a features de más bajo nivel y continuar actualizando únicamente los pesos de las últimas. Sin embargo, dado que cada dataset cuenta una gran cantidad de instancias no debería ser una consideración tan crítica. Se comprobó que esto ocurría así en las etapas preliminares de este trabajo y que, efectivamente, era conveniente no congelar capas.

En CERN se emplea un acercamiento ligeramente distinto, ya que en primer lugar se realiza una adaptación de la red LigthCNN, reteniendo las capas de más bajo nivel y reemplazando el resto por otras que implementan el procesamiento más fino con técnicas de atención y división en ensamble de clasificadores (ver Figura 3.2). Para el entrenamiento se utilizan dos learning rates: el más chico para intentar preservar el aprendizaje original asociado las features bajo nivel que deberían ser compartidos por ambas tareas (reconocimiento facial y FER), y el mayor para entrenar los nuevos módulos introducidos para adaptar la red a la tarea de FER in-the-wild.

Para intentar mitigar el impacto de la inconsistencia y ambigüedad de las etiquetas en grandes volúmenes de datos anotados permitiendo un entrenamiento supervisado más robusto, se considera el uso de Label Distribution Learning (LDL) [54] [14]. Conseguir estrategias de entrenamiento más refinadas que permitan extraer mayor desempeño merecen la atención en un contexto donde el incremento de la complejidad de la red esté fuertemente penalizado, como es el caso de las arquitecturas lightweight. El problema principal es que la mayoría de los datasets del dominio cuenta con etiquetas categóricas en lugar de distribuciones de emociones. Por lo tanto, se propone seguir la idea propuesta en el trabajo de EfficientFace [52] y construir un generador de distribuciones de emociones o

LDG (por su sigla en inglés, **L**abel **D**istribution **G**enerator) para luego implementar la estrategia de LDL sobre las arquitecturas de estudio. Concretamente, el modelo utilizado para el LDG es una ResNet-50 [17] que será entrenado para cada uno de los datasets que incluye nuestro benchmark. Al igual que los modelos de EfficientFace y CERN, en todo entrenamiento se hace uso de transfer learning para inicializar los pesos del LDG a partir de la tarea de Reconocimiento Facial sobre el dataset MS-CELEB. Una vez entrenado, el modelo permanece congelado y se lo utiliza en EfficientFace y CERN para consultar el ground truth de cada instancia durante el entrenamiento con Label Distribution Learning. De esta forma, el LDG permanece haciendo únicamente inferencia para poder asociarle una distribución de emoción estimada a cada instancia como se puede ver esquematizado en la Figura 3.1.

Al utilizar LDL estamos transformando las etiquetas categóricas de emociones en distribuciones que tienen una única activación. Como ya se discutió en la Sec. 2, la categorización de expresiones faciales en emociones es claramente ambigua y esto podría impactar en la utilidad de esta técnica, por lo que el uso de anotaciones descriptivas auxiliares como landmarks y Action units serían una forma de disminuir el grado de inconsistencia y ambigüedad.

A partir de esto, en este trabajo se propone incorporar al entrenamiento el uso de la topología del espacio de etiquetas auxiliares de la tarea de Action Unit Recognition de acuerdo a lo estudiado en Chen et al. [5]. Notar que esto no supone ningún aumento en la complejidad de la red ni tampoco afecta la velocidad de inferencia. Sin embargo, podría encarecer el tiempo de entrenamiento y el trabajo de pre-procesamiento asociado a cada dataset. De esta manera, pensamos dos alternativas: una indirecta, que incorpora esta idea en la función de pérdida del LDG para luego potenciar el entrenamiento de EfficientFace y CERN y otra directa, que lo hace únicamente sobre la función de pérdida de las redes lightweight consideradas.

La limitación de elegir Action Unit Recognition como tarea auxiliar es que la disponibilidad de datasets etiquetados no es demasiado alta, entre otras cosas porque requiere de cierta calificación y porque la tarea demanda tiempo considerable, por lo tanto se exploran las anotaciones generadas por dos herramientas del estado del arte: OpenFace[1] y MLCR [39] que usan técnicas de Machine Learning clásicas y end-to-end learning respectivamente.

Finalmente, para evaluar el desempeño de clasificación de los modelos, utilizamos principalmente Accuracy rate (tasa de aciertos) que es una medida habitual para estos problemas. En algunos casos particulares también será relevante analizar el desempeño por emociones, por lo que introduciremos matrices de confusión. Tomaremos como benchmark a un conjunto de datasets in-the-wild: CAER-S, RAF-DB, AffectNet, FER+.

Como se dijo previamente, parte de la idea de Label Distribution Learning es mejorar la robustez a las inconsistencias de las etiquetas. Concretamente, para evaluar esta tolerancia, se realizan modificaciones a algunos de los datasets para generar distintos niveles de ruido sintético.

Para comparar el costo computacional de las redes se tiene en cuenta que la elección de los FLOPs como medida puede ser engañosa. Esto tiene que ver con varias cosas: los FLOPs son agnósticos al costo de acceso a memoria y al grado de paralelismo de las operaciones y además dependiendo de la plataforma hay ciertos tipos de operaciones que están más optimizadas [33]. Por lo tanto, arquitecturas con FLOPs comparables pueden tener velocidades distintas razón por la cual utiliza como medida del costo computacional los por segundo (para el caso GPU) o bien imágenes por segundo en CPU.

3.1.1. Arquitecturas de estudio

EfficientFace

A partir de lo discutido en secciones previas, se introduce el framework completo de EfficientFace [52] que puede verse en la Figura 3.1.

Partiendo de una arquitectura ShuffleNet V2, se hace uso de estrategias de extracción de features más robustas que tengan en cuenta el contexto local y global para lidiar con el tradeoff entre capacidad de predicción y complejidad. En concreto, para adaptarse a escenarios in-the-wild, la arquitectura agrega en sus etapas iniciales, donde debería ser más fácil la extracción de features locales, un Local-Feature Extractor que aprende features de regiones locales partiendo la imagen en cuatro parches y empleando dos depthwise-convolutions encadenadas. El feature map resultante de este módulo luego es combinado con el proveniente del Channel-Spatial Modulator, que básicamente implementa el mecanismo de atención tipo BAM como se describió en la Sec. 2.2.2 sobre el feature-map resultante de la etapa 2. Finalmente, con la salida de estos dos módulos sea deberían computar features locales-globales que luego siguen su curso por el resto de las etapas.

Según el estudio de ablación realizado en [52], estas modificaciones suponen mejoras de entre 1.34 % y 1.16 % sin dañar la complejidad considerablemente. Con respecto a la arquitectura base de ShuffleNet V2, la cantidad de FLOPs crece un 2 % y la cantidad de parámetros un 1.5 %

La otra caracterización que se le dio a esta arquitectura fue la introducción de alguna mejora para el entrenamiento en el contexto in-the-wild. Esto se logra a través de Label Distribution Learning (LDL) implementado con un Label Distribution Generator (LDG) como se mencionó en [54], bajo el supuesto de que las expresiones humanas ocurren como combinaciones de emociones básicas y dado que podría ayudar a alivianar el problema de las inconsistencias de las anotaciones.

Como se puede ver en la Figura 3.1 el generador de emociones, representado por una ResNet50, actúa como módulo auxiliar en el entrenamiento de EfficientFace. Dado un input, el mismo realiza un forward-pass por esta red y por el generador. Cada red estimará la distribución de emociones para la instancia, con la particularidad de que la del generador es considerada la del ground-truth. Notar que, entonces, los únicos parámetros que se modifican durante la etapa de back-propagation, luego de computar la función de perdida, son los de EfficientFace.

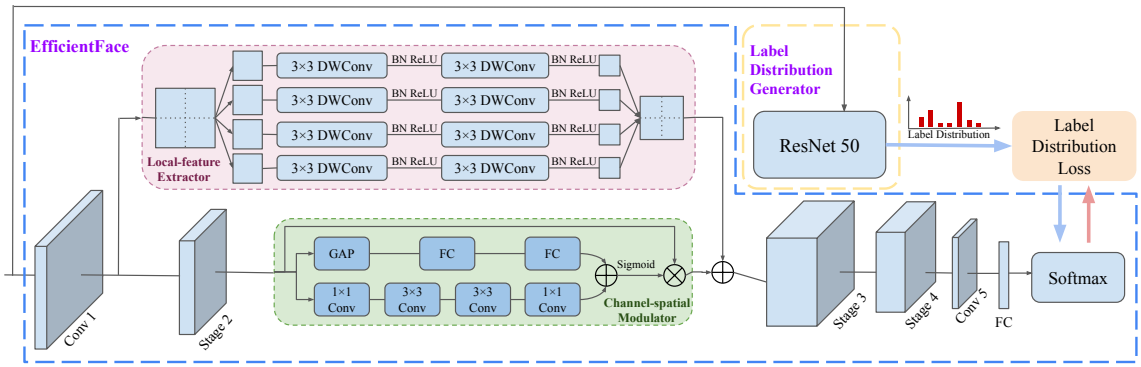


Fig. 3.1: Esquema completo del framework EfficientFace [52].

CERN

Al igual que en EfficientFace, el diseño de la red CERN [15] parte de una de las arquitecturas lightweight del estado del arte, LightCNN-29 [49] en una de sus versionas más completas (ver la fig Figura 2.14). Sin embargo, esta variante cuenta con aproximadamente 12M parámetros que podrían ser excesivos para este contexto. Interesantemente, la propuesta en CERN es utilizar las primeras tres de las cinco etapas de la red original (específicamente hasta la capa Pool3) argumentando que es allí donde se capturan los features más bajo nivel de las caras, como bordes, dejando de lado las capas finales que estarían asociadas a rasgos más refinados y específicos asociado a la identidad de la persona.

Como mecanismo de atención en CERN se utiliza ECA [48] como se lo describió en la Sec. 2.2.3 ya que supone un buen balance entre mejora del desempeño y complejidad adicional. De hecho, los módulos introducidos no agregan más de 4000 parámetros en CERN.

El framework completo puede observarse en la Figura 3.2 donde se distinguen varias etapas: Inicialmente el input es tomado por la versión recortada de LigthCNN29 que forma la red backbone de CERN.

A la salida se tiene un feature map de tamaño $16 \times 16 \times 192$ que aún contiene información relacionada con regiones donde podría haber, por ejemplo, oclusión dado el contexto in-the-wild. Para ello, sujeto a la restricción de complejidad, se toman regiones locales subdividiendo al volumen en cuatro pedazos de $8 \times 8 \times 192$. A partir de estos pedazos se genera un ensamble de clasificadores, cada uno de los cuales primero aplica un módulo de atención ECA seguido de una aplicación de Global Average Pooling y dos capas totalmente conectadas, siendo la última la que tiene las estimaciones de las categorías. De esta forma, cada uno aporta de forma independiente una componente para la función de pérdida. Para computar la estimación final, se toma el promedio de los cinco clasificadores (se agrega un clasificador que simplemente recibe el feature map global directamente de la salida de LightCNN).

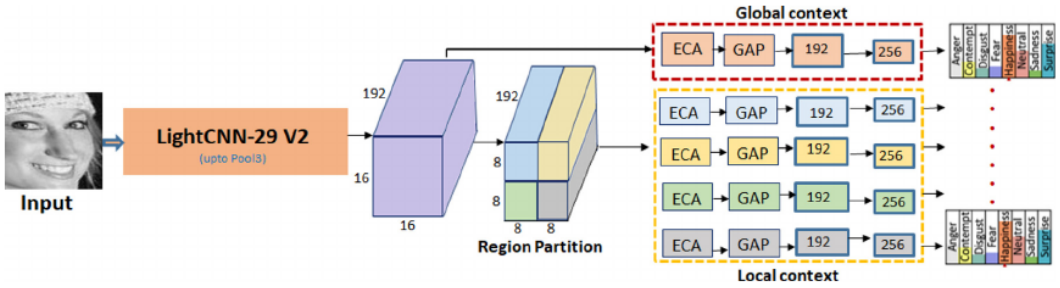


Fig. 3.2: Arquitectura completa CERN que puede verse en dos etapas: La primera que utiliza los primeros tres niveles de LightCNN29. La salida de esta red se subdivide en regiones, cada una de las cuales se utiliza para alimentar un clasificador del ensamble distinto. Cada clasificador opera, básicamente, seleccionado features salientes de la región que recibe mediante el módulo de atención ECA.

3.2. Formulación del problema de LDL

En este trabajo implementaremos Label Distribution Learning como técnica de entrenamiento para encontrar un modelo paramétrico $f(x|\theta)$ que mapee una instancia x_i de un espacio de imágenes de expresiones faciales $X \subseteq \mathbb{R}^{HxW}$ a distribuciones de emociones $D \subseteq \mathbb{R}^c$ sobre un conjunto de etiquetas $Y = \{y_1, y_2, \dots y_c\}$ siendo c la cantidad de etiquetas posibles de emociones.

Usaremos $d_{x_i}^{y_i}$ para denotar la intensidad de la emoción $y_i \in Y$ de la expresión facial representada por $x_i \in X$. Definiremos d_{x_i} la **distribución de emociones** de una instancia cuando $d_{x_i}^{y_i} \in [0, 1]$, $\sum_{y \in Y} d_{x_i}^{y} = 1$.

Sea un conjunto de entrenamiento $S = \{(x_1, l_1), (x_2, l_2), \dots, (x_n, l_n)\}$ con $x_i \in X$, $l_i \in L$ siendo L el conjunto de etiquetas tal que $l_i = (d_{x_i}^{y_1}, d_{x_i}^{y_2}, \dots, d_{x_i}^{y_c})$ es la distribución de emociones anotada para x_i que se utiliza como ground truth (i.e d_{x_i}). El objetivo es aprender a través de $f(x|\theta)$ la función de probabilidad condicional $p(y|x)$ a partir de S donde $x \in \chi$, $y \in Y$.

3.2.1. Entrenamiento del Label Distribution Generator

Definimos un Label Distribution Generator (LDG) en base a la arquitectura ResNet-50 para aprender $f(x|\theta)$ que dada una instancia del espacio de imágenes genera su distribución de emoción. Al no tener disponibles etiquetas de distribución de emociones en los datasets que utilizamos, el LDG aprende a generar distribuciones a partir de SLL usando las etiquetas categóricas de las siete emociones básicas.

Luego se debería considerar la forma de medir una noción de similaridad entre dos distribuciones para determinar la función de pérdida. En este sentido, se podrían nombrar la distancia Euclidiana, la divergencia de Jeferry, la divergencia de Kullback-Leibler entre otras. Sin embargo, en este trabajo, y al igual que en [52] adoptaremos Cross-Entropy como función de pérdida para entrenar al LDG sobre el conjunto S. Es decir:

$$\mathcal{L}_{ldg} = -\frac{1}{N \times c} \sum_{i=0}^{N-1} \sum_{j=0}^{c-1} d_{x_i}^{y_j} log(\widehat{d_{x_i}^{y_i}}) \tag{3.1}$$

donde $N = |S|$ y $\widehat{d_{x_i}} = (\widehat{d_{x_i}^{y_1}}, \widehat{d_{x_i}^{y_2}}, \dots, \widehat{d_{x_i}^{y_c}})$ es la predicción de la distribución de emociones para x_i.

En particular al tener disponible únicamente las emociones categóricas como etiquetas, para determinar L asumiremos que la distribución d_{x_i} se puede aproximar razonablemente bien considerando un vector lógico (one-hot) a partir de la etiqueta de la categoría que viene anotada. En otras palabras, entrenamos el LDG con hard labels y definimos el conjunto L de manera que:

$$\forall i \in \{0 \dots N-1\} \; \exists! \; k \in \{0 \dots c-1\} \; d_{x_i}^{y_k} \neq 0 \; \wedge \sum_{j=0}^{c-1} d_{x_i}^{y_j} = 1. \tag{3.2}$$

3.2.2. Entrenamiento de EfficientFace LDL

Como primer acercamiento, el modelo de EfficientFace en su versión Single Label Learning se entrena por medio de Cross-Entropy, de la misma manera que el LDG tal como en [52]. Es decir, siguiendo la misma función de pérdida que en la ecuación 3.1 y teniendo un

dataset construido con etiquetas que cumplen la ecuación 3.2 (pues solo se tiene disponible la anotación de emoción categórica).

Habiendo entrenado el LDG de manera independiente, se lo mantiene congelado para ayudar en el entrenamiento de EfficientFace. Se puede pensar que el LDG funciona como generador de ground truth, mapeando cada instancia de entrenamiento a una distribución de emoción de estimada.

Para computarla, tomamos el vector de features $v_{x_i} = (v_{x_i}^{y_1}, v_{x_i}^{y_2}, \ldots, v_{x_i}^{y_c})$ de la salida de la última capa FC del LDG y le aplicamos una función softmax. Esto termina por generar una distribución de emociones. A partir de lo anterior, se puede definir un conjunto de entrenamiento $S' = \{(x_1, l'_1), (x_2, l'_2), \ldots, (x_n, l'_n)\}$ con $x_i \in X$, $l'_i \in L'$ siendo L' el conjunto de distribución de emociones anotadas tal que $l'_i = (d_{x_i}^{y_1\prime}, d_{x_i}^{y_2\prime}, \ldots, d_{x_i}^{y_c\prime})$ y ahora:

$$d_{x_i}^{y_i\prime} = \frac{exp(v_{x_i}^{y_i})}{\sum_{j=1}^{c} exp(v_{x_i}^{y_j})} \tag{3.3}$$

Finalmente, habiendo conseguido S' con etiquetas que deberían ser más expresivas, soft labels, también utilizamos Cross-Entropy como se definió en la ecuación 3.1 para este caso y al igual que [52].

Para hacer inferencia sobre una instancia, se considera v_{x_i} el vector de features de la última capa FC (Fully Connected) de la red, se le aplica una función softmax y se toma el índice del máximo en este vector de probabilidades como el índice de la emoción asociada.

3.2.3. Entrenamiento de CERN LDL

Para entrenar CERN Single Label Learning, en [15] utilizan la función de pérdida Cross-Entropy aplicada a cada una de los clasificadores por región como se describe en la Sec. 3.1.1. Luego la pérdida total se computa como la suma de estas partes.

Una sutileza sobre este cálculo es que, en particular, para cada imagen del dataset, se hace un forward-pass para ésta y su versión espejada horizontalmente. Por lo tanto, cada clasificador obtiene dos estimaciones las cuales son luego agregadas tomando el promedio para definir una única probabilidad por categoría. En definitiva, sobre cada clasificador se acumula, como se dijo antes, la pérdida respecto a ambas versiones de la imagen.

Para entrenar CERN con Label Distribution Learning se incorpora el Label Distribution Generator modificando el dataset de igual forma que en el caso de EfficientFace LDL. Notar que entonces cada uno de los cinco clasificadores aporta su pérdida, computada contra etiquetas de distribución de emociones para computar la función de pérdida total.

3.2.4. Entrenamiento con LDL y AUs

Siguiendo la idea de Chen et al. [5], se propone utilizar el espacio de labels de una tarea auxiliar para utilizar información sobre la topología subyacente. Para esto consideramos un conjunto de entrenamiento $S = \{(x_1, l_1, w_1), (x_2, l_2, w_2), \ldots, (x_n, l_n, w_n)\}$ donde $w_i \in W$ siendo $W \subseteq \mathbb{R}^k$ el conjunto de etiquetas asociados a Action Unit Recognition, la tarea auxiliar. En este caso particular, cada etiqueta es un vector que representa la intensidad de activación de cada Action Unit. Inicialmente, consideramos L como en 3.2.

Dada una instancia del espacio de imágenes, el objetivo es guiar el aprendizaje de la red de manera que se considere la vecindad de la instancia pero sobre el espacio de la tarea auxiliar. Para esto, como en [5] se asume cierta noción de suavidad: imágenes que resulten cercanas en este espacio de etiquetas de AUs tienen mayor probabilidad de

tener distribuciones de emociones similares. De esta forma vamos a tener en cuenta las distribuciones de emociones de los vecinos ponderada por la importancia que la podremos medir tomando alguna noción de distancia. Al considerar una instancia sujeta a ruido en su anotación de emoción durante el entrenamiento, la información de la vecindad en el espacio auxiliar debería ayudar a contrarrestar el impacto, de otra forma, negativo en el aprendizaje.

Para introducir esta idea en cada framework, inicialmente consideramos hacerlo de manera indirecta a través del módulo del LDG con el objetivo de lograr un mejor desempeño que luego pueda aprovecharse al momento del entrenamiento de las arquitecturas ligthweight.

Para ello, vamos a descomponer la función de pérdida en dos términos, uno que, como antes, compute la pérdida contra la distribución de emociones determinada a partir del ground truth (el conjunto L) y otra que capture la pérdida respecto a la distribución de emociones estimada de cada uno de sus vecinos.

$$\mathcal{L}_{ldg-aus}(\theta) = \mathcal{L}(\theta) + \lambda\Omega(\theta). \tag{3.4}$$

A diferencia de [6], que utiliza Divergencia de Kullback-Leibler para $\mathcal{L}$, proponemos en su lugar la función de pérdida Cross-Entropy, ya que empíricamente notamos que se comporta mejor al entrenar el modelo.

Para Ω, en este caso sí, vamos a tomar la divergencia de Kullback-Leibler para medir la distancia ponderada entre las distribuciones de la vecindad de una instancia. Por lo tanto, si consideramos $f(x_i|\theta)$ como el vector softmax de salida del LDG con parámetros θ para la entrada x_i (tal como en la ecuación 3.3) y $D_{KL}(P, Q)$ la Divergencia de Kullback-Leibler entre las distribuciones P y Q, definimos:

$$\Omega(f(x|\theta)) = \sum_{ij} a_{ij} D_{KL}(f(x_j)||f(x_i)) \tag{3.5}$$

Para computar el coeficiente de similitud local a_{ij} siguiendo la misma idea de [6], utilizamos una noción de distancia tal que:

$$a_{ij} = \begin{cases} \exp\left(-\frac{d(w_i,w_j)}{\sigma^2}\right) & \text{si } w_j \in N(i) \\ 0 & \text{en otro caso} \end{cases} \tag{3.6}$$

donde $N(i)$ representa el conjunto de los K vecinos más cercanos de x_i en el espacio de etiquetas de la tarea auxiliar de Action Unit Recognition y $d(w_i, w_j)$ es una función de distancia sobre W. De esa forma queda definida la estrategia indirecta a partir de la función de perdida recién construida para la variante denominada LDG-AUs.

Alternativamente, se propone el método directo que consiste, simplemente, en agregarle a la función de pérdida de EfficientFace y de CERN el termino asociado a la función ω utilizando el LDG sin modificación alguna. Para este caso el dataset considerado es $S = \{(x_1, l'_1, w_1), (x_2, l'_2, w_2), \ldots, (x_n, l'_n, w_n)\}$ tomando W como antes y L' el conjunto de soft labels generados por el LDG. En la Figura 3.4 y la Figura 3.3 se describen los esquemas generales de ambas propuestas.

Volviendo sobre la ecuación 3.6, será necesario definir el vecindario de una imagen en el espacio de labels de Action Units utilizando alguna noción de distancia. Para esto, se puede construir un KNNG, que es un grafo en donde un vértice p tiene un eje hacia q si la distancia entre p y q se encuentra entre las k más pequeñas de todo el conjunto de distancias entre

p y otros vértices del conjunto de vértices total del grafo. El problema de construir KNNG exactos con fuerza bruta es su complejidad $O(n^2)$, algo que para para datasets demasiado grandes podría suponer un problema. Hay varias métodos para aproximar los grafos de vecinos como SW-GRAPH [36] que pueden reducir drásticamente el costo de construcción manteniendo buenos tiempos de consulta. En particular, utilizamos la herramienta NGT[1] como en [6].

En concreto, se computa de manera offline un índice general para cada dataset en estudio que dada la etiqueta de Action Units w_i de una instancia x_i permite consultar eficientemente los k vecinos más cercanos (en su forma aproximada) junto con las distancias respectivas. Estas distancias se tomaron como la norma euclídea, luego de probar con otras como similaridad coseno, distancia de Hamming sin conseguir mejores resultados. Luego, al comenzar el entrenamiento se puede construir por única vez una lista de similaridad para cada instancia que almacene el índice y el coeficiente de similaridad local a_{ij} que sea no nulos (es decir, únicamente para sus K vecinos más cercanos).

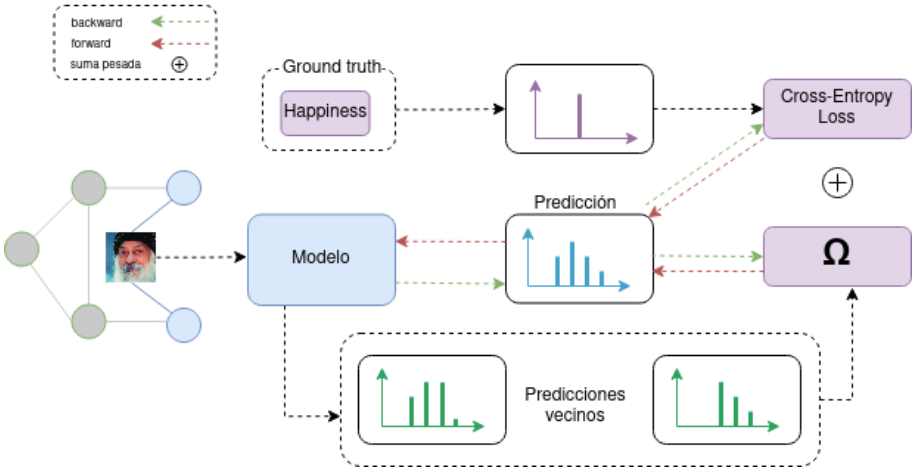


Fig. 3.3: Esquema para LDL AUS para el caso indirecto, donde modelo hace referencia al LDG en este caso una ResNet50. Luego, en caso de mejorar, se lo utiliza como antes para tratar de mejorar el entrenamiento de EfficientFace.

3.3. Anotación de Action Units

Las Action Units (AUs) codifican movimientos de grupos de músculos faciales según el Facial Action Coding System (FACS) [12]. En particular la última revisión del 2002 determina 9 Action Units en la parte superior de la cara y 18 en la parte inferior que pueden observarse en la Figura 3.5. Además de esto, cuenta con 14 relacionadas con la posición y movimiento de la cabeza, 9 relacionadas con la posición y movimiento de los ojos. También se consideran otras para acciones misceláneas.

Las AUs pueden ocurrir de manera aditiva. Es decir, la presencia de cada una puede activarse de manera independiente. Como un ejemplo, la AU 1+2 suelen ocurrir en la emoción de sorpresa describiendo las arrugas horizontales que aparecen en la frente producto de elevar la esquina interna (AU 1) y externa (AU 2) de las cejas. Los cambios reflejados

[1] https://github.com/yahoojapan/NGT

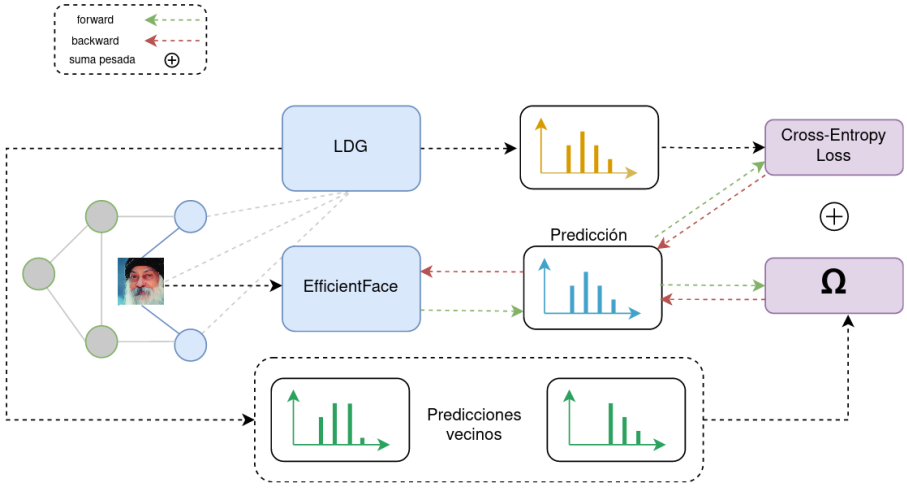


Fig. 3.4: Esquema para LDL AUS para el caso directo donde se hace agrega el término adicional Ω en la loss de EfficientFace y el LDG permanece sin cambios.

en el rostro son el producto de la activación conjunta de éstas. A su vez, también existen combinaciones no-aditivas, similar a lo que ocurre con la co-articulación en el habla, donde un fonema puede modificar a los contiguos. Un ejemplo es la combinación de AU 1+4 que suele ocurrir en la emoción de tristeza: Cuando AU 1 ocurre sola, la ceja interna se levanta y cuando AU 4 ocurre sola, las cejas se juntan y se bajan, pero al ocurrir juntas se cancela el descenso producido por AU 4 y el resultado es que las cejas internas se levantan y se juntan dándole una forma oblicua generando arrugas horizontales en el centro de la frente [8]. Otros ejemplos pueden observarse en la Figura 3.6.

En cuanto a la codificación, la forma más básica es simplemente detectar ausencia o presencia pero también se suele anotar la intensidad en una escala de 5 niveles. Sin embargo, las anotaciones de AUs en datasets de expresiones faciales son poco comunes, entre otras cosas por el tiempo que insume y por la habilidad específica requerida por los anotadores.

Los acercamientos para detectar Action Units de manera automática consiguen buenos resultados en datasets de laboratorio pero aún tienen dificultades para el caso de expresiones espontáneas, con variación en la posición de la cabeza, iluminación y oclusión. Como se menciona en [37], existen modelos basados en Support Vector Machines (SVM) para este problema que se adaptan relativamente bien comparados con modelos de Deep learning debido a que estos están muy limitados por la escasez de muestras anotadas disponibles.

Para este trabajo es necesario contar con las anotaciones de algún subconjunto de AUs para así determinar el espacio auxiliar de etiquetas que servirá al entrenamiento de las arquitecturas con Label Distribution Learning. En particular decidimos experimentar con dos herramientas del estado del arte: al igual que en Chen et al. [5] se usa Openface [1], utilizando técnicas de Machine Learning convencionales, y Multi Label Co-Regularization (MLCR) [39] como método alternativo basado en Deep learning.

El modelo de Openface fue entrenado principalmente a partir de datasets de videos de personas retratando emociones posadas: DISFA, BP4D-Spontaneous y SEMAINE. Con el proceso de face registration obtuvieron features geométricos a partir de los landmarks detectados. Luego se agregan a estos, features de apariencia, los Histograms of Oriented Gradients (HOGs). Finalmente se aplica Principal Component Analysis (PCA) para re-

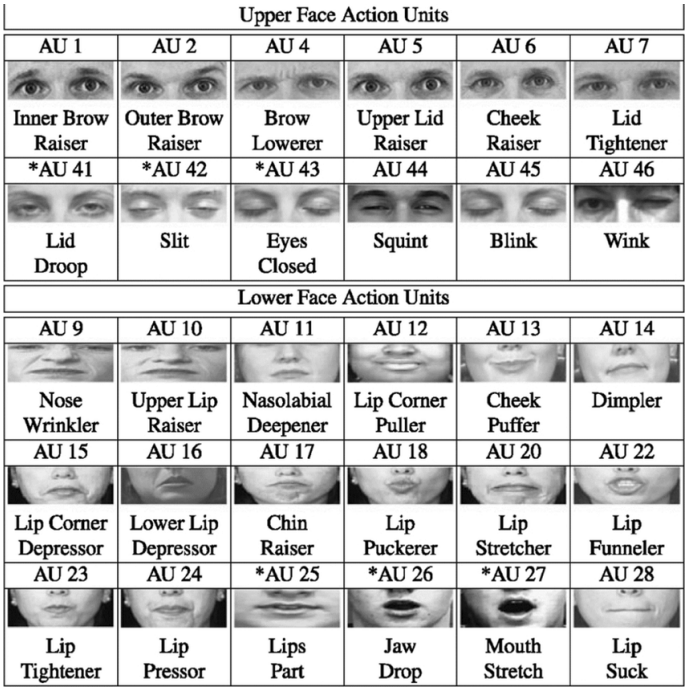


Fig. 3.5: Codificación de acciones faciales en Action Units según FACS.

ducir la dimensionalidad. Los modelos de detección y estimación de la intensidad están bastados en SVM y son los que determinan un subconjunto de 18 AUs con estimaciones de intensidad (continua).

Por el contrario, MLCR es un método basado en el campo de deep learning que aún tiene como desafío encontrar técnicas que puedan aprovechar los grandes volúmenes de imágenes con rostros aún no etiquetados con AUs. Otro de los problemas es, como en FER, que varios métodos suelen estar evaluados en condiciones controladas de laboratorio pudiendo tener una generalización pobre en situaciones in-the-wild. Concretamente el modelo semi-supervisado de MLCR se basa en técnicas de co-training que además explotan las correlaciones entre AUs mediante Graph Convolutional Network y para ellos utiliza una gran cantidad de imágenes no etiquetadas junto con un conjunto reducido anotado.

Fig. 3.6: Ejemplos de combinaciones de Action Units típicas.

4. EXPERIMENTOS Y RESULTADOS

4.1. Datasets

Para evaluar el desempeño de los frameworks de estudio, EfficientFace y CERN, junto con las variantes propuestas en este trabajo, utilizamos datasets de gran escala comúnmente adoptados en FER: RAF-DB [30], FER+ [2], CAER-S [28] y AffectNet [38].

RAF-DB: Tiene 30.000 imágenes faciales anotadas con emociones básicas y compuestas, anotadas a través de crowdsourcing por 40 personas. Para ser consistentes con los trabajos previos, solo se utilizarán las siete emociones básicas: neutral, happiness, sadness, surprise, fear, disgust y anger. Con esto se obtienen 12.271 imágenes para entrenamiento y 3.068 imágenes para test.

Las instancias ya se encuentran alineadas, y se dispone de anotaciones de 5 landmarks, 37 landmarks y bounding box. El dataset cuenta con sujetos de variadas edades, etnias, género, bajo distintas condiciones lumínicas, de oclusión y pose.

CAER-S: Surge ge a partir de la recolección de 20.484 clips de distintas series de TV. Seis anotadores fueron reclutados para etiquetar cada uno de estos clips con las siete emociones básicas y se aprovechó la disponibilidad de subtítulos y audio para facilitar la tarea. Cada clip fue evaluado por 3 anotadores distintos.

En este trabajo utilizaremos la variante CAER-S que toma cuadros representativos de los clips y los divide en dos conjuntos de entrenamiento, con 44.996 muestras, y test (con 20.987 muestras).

A diferencia de los otros datasets, las imágenes no vienen alineadas, por lo tanto se hizo uso de la herramienta MTCNN [50] para conseguir las bounding box, centrar las caras y luego usar los landmarks de los ojos para alinear el rostro. Algunos frames de ejemplo pueden observarse en la Figura 4.1, donde además se aprecia claramente el tipo de escenarios complejos a los cuales tienen que poder adaptarse los métodos.

Fig. 4.1: Muestras del dataset CAER-S con detecciones de bounding box con la herramienta MTCNN en escenas que presentan variación de pose e iluminación.

Al tratarse de un dataset introducido en 2019, su adopción como parte de benchmarks genéricos no es tan común según nuestro conocimiento. Por eso también es de interés aportar resultados sobre el mismo.

FER+: es una versión extendida de FER2013 y consiste de 28.709 imágenes de entrenamiento, 3589 para validación y 3.589 para test. De estos conjuntos, ignoramos el de validación, reportando accuracy sobre el conjunto de test.

Las imágenes fueron anotadas con las siete emociones básicas y contempt (la cual ignoramos) por 10 personas a través de crowdourcing. Como la distribución de los votos por emoción está disponibles, es posible definir una distribución de emociones por instancia que fue de utilidad para el diseño de uno de los experimentos de la Sec4.4.

AffectNet: contiene aproximadamente 450.000 imágenes que fueron manualmente anotadas con un total de 11 categorías de expresiones. En particular para este trabajo consideramos el subconjunto denominado AffectNet-7, por las siete emociones básicas, que tiene 283.901 imágenes para entrenamiento y 3.500 para test.
Similar a otros trabajos previos, para lidiar con el problema del desbalance de clases, utilizamos una técnica de resampling.

4.2. Detalles de implementación y configuración de experimentos

Para abordar el estudio de las arquitecturas propusimos una serie de experimentos afines. Inicialmente en la Sec. 4.3 se corroboraron los resultados obtenidos en los trabajos previos [52], [15] para Single Label Learning. En particular, se extendió el conjunto de datasets utilizados como benchmark básicos en dichos trabajos.

Luego en la Sec. 4.4 se hizo un estudio de ablación similar al de Zao et al. [52] para evaluar la eficiencia de la técnica de Label Distribution Learning para ambos modelos. A diferencia de EfficientFace, el trabajo original de CERN no experimenta con LDL por lo que buscamos que nuestro aporte corrobore la conveniencia de aplicar esta técnica al entrenamiento de otras redes.

En la Sec. 4.5 estudiamos una potencial mejora para la técnica de LDL empleando Action Unit Recognition como tarea auxiliar en ambos frameworks. A través de otro estudio de ablación analizamos la eficiencia de esta técnica en sus variantes directa e indirecta, según se explicó en la Sec. 2.3.2, sobre los datasets RAF-DB y FER+ únicamente.

Para evaluar la robustez del método de LDL ante la presencia de ruido e inconsistencias en las anotaciones, diseñamos el experimento de la Sec. 4.6 generando datasets con ruido sintético en distinta proporción.

Finalmente en la Sec. 4.7 evaluamos la complejidad de las redes a través de una medida más directa que los FLOPs y utilizamos la extensión de torch profiler que además permite obtener información más granular sobre los tiempos por operaciones.

Si bien en los trabajos originales de las arquitecturas estudiadas se reportaron algunas configuraciones para los hiperparametros con los cuales se consiguieron sus resultados, en ocasiones encontramos que estos no alcanzaban para llegar al accuracy esperado. Para cada modelo estudiado hicimos una exhaustiva búsqueda sobre el espacio de hiperparametros aunque restricciones de tiempo no permitieron agotar todas las combinaciones con hiperparametros menos relevantes. Se procuró explorar para todos los experimentos direcciones similares para no introducir sesgos.

En la mayoría de los experimentos, teniendo en cuenta los tiempos que insume el entrenamiento de estos modelos, realizamos varias corridas con semillas distintas. Dado que en su conjunto el entrenamiento está sujeto a numerosas fuentes de aleatoriedad, esto puede dar una idea general de la dispersión esperada aunque la muestra en cuestión

sea chica. Por lo tanto, si bien es de interés reportar el mejor resultado, entendiendo que así también lo hacen el resto de los trabajos previos, también se tuvieron en cuenta, principalmente, las medianas de las corridas.

Todos los experimentos fueron implementados con Pytorch y todos los modelos fueron entrenados en dos servidores, uno con dos GPUs GeForce RTX 3080 y dos CPUs Intel Xeon E5-2680v Ti y otro con dos GPUs Geforce 1080 Ti y una CPU Intel Xeon E5-1650 v4 @ 3.60GHz. También se utilizó una computadora con una GPU Geforce 1080 Ti y una CPU Intel Core i9-9900K CPU @ 3.60GHz.

Optimizadores

Para cada método se exploraron distintos optimizadores en la etapa de entrenamiento: Stocastic Gradient Descent (SGD) con momentum que es aquel utilizado en el trabajo original de EfficientFace, Adamax [25] utilizado en el trabajo original de CERN y finalmente Rectified Adam (RADAM) [32], propuesta en este trabajo, que es una variante del optimizador Adam que introduce un término para rectificar la varianza del learning rate.

Learning rate inicial y Schedulers

El principal hiperparámetro a ajustar fue el learning rate incial cuya elección es crucial para conseguir convergencias adecuadas.

También se experimentaron con distintos schedulers: Aplicando scheduler decay con intervalos fijos, "plateau" que reduce el learning rate si la función de pérdida permanece dentro de un umbral durante una cierta cantidad de epochs y 1cycle que incrementa el learning rate en cada epoch hasta alcanzar una cota superior establecida y luego decrementa hasta alcanzar la cota inferior al finalizar el entrenamiento. Para cada scheduler también se consideró ajustar sus parámetros más relevantes.

Batch size y epochs

A partir de los hiperparámetros establecidos de los componentes anteriores, las pruebas se corrieron con el batch size más grande que pudiera entrar en memoria en pos de incrementar la cantidad de FLOPS y aprovechar el paralelismo a nivel GPU. Esto es en detrimento de tener menos actualizaciones por cómputo y por lo tanto tener que visitar mas ejemplos para alcanzar el mismo error que se conseguiría con batch mas chico. Según Bengio et al. [3], este hiperparámetro debería impactar principalmente en la duración de entrenamiento por lo que puede optimizarse por separado (aunque se recomienda re-optimizar al final, que es lo que se hizo en general).

La cantidad de epochs la acotamos por lo que se reportaba en cada trabajo. En algunas ocasiones, para agilizar las pruebas, siguiendo una idea similar a early stopping, se estableció una cantidad menor de epochs (normalmente 40) y se otorgó una tolerancia para determinar si el error de validación mejoraba. En caso de que sí, se extendía la prueba una cantidad de epochs adicionales, repitiendo el procedimiento anterior para volver a evaluar la condición de terminación.

Preprocesamiento y Data augmentantion

Para ambos modelos respetamos las técnicas de augmentation sugeridas. Para el caso de EfficientFace, corroboramos que la transformación de Random Cropping tenía un efecto de regularización como se reportaba en [52]. Básicamente la técnica consiste en tomar un punto al azar y centrar sobre él una región cuya superficie también se elige como un porcentaje del tamaño original de la imagen tomado. Este porcentaje se toma de manera uniforme en un intervalo, el cual quedó fijo una vez que se encontró una combinación razonable para evitar, en general, señales de overfitting. La otra transformación que se utilizó como parte de data augmentation fue Random Horizontal Flipping. Para el caso de CERN, las transformaciones se eligieron con la técnica de AutoAugment [10].

Todos los datasets evaluados, excepto CAER-S, traen aparejado el problema de clases desbalanceadas. Una técnica para lidiar con esto es implementar alguna estrategia de resampling. Por ejemplo, una forma directa de implementar oversampling es seleccionar instancias de la clase minoritaria y duplicarlas. Sin embargo esto puede traer otros inconvenientes, como overfitting, si no se toman ciertos cuidados como hacer uso de técnicas de data augmentation. Otra alternativa es incorporar los pesos de las clases en el entrenamiento, por ejemplo modificando la función de pérdida para que las instancias de la clase sub-representada tengan más importancia.

En particular, para el dataset AffectNet se utilizó la misma técnica de resampling que los trabajos previos de EfficientFace y CERN [52] [15] la cual combina oversampling y undersampling como se ve en la Figura 4.2[1].

Para el resto de los datasets no se implementó ninguna técnica para lidiar con este problema similar al resto de los trabajos. Sin embargo, avanzado el trabajo, se dedicó un tiempo minoritario para buscar posibles mejoras usando Weighted Cross-Entropy como función de pérdida en EfficientFace con el dataset RAF-DB. Los resultados logran aproximar a los mejores pero se decidió descartar esta dirección, incluso dejando la posibilidad de incorporar otras funciones de pérdida para este fin como Focal Loss.

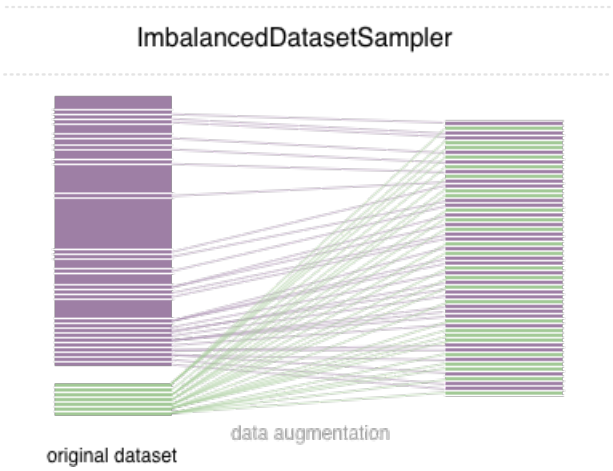


Fig. 4.2: Oversampling y resampling en función de la inversa de los pesos de cada clase.

[1] https://github.com/ufoym/imbalanced-dataset-sampler

4.3. Replicación resultados baseline

En el trabajo de Zhao et al. [52], al presentar el framework de EfficientFace se comparan resultados contra la arquitectura de base, ShuffleNet V2, como baseline. Sus resultados muestran que, efectivamente, la incorporación de los módulos de atención y de extracción de features de contextos locales aportan una mejora de $1,61\,\%$ en RAF-DB y $2,16\,\%$ en CAER-S habiendo preentrenado los modelos con el dataset MS-CELEB-1M. Por otro lado, en CERN [15] se muestra una ganancia de $3,4\,\%$ respecto de la arquitectura base LightCNN29 también en el dataset RAF-DB.

Considerando esos resultados como dados, como experimento preliminar de este trabajo buscamos corroborar los resultados obtenidos para las arquitecturas de CERN y EfficientFace en su versión Single Label Learning (de ahora en más EF SLL y CERN SLL respectivamente) sobre los datasets propuestos en cada trabajo. Para poder compararlas entre ellas, también extendimos los resultados a otros datasets cuando fue necesario.

Para obtener los resultados de este experimento, ajustamos los hiperparámetros durante el entrenamiento según los valores sugeridos en los trabajos en caso de consignarse. Además, nunca se entrenó por más epochs que los explicitados en cada trabajo para los resultados reportados. En ocasiones los resultados obtenidos de esta forma se acercaban a lo reportado, pero en algunas situaciones existían discrepancias importantes. En cualquier caso, siempre se siguieron los lineamientos explicados en la Sec. 4.2 para buscar en el espacio de hiperparámetros. Por ejemplo, aunque con el optimizador SGD para EfficientFace en general se conseguían resultados como los de [52], se encontró que con el optimizador RADAM podía extraerse un poco más de perfomance. Lo mismo ocurrió al introducir schedulers alternativos como Plateau y 1cycle con los que logramos aún mejores resultados.

Conseguimos mejorar ampliamente los resultados reportados por los trabajos previos en la mayoría de los datasets. Los resultados pueden verse en la Tabla 4.2 para las filas de EfficientFace (EF) y CERN con método de entrenamiento SLL en la columna Train. Cuando haga falta utilizaremos como referencia la Tabla 4.3, recopilada de [15] y [52], que contiene el accuracy de distintas arquitecturas lightweight del estado del arte y otras arquitecturas más complejas evaluadas sobre los datasets de interés. Estos modelos también fueron preentrenados en MS-CELEB.

Comenzamos con un análisis para los resultados máximos obtenidos contra los reportados en los trabajos [52] [15]. Para el caso de EfficientFace SLL, conseguimos mejorar el accuracy en $1,92\,\%$ en RAF-DB aunque el obtenido sobre CAER-S cae un $0,21\,\%$. Para el primer caso, asociamos esa mejora principalmente al uso del optimizador RADAM. Aunque el trabajo original no reporta sobre FER+, está a la vista que es es un muy buen resultado al compararlo con CERN y con el resto de las arquitecturas de 4.3. Una vez más, RADAM admitió mejores resultados que el resto de los optimizadores considerados para este caso.

No podemos comparar contra los resultados específicos de esta parte sobre AffectNet ya que no pudimos encontrarlo en el trabajo original. Notamos que para EF SLL sobre AffectNet el valor de accuracy es radicalmente más bajo que el resto, esto tiene que ver, posiblemente, con la gran cantidad de instancias que tiene a comparación de los otros datasets y por el nivel de ruido introducido globalmente. Sin embargo, revisando los resultados de la Tabla 4.3, notamos que el accuracy de una arquitectura ShuffleNet base, sin ningún tipo de módulo para el problema de FER in-the-wild, alcanza los $61,1\,\%$ sobre

AffectNet. Siendo que el máximo valor encontrado para nuestro caso fue de $57,08\,\%$, probablemente esta situación anómala pueda subsanarse si incrementamos el presupuesto del tiempo asignado a optimizar este caso.

En cuanto a CERN, obtuvimos mejoras menos drásticas: un aumento de $0,49\,\%$ y $0,58\,\%$ en FER+ y AffectNet respectivamente y una pérdida de $0,59\,\%$ en RAF-DB. La mayoría de los resultados fueron obtenidos con ADAMAX como en [15]. Creemos que las diferencias pueden venir de variaciones propias del entrenamiento con este tipo de métodos. En el trabajo se reporta el uso de iguales learning rates para todos los datasets, en nuestro caso en ocasiones obtuvimos combinación que se ajustaban un poco mejor a algún dataset.

Comparando los desempeños entre los dos métodos estudiados a través de las medianas, vemos que no existe un claro ganador (ver 4.3). EfficientFace es mejor por $1,21\,\%$ y $0,44\,\%$ en RAF-DB y FER+ respectivamente y CERN es considerablemente mejor por $5,5\,\%$ y $4,75$ en AffectNet y CAER-S. Probablemente el resultado sobre el primero esté un poco exagerado por la anomalía que reportamos previamente.

Sorprendentemente, el resultado obtenido con el dataset CAER-S es también demasiado bueno. Siguiendo lo sugerido para la mayoría de los frameworks de FER, todas las imágenes provenientes de las escenas de series de TV fueron preprocesadas para detectar y frontalizar el rostro. Durante el entrenamiento de CERN nos topamos con que el mejor Accuracy conseguido era relativamente bajo comparado con el resto de los modelos, aproximadamente un $10\,\%$ por debajo. Al probar, equivocadamente, un entrenamiento sin estas transformaciones fue que se obtuvieron los mejores resultados reportados. Revisando otros trabajos recientes [53] y el trabajo original donde se propone este dataset junto con la CAER-S Netowrk [28] notamos que la misma optaba por aprovechar el contexto de la imagen, argumentando que aislando el rostro se pierde información gesticular valiosa que puede dar indicios sobre la emoción retratada. Volviendo sobre el diseño de CERN, la arquitectura emplea mecanismos de atención que pueden destacar regiones salientes locales y globales, a la vez que particiona la imagen en pedazos a ser evaluados por el ensamble de clasificadores. Por lo tanto, esto podría estar siendo lo suficientemente robusto como para aprovechar las condiciones particulares de este dataset, donde justamente al tratarse de escenas, es probable que el encuadre sea lo suficientemente abierto para destacar otras partes claves del contexto. Esta situación sin duda alienta a plantear otro experimento para corroborar si lo mismo ocurriría con EfficientFace.

Por el lado de la complejidad, aunque ambos modelos tengan una cantidad de parámetros del mismo orden, los MFLOPs de CERN son aproximadamente 10 veces más. A priori, en base a los a desempeños que no inclinan la balanza de manera general, este punto es interesante de remarcar. Sin embargo, es una medida indirecta de complejidad y por lo tanto debe ser tomada con cuidado. A su vez, como se esperaba, estas arquitecturas diseñadas específicamente para lidiar con el problema de FER in-the-wild superan cómodamente a otras redes lightweight del estado del arte como MobileNet-V2 (ver la Tabla 4.3).

Otro punto destacable, es que ambas se desempeñan en general de manera equiparable a modelos más complejos. Por ejemplo, en la Tabla 4.3 se ve que SCAN tiene aproximadamente 70 veces más parámetros pero en RAF-DB y FER+ no consigue más de una mejora $1,8\,\%$ respecto a EfficientFace. Además, otro punto de comparación que resulta interesante de mencionar es el mismo LDG entrenado para ser usado con Label Distribution Learning en la Sec. 4.4, que fue implementado con una ResNet-50 teniendo 10 veces más parámetros y casi 40 veces más MFLOPs que EfficientFace. Puede verse en la Tabla 4.2 que el margen

de mejora contra los modelos lightweight SLL es escaso, lo cual es alentador.

Método	#Params (M)	MFLOPs	RAF-DB	FER+	CAER-S	AffectNet
MobileNet-V2	2,2	312,86	85,04	87,06	79,23	61,4
ShuffleNet	1,3	147,79	84,58	84,06	84,06	61,1
ResNet-18	11,1	1818,56	85,65	87,37	84,67	60,69
SCAN	70M	7005	89,02	89,42	-	65,14
ResNet-50	23,52	4109,48	88,69	89,29	86,17	59,2
EF SLL	1,28	154,18	87,58	88,44	84,30	57,08
CERN SLL	1,45	1781	86,37	88,66	88,10	62,64

Tab. 4.1: Accuracy máximo de arquitecturas del estado del arte tipo lightweight, excepto SCAN, según lo reportado en [15] [52] y nuestros valores hallados para EfficientFace y CERN SLL al igual que ResNet-50 (LDG). Los resultados se reportan a partir de los modelos preentrenados en MS-CELEB.

En resumen, a través de este experimento corroboramos los resultados de los trabajos previos permitiéndonos continuar con mayor confianza hacia el resto del estudio. Los valores alcanzados durante el proceso de optimización en algunos casos muestran mejoras considerables con respecto a lo que se reporta en los trabajos base.

Configuración de hiperparámetros: específicamente, para entrenar sobre CAERS se entrenó durante 350 epochs con un batch-size de tamaño 128, 1cycle scheduler con learning rate máximo 0.001 y optimizador RADAM, con $\beta_1 = 0,9$ $\beta_2 = 0,999$ $eps = 1e-8$. Sobre AffectNet se entrenó durante 20 epochs con un batch-size de tamaño 16, scheduler decay con un factor de 0,1 cada 15 epochs, learning rate inicial de 0,001, y optimizador SGD con weight decay 0,0001. Sobre RAF-DB se entrenó durante 100 epochs con un batch-size de tamaño 128, scheduler "plateau", learning rate inicial 0.001 y optimizador RADAM como en CAER-S. Sobre FER+ se entrenó durante 100 epochs con early stopping, batch-size 128, scheduler decay con un factor 0,1 cada 15 epochs, learning rate inicial de 0,001 y optimizador RADAM como en CAER-S. Todos los modelos anteriores se entrenaron con random crop y random horizontal flip como técnicas de data-augmentation y con el objetivo de evitar overfitting. Cuando fue necesario, se aplicó MTCNN [50] para detectar y centrar los rostros.

CERN se entrenó sobre AffectNet con learning rate 0,0001 sobre la red base que se conserva de LightCNN-29 y 0,001 para el resto. Para FER+ y CAER-S se usó 0,001 para ambas partes de la red. Sobre RAF-DB se usó 0,001 y 0.01 respectivamente. Todos los modelos anteriores se entrenaron hasta 100 epochs con early stopping, batch size 64 excepto 128 en rafdb, con scheduler plateau, optimizador ADAMAX, excepto FER+ donde se usó RADAM, y con la técnica de auto-augmentation. Sobre el dataset CAER-S se reportaron los mejores resultados obtenidos que se corresponden con las imágenes tomadas sin face registration como se mencionó previamente en esta sección.

4.4. Efectividad de Label Distribution Learning

4.4.1. Estudio de ablación

A través de este estudio de ablación pudimos evaluar la eficiencia al emplear la estrategia de Label Distribution Learning. Para ello, se obtuvo en primer lugar un generador de distribuciones (LDG) siguiendo los lineamientos de la Sec. 4.2.

El LDG, básicamente una arquitectura ResNet-50, se entrenó sobre RAF-DB y FER+ durante 100 epochs con un batch-size de tamaño 16, scheduler decay con un factor de 0,1 cada 15 epochs, learning rate inicial de 0,001 y optimizador SGD con weight decay 0,0001. Sobre CAER-S se utilizó optimizador ADAM con batch size 64 y scheduler 1cycle con learning rate máximo de 0,001 durante 300 epochs. Se usaron las mismas técnicas de data-augmentation y preprocesamiento que en EfficientFace.

El mejor accuracy del LDG, sobre cada dataset, está, en general, por arriba de los reportado en [52]: un aumento de $1,76\,\%$ y $1,35\,\%$ en RAF-DB y CAER-S respectivamente.

A partir de los generadores ajustados a cada dataset, se entrenó a EfficientFace LDL y CERN LDL adaptadas para tener en cuenta las distribuciones de emociones generadas por el LDG adecuado y se obtuvieron los resultados de la Tabla 4.2 (aquellos con categoría LDL en la columna Train).

Dataset		LDG	EfficientFace		CERN	
		SLL	SLL	LDL	SLL	LDL
RAF-DB	min	88,30	86,50	87,38	86,04	86,50
	med	**88,41**	86,92	88,03	86,27	86,60
	max	88,69	87,58	88,52	86,37	86,76
CAER-S	min	85,77	84,05	84,01	87,69	88,79
	med	85,97	84,17	84,20	87,89	**88,95**
	max	86,17	84,30	84,40	88,10	89,11
FER+	min	88,72	87,96	88,62	88,09	88,27
	med	**88,97**	88,09	89,10	88,39	88,66
	max	89,29	88,44	89,32	88,66	88,72
AffectNet	min	57,94	55,60	57,58	62,02	62,42
	med	58,57	56,70	57,87	62,33	**62,75**
	max	59,2	57,08	58,17	62,64	63,08

Tab. 4.2: Resultados para el experimento de ablación de la la Sec. 4.4. Se reportan valores de accuracy rate (%) mínimo, mediana y máximo de un total de 5 corridas, excepto en los datasets más grandes, CAER-S y AffectNet, donde se hicieron 2 (para estos dos datasets el valor de la mediana se corresponde con el promedio del mínimo y máximo). Se agrega también el accuracy rate alcanzado por cada LDG utilizado para entrenar a las otras arquitecturas.

Como primer punto a remarcar, los resultados del estudio son alentadores. Bajo nuestro benchmark, las dos arquitecturas mejoraron su rendimiento con la estrategia de Label

Distribution Learning en todos los datasets. Específicamente, EfficientFace mejora su accuracy mediana por $1,11\,\%$, $0,03\,\%$, $0,1,01\,\%$, $1,17\,\%$ para RAF-DB, CAER-S, FER+ Y AffectNet respectivamente. Por el lado de CERN, $0,33\,\%$, $1,06\,\%$, $0,27\,\%$ y $0,42\,\%$ para el mismo orden anterior. A simple vista, CERN parece ser menos susceptible a la incorporación de LDL con incrementos que no superan los 0,5 %. La excepción es sobre CAER-S, que particularmente es la única en la que supera a EfficientFace.

Aunque las mejoras son notorias, las mismas son considerablemente menos abultadas que los resultados del trabajo original de EfficientFace donde se llegan a ver mejoras de hasta $2,7\,\%$ en RAF-DB. Es posible que el desempeño de EfficientFace SLL haya sido subestimado en sus resultados, ya que como se reportó previamente en la Sec. 4.3 con RADAM obtuvimos amplías mejorías para ese caso (y aún así, con una configuración similar a la del trabajo original ya se había conseguido superar su resultado por aproximadamente un $1\,\%$). Por otro lado, consideramos los resultados sobre CERN un aporte, positivo, al trabajo original.

Si comparamos los resultados contra el LDG encontraremos otro punto destacable: A pesar tratarse de redes mucho menos complejas que la ResNet-50, EfficientFace y CERN logran resultados comparables. De hecho, en ocasiones logran mejorarlo: El mejor resultado sobre CAER-S lo consigue CERN LDL superando por $2,94\,\%$ al LDG. Lo mismo ocurre con AffectNet, donde CERN la supera por $3,88\,\%$. Sin embargo, en este caso pensamos que, al igual que lo discutido sobre este dataset en la Sec. 4.3, se trata de un resultado anómalo con un posible sesgo de entrenamiento. Por otra parte, sobre FER+ el mejor resultado lo consigue EF SLL aunque por un margen prácticamente despreciable comparado con el LDG. Es así que en el único dataset en el cual el LDG consigue el mejor resultado es en RAF-DB y lo hace por un margen menor a $0,5\,\%$.

En definitiva, resaltamos la capacidad de las redes lightweight estudiadas para equiparar y en ocasiones mejorar, a menor costo, el desempeño sobre una arquitectura del estado del arte de mayor complejidad (en términos de parámetros y FLOPs). Notar que parte de esta mejora tiene que ver no solo con la introducción de LDL, si no de los distintos módulos para adaptarse al problema específico de FER in-the-wild sin dañar significativamente la complejidad. El lector es referido a los estudios de ablación de los trabajos originales [52][15] para cuantificar precisamente el aporte de cada uno de estos módulos.

Concentrándonos en el análisis más fino del desempeño de los modelos, resulta de interés comparar el accuracy rate por clase. En particular, observamos que en general las clases negativas suelen parecerse, ya que comparten activaciones de músculos faciales en determinadas regiones. No solo eso, si no que suelen estar subrepresentadas (particularmente sadness, fear y anger) por la dificultad de obtener anotaciones de calidad de las mismas.

Observando las matrices de confusión de las figuras Figura 4.3 queda claro que las emociones positivas (neutral, happy, surprise) son de las más fáciles de reconocer, consiguiendo accuracy rate superiores al $90\,\%$ para EfficientFace y CERN en los datasets RAF-DB y FER+. En particular, happiness es la emoción que prevalece como la mejor clasificada, algo que tiene sentido a partir de lo entendido en la exploración de datos y trabajos previos.

Consistente con lo anterior, las emociones negativas suelen ser las más difíciles en RAF-DB y FER+, en particular fear y disgust no consiguen sobrepasar el $65\,\%$. Más aún, fear tiende a confundirse mayormente con sad y surprise. Por otro lado, disgust es fácilmente

confundible con anger en FER+ y con Neutral en RAFDB. El caso más marcado es el de CERN en FER+, que solamente clasifica bien las emociones de este tipo un 36 % de las veces y un 48 % lo confunde con anger.

Observando las matrices de confusión de las figuras Figura 4.4, notamos que sobre CAER-S ambos modelos clasifican bien el general de todas las emociones, yendo a contracorriente de lo que se intuye a partir de lo anterior. Esto puede deberse, por un lado, a que el dataset tiene más del doble de instancias que FER+ y que, además, las clases están balanceadas. A su vez AffectNet presenta las dificultades esperadas con las clases negativas. Además, es llamativa la tendencia a equivocarse prediciendo Neutral a cualquier categoría. Por ejemplo, el 25 % de las instancias de surprise las asigna a Neutral, lo mismo con un 27 % de instancias de sadness.

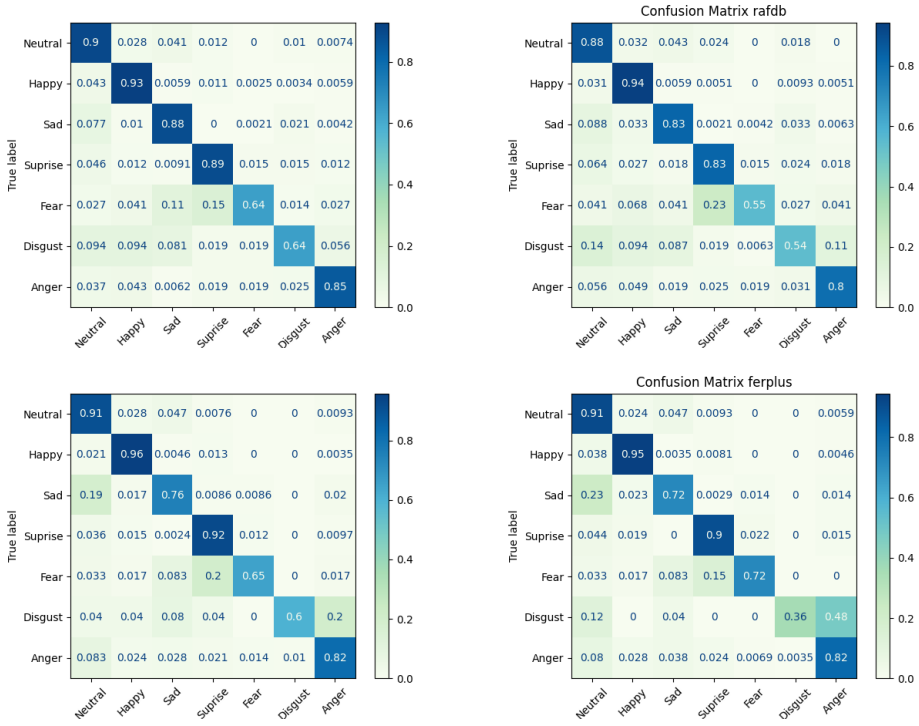


Fig. 4.3: Matrices de confusión para RAF-DB (arriba) y FER+ (abajo) para EfficientFace (izq) y CERN (der).

4.4.2. Comparación de LDL via LDG y LDL con etiquetas de distribución

Continuando con la línea de este experimento, una pregunta que podemos contestar para el caso de FER+, es que tan efectiva es la técnica de LDL a través del LDG comparada con anotaciones manuales de la distribución de emoción.

Aprovechando que FER+ tiene anotados para cada instancia los votos que recibió cada una de las categorías, se puede estimar una distribución de emoción y usarlas como ground-truth en Label Distribution Learning sin la necesidad de pasar por el LDG. Realizamos esta prueba con EfficientFace y luego de encontrar el mejor modelo, notamos que, aún así,

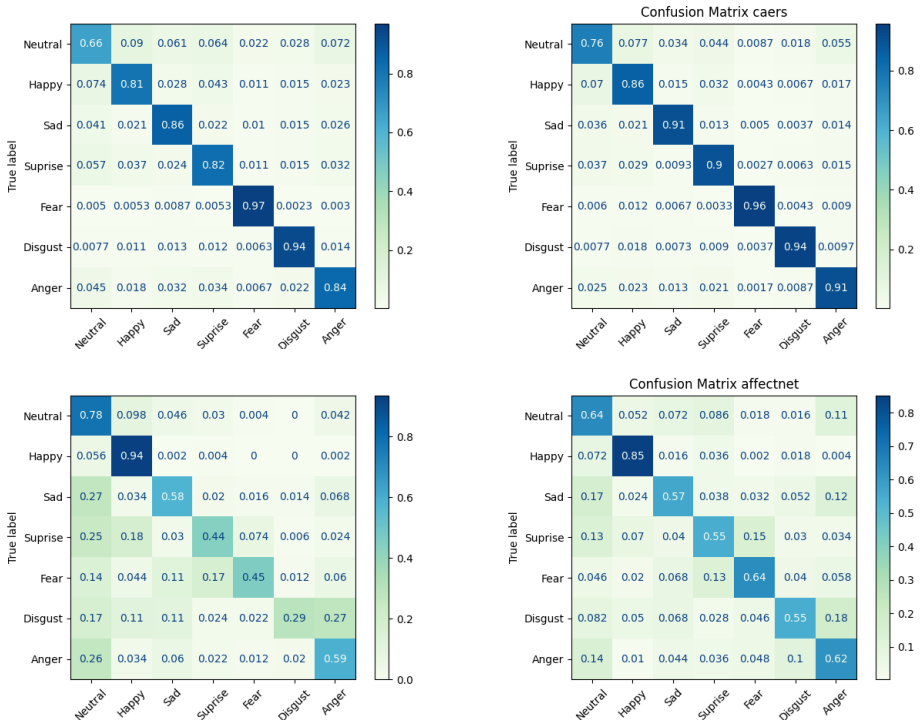


Fig. 4.4: Matrices de confusión para CAER-S (arriba) y AffectNet (abajo) para EfficientFace (izq) y CERN (der).

el uso del LDG es equiparable a la anotación manual, siendo, específicamente, un $0,2\,\%$ mejor.

4.4.3. Influencia de la calidad del LDG

Vimos que la incorporación el uso del LDG admite una ganancia de rendimiento razonable en ambas arquitecturas. Luego, otra pregunta que surge es que tan sensible es el entrenamiento de las redes lightweight a la calidad de anotaciones provistas por el LDG. Dicho de otra manera, ¿qué tan importante es maximizar la capacidad del LDG entendida como su accuracy? Para ello, planteamos escenarios controlados y evaluamos como cambia el desempeño de la red lightweight en función del LDG.

Considerando el dataset RAF-DB y la arquitectura de EfficientFace, se entrenaron 4 LDGs cuyos valores de accuracy sobre el test se se corresponden con la primer columna de la Tabla 4.4.3. Se utilizó a cada uno de ellos en el modelo EF LDL entrenada siempre con la misma configuración de hiperparametros. Luego se repitió este procedimiento, para todos los casos, cuatro veces con semillas distintas. Los resultados de los retornos de accuracy promediados para cada caso se observan en la Tabla 4.4.3, donde apreciamos lo esperado: la capacidad del LDG está directamente ligada con la calidad de aprendizaje que pueda conseguir EfficientFace.

Observamos que EF LDL consigue menor retorno a medida que el accuracy del LDG se acerca al máximo de su capacidad y, de hecho, se vuelve negativo a partir de un umbral cercano a dicho máximo. Esto indicaría que no es necesario entrenar el LDG a su máxima capacidad: si pensamos que al optimizar la función de pérdida del LDG estamos minimi-

zando el error entre la estimación del generador, soft label, contra el ground truth, hard label, la distribución aprendida podría terminar pareciéndose demasiado a la determinada por el vector lógico de la emoción categórica del ground-truth. Esto se volvería contraproducente ya que no estaría capturando la ambigüedad propia de las expresiones siendo este el propósito de Label Distribution Learning en primer lugar.

Por último, examinando las curvas de entrenamiento entendemos que para valores de accuracy suficientemente bajos, como el de $63,10\,\%$ el modelo de EF LDL no entrena correctamente. Es decir, el nivel de ruido que provee el LDG dificulta demasiado el aprendizaje, mostrando una función de pérdida de entrenamiento que tempranamente se vuelve constante y un accuracy de test marcadamente superior al de train.

Acc. LDG	**Retorno Acc. EF LDL**
88.70	-0.2
87.80	+0.5
85.80	+1.2
83.10	+1.8
63.10	×

Tab. 4.3: El retorno de accuracy es reportado sobre el promedio de 4 corridas sobre RAF-DB. La cruz indica que el modelo no fue capaz de entrenar correctamente.

Configuración de hiperparámetros: EfficientFace se entrenó sobre RAF-DB y FER+ durante 100 epochs con un batch-size de tamaño 128, scheduler plateau, learning rate inicial de 0.001 y optimizador RADAM . Sobre CAER-S, misma configuración pero usando scheduler 1cycle con learning rate máximo de 0.001 durante 350 epochs. Se usaron las mismas técnicas de data-augmentation y preprocesamiento que en EfficientFace.

Para entrenar CERN se utilizó 0.001 para los dos learning rates en todos los casos , 100 epochs con early stopping, batch size 64, scheduler plateau, optimizador ADAMAX y la técnica de auto-augmentation. Sobre CAERS no se usó face registration.

4.5. Efectividad del uso de Action Unit Recognition como tarea auxiliar

4.5.1. Análisis exploratorio de los datos obtenidos

En la Sec. 3.2 se describe la propuesta de este trabajo en relación a la incorporación del espacio de etiquetas de Action Units para guiar el entrenamiento de ambas redes. Más precisamente esto se instrumenta a través de la ecuación 3.5 que requiere etiquetar todas las instancias del dataset con un vector de intensidad de Action Units.

Para ello, asumimos cierta noción de suavidad: Si dos imágenes están cerca en el espacio de etiquetas de AUs entonces deberían tener distribuciones de emociones similares.

Aplicando una técnica de reducción de dimensionalidad y visualización de datos como t-SNE [35] se puede mapear el espacio de AUs al plano y así obtener una primera impresión sobre la validez del principio de suavidad. En primer lugar, hay que notar que en este tipo de visualizaciones el tamaño de los clusters y la distancia inter-cluster de la representación obtenida no siempre refleja la topología original. Sin embargo las vecindades en dimensión alta deberían preservarse. Además, la forma de las representaciones obtenidas depende fuertemente de la elección del hiperparámetro perplexity que puede interpretarse como la medida de la cantidad estimada de vecinos [35]. Por lo tanto se abordó el análisis luego de observar resultados para distintas elecciones de lo anterior.

Para tener una noción preliminar sobre calidad de las anotaciones y topología, consideramos el dataset Cohn-Kanade [23], que aunque no es parte del benchmark, ya cuenta con anotaciones de Action Units. Además, estas anotaciones fueron hechas por expertos sobre secuencias de videos tomadas en condiciones ideales de laboratorio. No solo eso, si no que también los datos fueron tratados: aquellos anotaciones donde había bajo coincidencia entre-anotadores se descartaban, al igual que aquellas que no cumplían ciertas reglas de codificación. En particular, las instancias con presencia de AUs inconsistentes para la emoción anotada eran filtradas al igual que aquellas con ausencia de AUs necesarias para formar la emoción prototípica asociada (según EMFACS [40]).

Si vemos el resultado de una de las representaciones t-SNE obtenidas para este caso en la Figura 4.5 daremos cuenta de que es realmente una situación ideal. El procesamiento de los datos y las condiciones controladas de toma de los mismos da como resultado una clusterización bien marcada para cada categoría con tan solo algunos puntos que se mezclan.

Por el contrario, si anotamos el dataset con la herramienta MLCR obtendremos un espacio sumamente diverso. Podemos observar en la Figura 4.5 dicha situación donde, a excepción de la categoría happiness cuyo grupo se encuentra bien diferenciado, el resto de las instancias no quedan bien estructuradas en clusters. Sin embargo, sí es cierto que hay regiones donde prevalece la población de un cierto tipo de emoción como por ejemplo surprise y disgust.

Dado lo anterior, se puede pensar que, al menos utilizando MLCR, la calidad de anotaciones no favorece una topología donde tenga tanto sentido pensar en el principio de suavidad. Más aún, la situación para contexto in-the-wild podría agravarse.

Al considerar algunos dataset in-the-wild relevantes para este trabajo como RAF-DB y FER+, hay que tener que en cuenta, además, que el balance de clases tampoco es uniforme (ver la Figura 4.6) ya que normalmente hay una sobre-representación de clases positivas (happiness, surprise, neutral). Esta falencia es una situación bastante común en datasets cuyas imágenes son recolectadas de Internet dado que es más difícil conseguir emociones negativas (disgust, fear, anger) principalmente por el grado de similaridad en

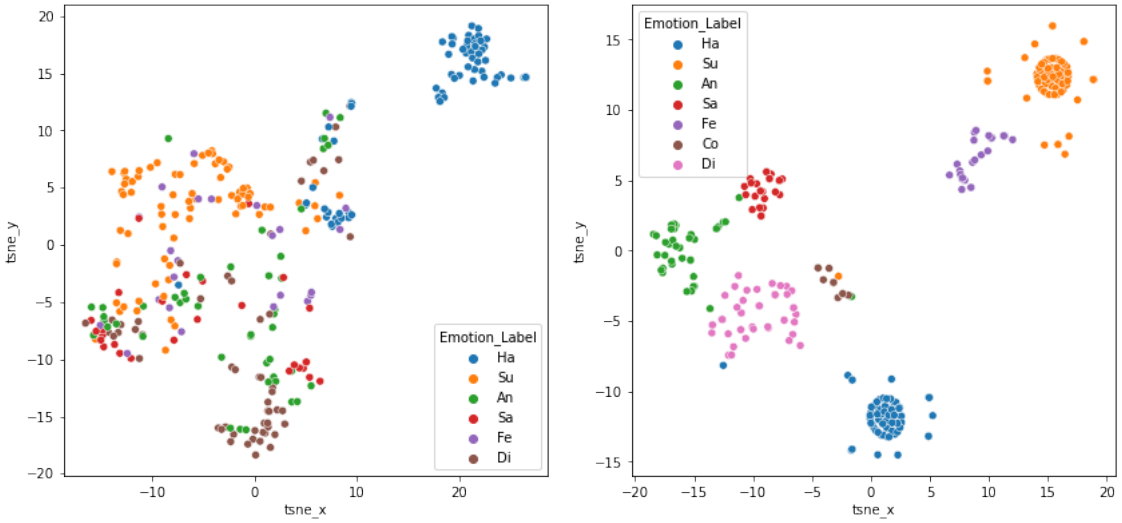


Fig. 4.5: Visualizaciones de los descriptores t-SNE para el dataset Cohn Kanade a partir de las anotaciones automáticas de MLCR (izq) y las anotaciones manuales ya incorporadas (der).

los movimientos musculares que exhiben estas expresiones. Así, muchas instancias son descartadas al no contar con un grado de confianza en la anotación suficiente.

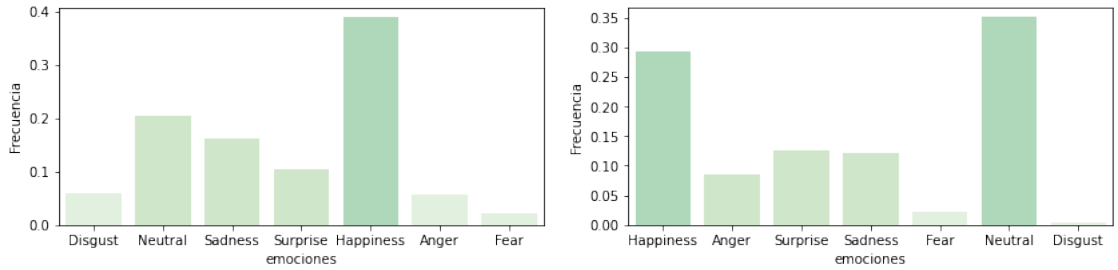


Fig. 4.6: Balance de clases para el dataset RAF-DB (izq) y FER+ (der).

Analizaremos la calidad de las anotaciones obtenidas por las herramientas de MLCR y OpenFace, primero enfocándonos en los resultados de las representaciones t-SNE obtenidas para el dataset RAF-DB como puede verse en la Figura 4.7.

A simple vista, queda claro que en ambos casos resulta difícil definir una buena clusterización. Notar también que, a diferencia de datasets como Cohn-Kanade, en estos casos hay que lidiar con un cantidad mucho mayor de anotaciones ruidosas. A pesar de eso, y similar a lo observado previamente en la Figura 4.5, en ambas técnicas las emociones positivas (principalmente happiness) muestran mayor estructura y tienen regiones de aparente homogeneidad. Al contrario, la mayoría de las instancias negativas suelen estar dispersas y regiones que las caractericen. De hecho, en ambas se pueden observar regiones donde destacan mezcla de instancias de anger y sadness. Este tipo de emociones se caracterizan por movimientos de músculos similares en el área de los ojos (AU 4+5), pudiendo explicar la dificultad en diferenciarlas y la tendencia a aparecer juntas en las representaciones.

Analizando los resultados sobre FER+ utilizando MLCR (ver Figura 4.8) encontramos una situación similar a lo resaltado en RAF-DB aunque las instancias de las categorías

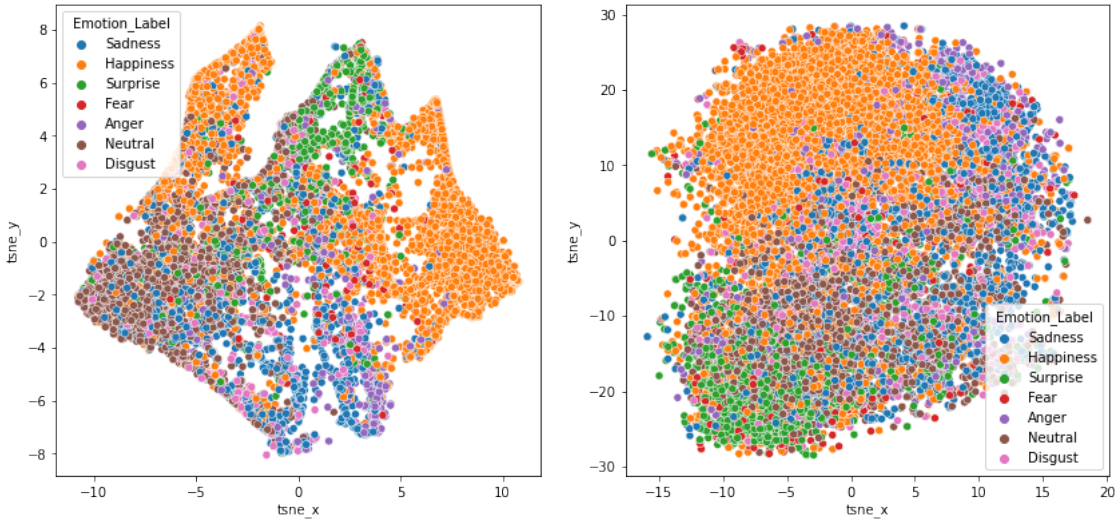


Fig. 4.7: Visualizaciones de los descriptores t-SNE para el dataset RAF-DB a partir de las anotaciones automáticas de MLCR (izq) y las anotaciones de OpenFace (der).

positivas están visiblemente mejor agrupadas.

En conclusión, considerando los resultados mostrados en las figuras previas y otras visualizaciones con perplexity distintas, podemos pensar que el principio de suavidad tiene sentido pensado de manera local sobre ciertas regiones del espacio. Durante la experimentación corroboraremos si aún bajo esta consideración más débil sigue teniendo sentido incorporar esta estrategia.

Para continuar evaluando la calidad de las anotaciones, utilizando como guía la codificación de EMFACS [40], simplificada en la Tabla 4.4, que define emociones básicas prototípicas a partir de la presencia de Action Units en las expresiones faciales también podremos analizar la calidad de las anotaciones. Notar que, por simplicidad, la tabla muestra las AUs necesarias para formar una emoción prototípica pero también es posible definir criterios para admitir la presencia de otras AUs que no sean inconsistentes.

Centrándonos en el dataset RAF-DB analizamos, para cada emoción, la intensidad promedio, entre todas las instancias, de cada Action Unit y se las normaliza al rango $[0, 1]$. En la Figura 4.9 resumimos estos resultados a través de dos visualizaciones luego de aplicar MLCR y en la Figura 4.10 se ve lo mismo pero con la aplicación de OpenFace.

A simple vista, comparando las activaciones promedio a través de los gráficos de barra de cada herramienta, notamos que en MLCR cada emoción presenta entre dos o tres AUs cuya activación se destacan del resto y el umbral de activación para estas está normalmente por arriba de 0.5.

Por el contrario, OpenFace genera activaciones que son más parejas y las AU que deberían sobresalir, según EMFACS, lo hacen con una intensidad que nunca supera los 0.5. Es importante tener en cuenta que una activación menor a 0.2 para algunas AUs de algunas emociones puede ser considerada insuficiente según EMFACS.

Yendo un poco más en profundidad, chequeamos si se cumplían algunas de las condiciones empleadas en [23] para filtrar anotaciones de AU inconsistentes.

En particular, AU 4 (brow lowerer) es una componente de emoción negativa que debería estar ausente en el resto. Observando Figura 4.9, vemos que, efectivamente, la activación

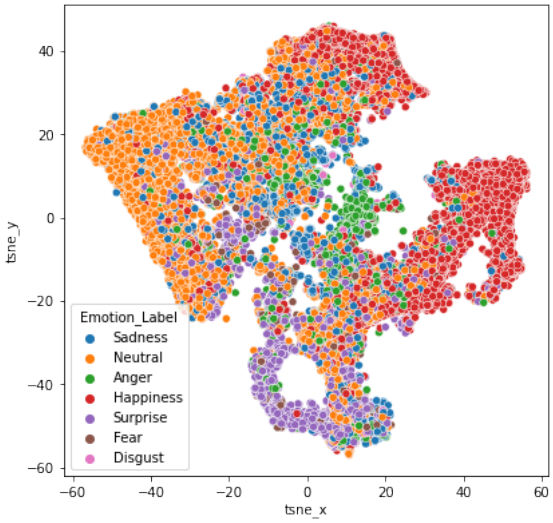


Fig. 4.8: Visualización de los descriptores t-SNE para el dataset FER+ a partir de las anotaciones automáticas de MLCR.

promedio en las emociones positivas es considerablemente baja y según el violin plot asociado, gran parte de la densidad se acumula en estos valores bajos. Una situación similar ocurre con las activaciones promedio de Figura 4.10. Sin embargo, como se dijo anteriormente, resulta llamativo que la intensidad en emociones negativas esté atenuada.

Otro caso es el de AU 9 (nose wrinkler) que solo debería estar presente en disgust y, posiblemente, en anger con un activación menor a 0.2. En Figura 4.9 observamos que en Anger se encuentra dentro de las tres AUs prevalentes, pero viendo la distribución observamos cierta bimodalidad indicando que hay un cantidad de instancias que tiene una activación muy marcada contrario a lo esperado. Para el caso de disgust, a pesar de ser una de los rasgos más distintivos de esta emoción prototípica, la intensidad media es relativamente baja y poco preponderante en ambos métodos.

Emotion	**Action Units**
Happiness	6+12
Sadness	1+4+15
Surprise	1+2+5+26
Fear	1+2+4+5+7+20+26
Anger	4+5+7+23
Disgust	9+15+17

Tab. 4.4: Codificación de emociones prototípicas a partir de presencia de Action Units según EMFACS [40].

Si consideramos happiness, que es una de las categorías que observamos se diferenciaban y agrupaban mejor que el resto en las representaciones t-SNE para ambos métodos, las Action Units que necesitan estar presente según la Tabla 4.4 son AU 12 (lip corner puller) y AU 6 (cheek raiser). En la Figura 4.9 verificamos que, en el caso de AU 12, no solo su activación promedio es de las más altas de todas las emociones, si no que la distri-

bución de intensidades se nota asimétrica negativa (asimetría a izquierda), indicando que la mayor densidad realmente se acumula sobre valores de intensidad altos. Una situación similar ocurre con AU 25 que, aunque no necesaria, es una de las variantes admitidas en este tipo de emoción. La AU 6 (cheek raiser), en cambio, parece seguir un distribución bimodal con amplitudes similares, denotando que también existe densidad considerable de instancias con bajas activaciones y por lo tanto algo no ideal.

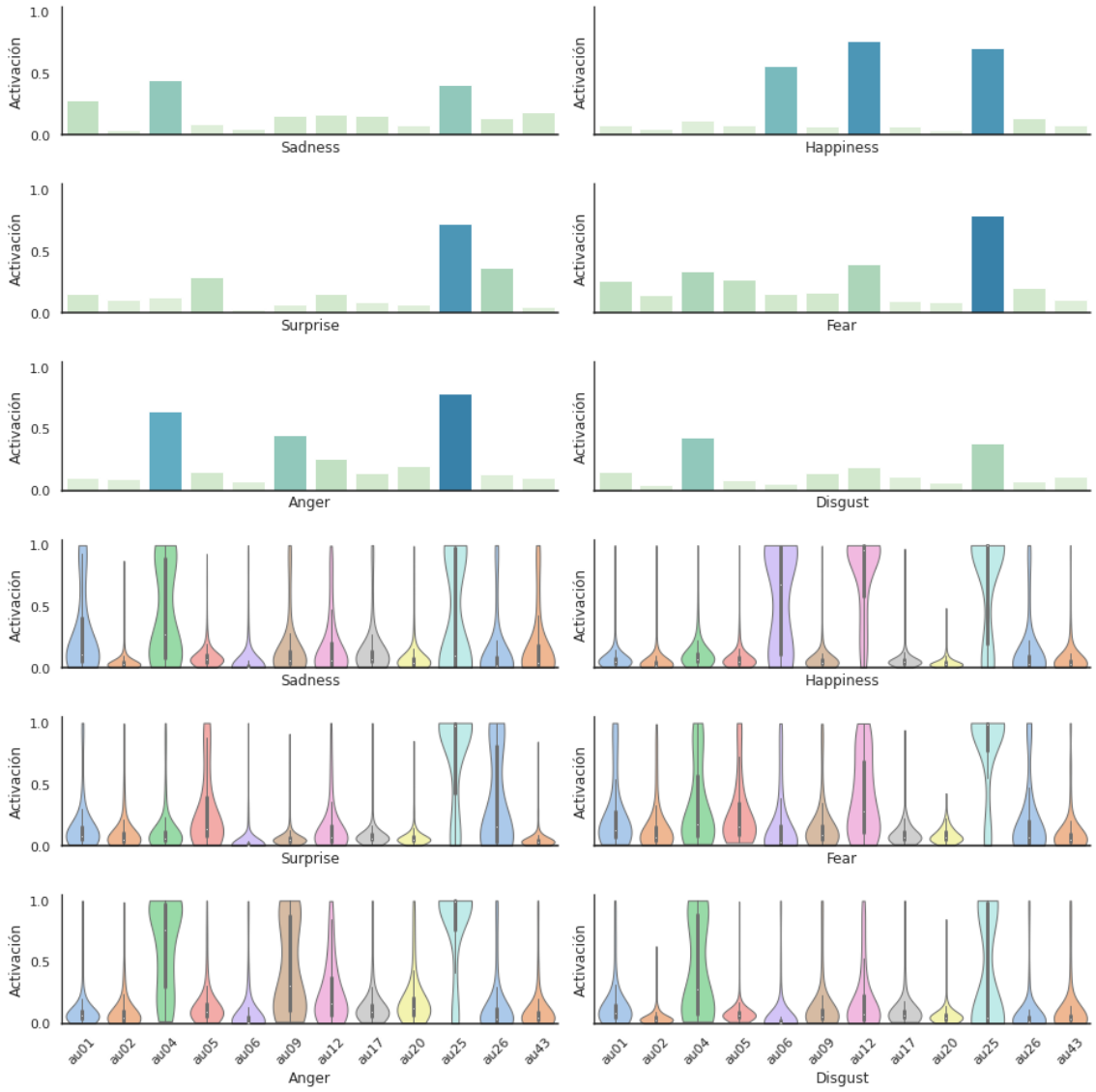


Fig. 4.9: Intensidad promedio por Action Unit según tipo de emoción utilizando MLCR sobre el dataset RAF-DB (arriba) y distribución de intensidades por Action Unit según tipo de emoción (abajo).

La situación para esta misma emoción en OpenFace es razonablemente buena ya que las AU 6, AU 12, AU 25 son las que más destacan a pesar de su baja intensidad. Sin embargo, están rodeadas de otras activaciones con intensidades similares como AU 14 y AU 10, está última resultando llamativa ya que junto con AU 25 suelen aparecer en disgust.

Para la otra emoción positiva, surprise, se esperaría la activación de AU 1, AU 2, AU 5 y AU 26. Esta vez más, viendo la Figura 4.9, notamos la activación anómala de AU 25 (lips part) que además es la más intensa. Es posible que esto ocurra por que AU 26 (jaw drop), en algunos casos, puede inducir la activación de AU 25, y ser un caso complejo de detectar. También es llamativo que las activaciones estén todas por debajo de 0.5 y que tanto AU 1 como AU 2 no destaquen lo suficiente.

Tomando esta misma emoción en OpenFace y observando Figura 4.10 vemos que, a pesar del problema de las bajas activaciones, todas las Action Units que destacan son las que se esperan (con excepción de AU 25).

Teniendo en cuenta ahora las emociones negativos, encontraremos una situación común para ambos métodos: tanto sadness como disgust tienen activaciones promedio muy bajas para la mayoría de sus AU, en ningún caso sobrepasan los 0.5. Además, de las AUs esperadas en disgust, la única que aparece con cierta relevancia es AU 25 pero solo en MLCR.

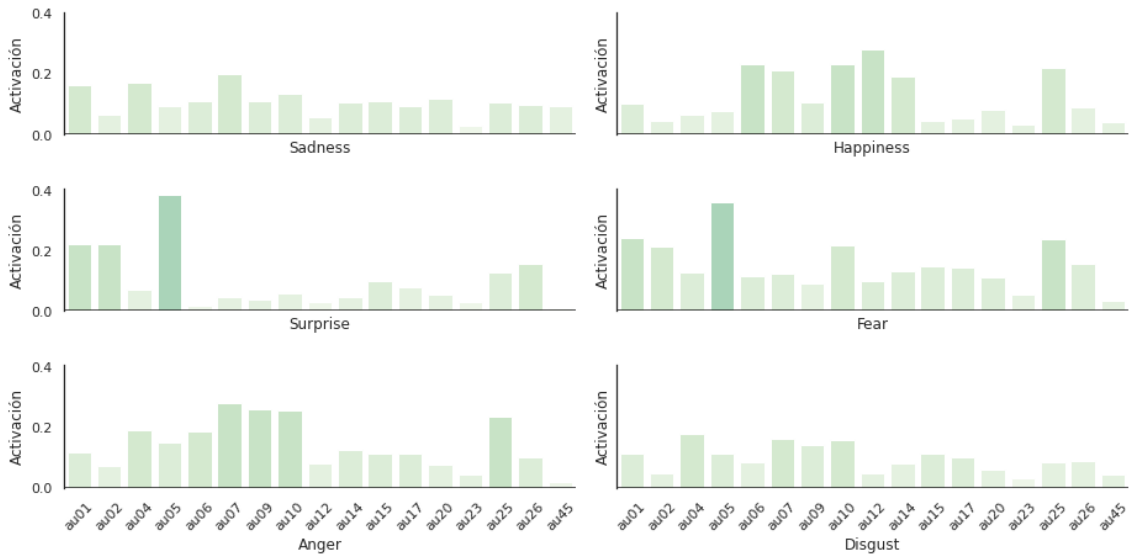


Fig. 4.10: Intensidad promedio por Action Unit según tipo de emoción utilizando Openface sobre el dataset RAF-DB.

En la categoría de emociones negativas encontramos mayor cantidad de activaciones que pueden ser inconsistentes: por ejemplo en MLCR, fear tiene AU 12 (lip corner puller) como segunda en intensidad promedio, pero no está asociada a la emoción prototípica ni sus variantes más comunes. Lo mismo ocurre con la presencia marcada de AU 25 en anger y fear.

Una de las características de las activaciones de Action Units en general es que existen pares que suelen activarse típicamente en simultaneo, por ejemplo AU 12 (lip corner puller) y AU 6 (cheek raiser) presentes en lo que se conoce como sonrisa genuina (o de Duchenne) o bien otras como AU 9 (nose wrinkler) y AU 4 (brow lowerer).

Como referencia, en la Figura 4.11 se muestran las correlaciones del dataset BP4D y DISFA cuyas imágenes fueron anotadas con AU manualmente.

Luego, dadas las matrices de correlación de las anotaciones obtenidas para RAF-DB bajo los dos métodos (ver la Figura 4.12) corroboramos que efectivamente AU 6+12 es la combinación que mayor correlación tiene. Lo mismo ocurre con otras que deberían destacar según Figura 4.11: AU 4+9 y AU 12+25, aunque en OpenFace resultan un poco más débil, y AU 1+2, más débil en MLCR. De las correlaciones negativas más fuertes que se observan en otros datasets: AU 25+2 y AU 25+4 aparecen muy poco correlacionadas y de forma positiva, siendo esto un caso desfavorable en ambos métodos.

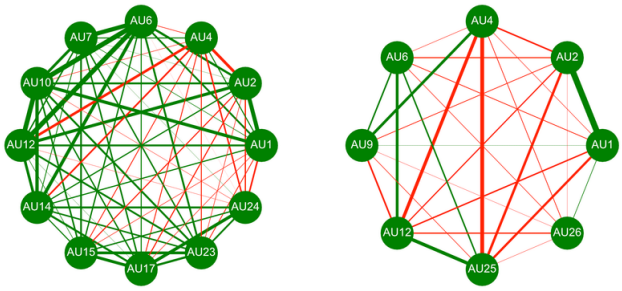


Fig. 4.11: Grafo de correlaciones entre Action Units en dataset BP4D[2].

Considerando todas las observaciones hechas sobre el problema de la anotación de Action Units sobre los datasets in-the-wild de nuestro estudio, concluimos con una mirada general.

Las calidad de las notaciones de AU sobre el dataset RAF-DB depende fuertemente del tipo de emoción. Las positivas, en particular happiness, muestran una buena representación, considerablemente fiel según lo que se espera de estas emociones prototípicas en [40]. Lamentablemente, ambas herramientas tienen problemas para representar adecuadamente a las emociones negativas e introducen inconsistencias con las cuales hay que lidiar. Entre ambos métodos, MLCR muestra una característica favorable estimando intensidades considerablemente más altas para las AU principales.

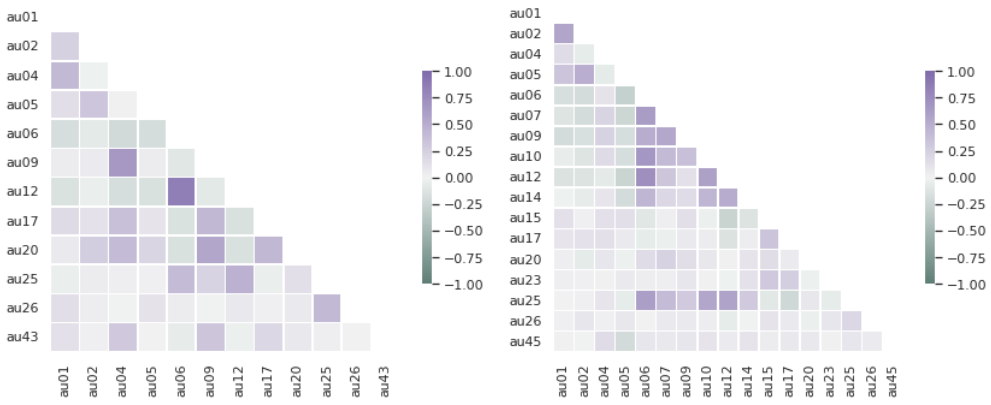


Fig. 4.12: Matriz de correlaciones entre Action Units sobre el dataset RAF-DB según las anotaciones de MLCR (izq) y Openface (der).

4.5.2. Resultados

Volviendo sobre la necesidad de definir una topología sobre el espacio de etiquetas auxiliar, notamos que aún eligiendo una tarea adecuada, conseguir una buena representación para el espacio auxiliar es complejo para bajo situaciones in-the-wild con condiciones tan cambiantes. Sin embargo, es claro que el propósito no es conseguir anotar features que discriminen de la mejor manera a las imágenes de expresiones faciales según las emociones categóricas, si no conseguir un espacio suficientemente suave como herramienta para estabilizar el aprendizaje de los modelos lightweight en presencia de instancias con anotaciones primarias con un alto grado de ambigüedad.

Al considerar la topología del espacio de etiquetas de Action Units para hacer más robusto el entrenamiento con Label Distribution Learning, inicialmente se propusieron dos acercamientos: El directo, que adaptaba la función de pérdida de EfficientFace y el indirecto, que adaptaba la del LDG. Nuestra intención era de esta manera conseguir potenciar el aprendizaje de EfficientFace via un LDG potencialmente mejorado (LDG-AUS).

Los resultados de nuestra experimentación sobre el dataset RAFDB y FER+ mostraron que el LDG-AUS equiparaba el accuracy del LDG baseline pero nunca obtuvo una mejora sustancial. Aunque lo hubiera conseguido, por lo visto en el experimento de la Sec.4.4.3, el retorno de acurracy debería ser negativo y al estar por encima del umbral de entrenamiento ideal el impacto sobre el modelo lightweight debería ser prácticamente nulo.

Por otro lado, el método directo tampoco pudo conseguir mejores resultados en RAFDB y FER+, cayendo por debajo de lo reportado en 4.2 por aproximadamente $1\,\%$ y $0,6\,\%$ en RAF-DB y FER+ respectivamente.

Teniendo en cuenta la discusión de la Sec.4.5.1, es posible que la calidad de anotaciones obtenidas haya influido en la construcción de un espacio que no fue lo suficientemente coherente para poder auxiliar al entrenamiento con los margenes de mejora esperados. La situación también amerita replantear la definición de la función de pérdida como otra alternativa a explorar.

[2] Fuente: Explainable Early Stopping for Action Unit Recognition.

4.6. Tolerancia a inconsistencias de anotaciones

Con el fin de evaluar la robustez de la técnica de Label Distribution Learning al ruido presente en las anotaciones de emociones, tomamos al framework de EfficientFace en tres escenarios distintos, cada uno con un nivel de ruido controlado. Para eso, se construyeron dos datasets alternativos con ruido sintético, es decir, eligiendo aleatoriamente un subconjunto del total de instancias y alterando la etiqueta de emoción por alguna otra elegida al azar pero distinta de la anterior.

En primer lugar se entrenó como baseline a la arquitectura de EfficientFace con Single Label Learning (EF SLL) para cada uno de los escenarios. Para hacer lo mismo con EF LDL se entrenó un LDG a partir de los datasets alterados. Aunque en el experimento de la Sec4.5 haya mostrado que el uso de la topología de AUS no beneficiaba lo suficiente al modelo, en términos de accuracy rate alcanzado, pensamos que ante un escenario con niveles de ruido más considerables, la técnica podría mostrarse más robusta que el entrenamiento SLL directo. Esto viene del hecho de que para una instancia afectada, se podría compensar y suavizar el efecto negativo si se llegara a considerar la información coherente de parte de su vecindad. Sin embargo, volvimos a comprobar que el LDG-AUS no obtuvo ninguna mejora notoria con respecto al LDG SLL. Decidimos tomar el LDG AUS pero podría haber sido el otro, ya que ambas curvas de entrenamiento se parecían.

Notar que el ruido introducido a los datasets no afecta directamente al entrenamiento directo de EfficientFace ya que el ground-truth utilizado es el estimado por el LDG. Si el generador se muestra lo suficientemente robusto para poder entrenar a pesar del nivel de ruido, entonces como se vio en la Tabla 4.4.3 de la Sec.4.4, debería permitir también un entrenamiento estable en EfficientFace con, potencialmente, un desempeño menor a causa de la pérdida de accuracy en el LDG. Por esta razón, también se esperaría que el resultado pueda generalizarse bien sobre otras arquitecturas como CERN.

Los resultados de la Tabla 4.6 confirman que, efectivamente, el método de Label Distribution Learning ayuda a tolerar el ruido presente en las anotaciones. Es remarcable como, con un 20 % de ruido presente en el dataset, EF LDL consigue entrenar manteniendo un desempeño razonable a comparación con la versión SLL. Por supuesto, como se esperaba, el LDG hizo posible esta situación al conseguir sostener su accuracy en $85,44\,\%$ y $83,11\,\%$ para los dos niveles de ruido.

Train data	Método	Accuracy (%)
Limpio	EF SLL	87,58
	EF LDL	**88,52**
10 % ruido	EF SLL	85,00
	EF LDL	**86,55**
20 % ruido	EF SLL	80,24
	EF LDL	**85,78**

Tab. 4.5: Tolerancia a tres niveles de ruido sintético sobre las etiquetas del dataset RAF-DB. EF SLL fue entrenada con un LDG-AUS entrenado para cada nivel de ruido.

4.7. Análisis de complejidad de las redes

Como se discutió en la Sec.3, comparar el costo computacional a través de FLOPs puede ser engañoso ya que esta es agnóstica al costo de acceso a memoria y al grado de paralelismo de las operaciones. Por lo tanto, arquitecturas con FLOPs comparables pueden tener velocidades distintas.

Para evaluar la complejidad de una manera directa, considerando que este tipo de arquitecturas pueden tener una plataforma objetivo con recursos acotados o bien pueden encontrarse en situaciones donde tengan restricciones en términos de velocidad, se tomaron 10 muestras del tiempo de ejecución de los modelos durante el proceso de inferencia sobre algún dataset. Estas corridas se hicieron con el modelo ejecutando en CPU o bien en GPU utilizando la herramienta Torch Profiler, bajo una computadora con una GPU Geforce 1080 Ti y un CPU Intel Core i9-9900K CPU @ 3.60GHz.

Al considerar el grado de paralelismo disponible en las GPUs, la medida que optamos por reportar es batches/s considerando batches de tamaño 16. Para el caso de la CPU se fuerza el batch a tamaño 1 y se reporta le medida de imágenes/s (o frames per second).

Método	#Params(M)	#FLOPs	Batch/s (GPU)	Imgs/s (CPU)	Mem, (MB)
ResNet50	23,52	4109,4	97,08(SD=7,45)	11,11 (SD=0,6)	85,57 MB
EF	1,28	154,1	65,65(SD=3,32)	45,91(SD=6,42)	26,34 MB
CERN	1,28	1781	97,93(SD=0,66)	12,97(SD=0,06)	68,39 MB

Tab. 4.6: Se reportan la media de 10 corridas para cada caso junto con el desvío estándar. El tamaño del batch quedó fijo en 16 para las corridas en GPU y en 1 para las de CPU. Con memoria se consigna el consumo en la etapa de inferencia según torch profiler.

Los resultados obtenidos muestran que, como se esperaba, EfficientFace consigue un consumo de memoria y velocidad considerablemente menor cuando se trata de ejecuciones en CPU, aproximadamente 4 veces más. Esto reafirma la capacidad de este modelo de desenvolverse en los contextos para los cuales fue diseñados. Pero más aún, vimos durante este trabajo que para algunos datasets de nuestro benchmark puede obtener mejores resultados que arquitecturas más complejas del estado del arte como ResNet-50.

Por otro lado, la velocidad de CERN queda a la par de ResNet-50 lo cual es sorpresivo, considerando los reportes del trabajo original [15]. Esto lo deja con la única ventaja a nivel consumo de memoria, pero por un margen menor al de EfficientFace.

Finalmente, corriendo torch profiler se pudo evaluar el porcentaje del tiempo de ejecución que insume cada una de las operaciones involucradas en la inferencia del modelo en CPU. Se agruparon las operaciones según su tipo, eligiendo poner como elemwise a aquellas que, según [33] tienen un alto ratio $\frac{MAC}{FLOPs}$ (por ejemplo, ReLu, AddTensor, e incluso depth-wise convolutions).

Probablemente lo más llamativo es que a pesar de que la arquitectura de base de EfficientFace fue diseñada para evitar penalizar en operaciones elemwise, sigue ocurriendo que el porcentaje del tiempo total insumido por estas es para nada despreciable, llegando casi al 40 %. En particular, las depth-wise convolutions que se encuentran dentro de este grupo representan el 65 % del tiempo.

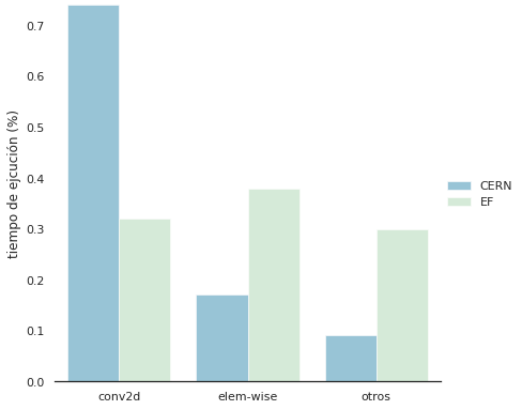


Fig. 4.13: Porcentajes del tiempo de ejecución según tipo de operación en CPU.

Interesantemente, los agregados para lidiar con el problema de FER en el contexto in-the-wild resultaron tomar tiempos similares. En EfficientFace, observamos que el tiempo insumido por el modulo de local feature extractor y del módulo de atención es de 3 ms y 1,28 ms respectivamente. Esto lo deja muy cerca de los 6,2 ms que necesita CERN para su bloque final, donde aplica los módulos ECA.

5. CONCLUSIONES

En este trabajo se abordó el problema de Facial Expression Recognition (FER) con arquitecturas de redes neuronales convolucionales lightweight, que intentan ajustarse al tradeoff entre costo computacional y capacidad de predicción del modelo cuando la plataforma objetivo cuenta con recursos acotados.

En particular, se enmarcó el análisis sobre datasets de FER in-the-wild que presentan condiciones de entorno (iluminación, oclusiones) y variaciones de pose sumamente cambiantes, haciendo este un problema desafiante incluso para redes más complejas del estado del arte. Además, estos datasets suelen traer aparejado el problema de las inconsistencias en las anotaciones de las emociones que puede sesgar el aprendizaje de los modelos impactando en el rendimiento final obtenido. Estas inconsistencias pueden venir de dos fuentes: la ambigüedad intrínseca que surge en la tarea de clasificar expresiones faciales en emociones y el sesgo de anotación propio del contexto cultural y habilidad de las personas.

Para el estudio fueron consideradas dos arquitecturas lightweight del estado del arte, EfficientFace [52] y CERN [15]. Ambas aprovechan tres técnicas que vienen demostrando ser claves para problemas de FER in-the-wild: la incorporación de mecanismos de atención para enfatizar regiones salientes para la tarea en cuestión, la captura de features representativas del contexto local y global, y el uso de transfer learning desde el dominio de Facial Recognition. Además, considerando el problema de la inconsistencia de anotaciones, en el trabajo original de EfficientFace se propone el uso de Label Distribution Learning (LDL) que permite entrenar modelos a partir de etiquetas que son la distribución de emoción asociada a cada imagen facial. Para ello se adopta un Label Distribution Generator (LDG) como modelo para aprender a generar este tipo de anotaciones que luego puedan utilizarse en el entrenamiento de la red lightweight en cuestión.

Mediante el benchmark propuesto en este trabajo, el cual extiende a los trabajos originales, se consiguió evaluar el desempeño de las arquitecturas en su versión Single Label Learning bajo distintas condiciones. El experimento de la Sec. 4.3 corroboró los resultados reportados sobre los datasets in-the-wild de los trabajos de CERN y EfficientFace, mostrando que pueden alcanzar un accuracy rate equiparable a arquitecturas del estado del arte de mayor complejidad, como las ResNet-50, incluso en condiciones de entorno complejas. Si bien ambas arquitecturas lograron resultados comparables, EfficientFace mostró destacarse por sobre CERN en CAER-S y RAF-DB. Por último, los valores máximos de accuracy hallados durante el entrenamiento para cada modelo son, en la mayoría de los casos, considerablemente mejores que los reportados en los trabajos.

El estudio de ablación del experimento 4.4 para evaluar la eficiencia de Label Distribution Learning en este contexto, acompañó los resultados de [52] sobre EfficientFace, mostrando que efectivamente la incorporación de LDL via LDG mejora el comportamiento del modelo en términos del accuracy rate a lo largo de todos los datasets del benchmark. Sin embargo, los márgenes de ganancia en nuestra EfficientFace no son tan abultados como se esperaban, consiguiendo aproximadamente subas de $1\,\%$ y una suba de $0,75\,\%$ promedio. La situación de CERN, sobre la cual se introdujo el LDG como novedad respecto del trabajo original, exhibe mejoras que son mucho menos evidentes en general, con una suba promedio de $0,44\,\%$ en accuracy.

Al incorporar las anotaciones de Action Units para definir la topología del espacio auxiliar de etiquetas, se esperaba hacer más robusta la técnica de Label Distribution Learning a través de la adaptación de la función de pérdida para considerar la información de los vecinos relevantes a cada instancia. A pesar de considerar anotaciones automáticas de AUs provenientes de distintos métodos y evaluar algunas variantes para la modificación de la función de pérdida, los resultados del experimento de la Sec. 4.5 no evidencian mejora alguna para los datasets considerados en este caso, RAF-DB y FER+. Se piensa que esto puede deberse principalmente al ruido inducido por las anotaciones automáticas, no permitiendo establecer una topología lo suficientemente suave para auxiliar al aprendizaje.

Los resultados del experimento 4.6 muestran que el método de Label Distribution Learning es robusto ante presencia de porcentajes notorios de ruido en anotaciones de etiquetas sobre el dataset RAF-DB. Sin embargo, parte de esa capacidad se la debe al tipo de red utilizada como generador.

Por último, obtuvimos una comparación más directa de la complejidad de las arquitecturas estudiadas a partir de las mediciones de velocidad (expresadas en imágenes/segundo o bien batches/segundo según sea el caso de ejecución en CPU o GPU) realizadas en la Sec. 4.7. Lo más destacable es que EfficientFace consigue un consumo de memoria y velocidad aproximadamente 4 veces mayor en CPU que una arquitectura compleja como ResNet-50, y lo hace sin penalizar significativamente la capacidad de predicción. De hecho, se vio a lo largo de nuestro benchmark que para algunos datasets puede obtener mejores resultados. Contrario a esto, la velocidad de CERN queda a la par de ResNet-50 lo cual es sorpresivo, considerando las intenciones de diseño de cada arquitectura. La única ventaja que admite es a nivel consumo de memoria, pero por un margen menor al de EfficientFace.

Durante el estudio de este trabajo se pudo hacer una comparación de dos arquitecturas lightweight sobre varios datasets para el problema de FER in-the-wild. Los resultados confirman que el desempeño obtenido, medido como accuracy rate, es equiparable al de modelos más complejos del estado del arte. No solo eso, si no que, en particular para EfficientFace, se logran velocidades de inferencia y consumo de memoria considerablemente mejores. Además, con la incorporación de Label Distribution Learning se pueden conseguir mejoras de hasta 1 %, dependiendo del dataset y arquitectura, y una mejor adaptación al ruido en las anotaciones sin agregar complejidad a la arquitectura.

Como trabajo a futuro se espera continuar explorando la dirección iniciada en esta tesis, inspirada en [5], para aprovechar el espacio de Action Units en Label Distribution Learning. Podrían considerarse otras funciones de pérdida que capturen la información de la distribución de los vecinos o bien refinar de alguna manera las anotaciones automáticas de AUs. Por ejemplo, una posibilidad es hacer detección de anomalías con modelos como One-Class SVM. De esta forma se podría desestimar este tipo de datos para la construcción del grafo de vecinos pero mantenerlos como parte del datase.

Si bien lo anterior permitiría continuar mejorando a las arquitecturas sin agregarles complejidad adicional, podría experimentarse con el tradeoff de costo computacional y capacidad de predicción a través de mecanismos de atención. Por ejemplo, el costo introducido por cada módulo ECA en CERN es mucho menor al módulo BAM de EfficientFace, por lo que podría considerarse su adopción.

Finalmente, el benchmark propuesto podría extenderse con conjuntos de test que son específicamente difíciles y que permiten evaluar la robustez de los modelos. Junto con esto, también sería deseable evaluar la generalización cross-dataset, en particular al reforzar a los modelos con técnicas de LDL.

Bibliografía

[1] Baltrušaitis, T., Mahmoud, M., Robinson, P.: Cross-dataset learning and person-specific normalisation for automatic action unit detection. In: 11th IEEE Int. Conf. and Workshops on Aut. Face and Gesture Recognition (FG). vol. 06, pp. 1–6 (2015)

[2] Barsoum, E., Zhang, C., Ferrer, C.C., Zhang, Z.: Training deep networks for facial expression recognition with crowd-sourced label distribution. p. 279–283. ICMI '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2993148.2993165

[3] Bengio, Y.: Practical recommendations for gradient-based training of deep architectures (2012). https://doi.org/10.48550/ARXIV.1206.5533

[4] Burrows, A., Cohn, J.F.: Comparative anatomy of the face. in handbook of biometrics, 2nd ed. springer. **20**, 1–10 (2014)

[5] Chen, S., Wang, J., Chen, Y., Shi, Z., Geng, X., Rui, Y.: Label distribution learning on auxiliary label space graphs for facial expression recognition. In: IEEE/CVF Conf. on Computer Vision and Pattern Recog. (CVPR). pp. 13981–13990 (2020)

[6] Chen, W., Zhang, D., Li, M., Lee, D.J.: Stcam: Spatial-temporal and channel attention module for dynamic facial expression recognition. IEEE Trans. on Affective Computing (2020)

[7] Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Conf. on Comp. Vision and Patt. Recog. (CVPR). pp. 1800–1807. USA (2017)

[8] Cohn, J.F., Ambadar, Z., Ekman, P.: Observer-based measurement of facial expression with the facial action coding system. (2007)

[9] Corneanu, C.A., Simón, M.O., Cohn, J.F., Guerrero, S.E.: 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 (2016). https://doi.org/10.1109/TPAMI.2016.2515606

[10] Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data (2018). https://doi.org/10.48550/ARXIV.1805.09501

[11] Ekman, P.: Universal and cultural differences in facial expression of emotion (1971)

[12] Ekman, P., Friesen, W.V.: Facial action coding system: a technique for the measurement of facial movement (1978)

[13] Frenay, B., Verleysen, M.: Classification in the presence of label noise: A survey. IEEE Transactions on Neural Networks and Learning Systems **25**(5), 845–869 (2014). https://doi.org/10.1109/TNNLS.2013.2292894

[14] Gao, B.B., Xing, C., Xie, C.W., Wu, J., Geng, X.: Deep label distribution learning with label ambiguity. IEEE Transactions on Image Processing **26**(6), 2825–2838 (jun 2017). https://doi.org/10.1109/tip.2017.2689998

[15] Gera, D., Balasubramanian, S., Jami, A.: Cern: Compact facial expression recognition net. Pattern Recognition Letters **155**, 9–18 (2022). https://doi.org/10.1016/j.patrec.2022.01.013

[16] Guo, Y., Zhang, L., Hu, Y., He, X., Gao, J.: Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In: Computer Vision – ECCV. pp. 87–102 (2016)

[17] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conf. on Comp. Vision and Pattern Recog. (CVPR). pp. 770–778 (2016)

[18] Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network (2015)

[19] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017). https://doi.org/10.48550/ARXIV.1704.04861

[20] Hu, J., Shen, L., Albanie, S., Sun, G., Wu, E.: Squeeze-and-excitation networks (2017). https://doi.org/10.48550/ARXIV.1709.01507

[21] Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks (2016). https://doi.org/10.48550/ARXIV.1608.06993

[22] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR **abs/1502.03167** (2015)

[23] Kanade, T., Cohn, J., Tian, Y.: Comprehensive database for facial expression analysis. In: Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580). pp. 46–53 (2000). https://doi.org/10.1109/AFGR.2000.840611

[24] Khan, S.A., Hussain, S., Xiaoming, S., Yang, S.: An effective framework for driver fatigue recognition based on intelligent facial expressions analysis. IEEE Access **6**, 67459–67468 (2018)

[25] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014). https://doi.org/10.48550/ARXIV.1412.6980

[26] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (may 2017). https://doi.org/10.1145/3065386

[27] Lee, J.R.H., Wang, L., Wong, A.: Emotionnet nano: An efficient deep convolutional neural network design for real-time facial expression recognition (2020). https://doi.org/10.48550/ARXIV.2006.15759

[28] Lee, J., Kim, S., Kim, S., Park, J., Sohn, K.: Context-aware emotion recognition networks (2019). https://doi.org/10.48550/ARXIV.1908.05913

[29] Li, B., Mehta, S., Aneja, D., Foster, C., Ventola, P., Shic, F., Shapiro, L.: A facial affect analysis system for autism spectrum disorder. In: IEEE Int. Conf. on Image Processing (ICIP). pp. 4549–4553 (2019)

[30] Li, S., Deng, W., Du, J.: Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2584–2593 (2017). https://doi.org/10.1109/CVPR.2017.277

[31] Lin, M., Chen, Q., Yan, S.: Network in network (2013). https://doi.org/10.48550/ARXIV.1312.4400

[32] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J.: On the variance of the adaptive learning rate and beyond (2019). https://doi.org/10.48550/ARXIV.1908.03265

[33] Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn arch. design. In: Proc. of Eur. Conf. on Comp. Vision (ECCV) (2018)

[34] Maat, L., Pantic, M.: Gaze-x: adaptive affective multimodal interface for single-user office scenarios. In: International Conference on Multimodal Interaction (2006)

[35] van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(86), 2579–2605 (2008)

[36] Malkov, Y., Ponomarenko, A., Logvinov, A., Krylov, V.: Approximate nearest neighbor algorithm based on navigable small world graphs. Information Systems **45**, 61–68 (2014). https://doi.org/https://doi.org/10.1016/j.is.2013.10.006

[37] Martinez, B., Valstar, M.F., Jiang, B., Pantic, M.: Automatic analysis of facial actions: A survey. IEEE Transactions on Affective Computing **10**(3), 325–347 (2019). https://doi.org/10.1109/TAFFC.2017.2731763

[38] Mollahosseini, A., Hasani, B., Mahoor, M.H.: AffectNet: A database for facial expression, valence, and arousal computing in the wild. IEEE Transactions on Affective Computing **10**(1), 18–31 (jan 2019)

[39] Niu, X., Han, H., Shan, S., Chen, X.: Multi-label co-regularization for semi-supervised facial action unit recognition. CoRR **abs/1910.11012** (2019)

[40] P. Ekman, W.V.F., Hager, J.C.: Facial action coding system: The manual on cd rom. a human face (2002)

[41] Park, J., Woo, S., Lee, J.Y., Kweon, I.S.: Bam: Bottleneck attention module (2018). https://doi.org/10.48550/ARXIV.1807.06514

[42] Plutchik, R.: Chapter 1. a general psychoevolutionary theory of emotion. In: Plutchik, R., Kellerman, H. (eds.) Theories of Emotion, pp. 3–33. Ac. Press (1980)

[43] Schmidt, K.L., Cohn, J.F.: Human facial expressions as adaptations: Evolutionary perspectives in facial expression research.yearbook of physical anthropology **116**, 8–24 (2001)

[44] Shariff A. F., Tracy, J.L.: What are emotion expressions for current directions in psychological science **20**(6), 395–399 (2011). https://doi.org/10.1177/096372141142473

[45] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014). https://doi.org/10.48550/ARXIV.1409.1556

[46] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting **15**(1), 1929–1958 (jan 2014)

[47] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions (2014). https://doi.org/10.48550/ARXIV.1409.4842

[48] Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W., Hu, Q.: Eca-net: Efficient channel attention for deep convolutional neural networks (2019). https://doi.org/10.48550/ARXIV.1910.03151

[49] Wu, X., He, R., Sun, Z., Tan, T.: A light cnn for deep face representation with noisy labels (2015). https://doi.org/10.48550/ARXIV.1511.02683

[50] Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters **23**(10), 1499–1503 (oct 2016). https://doi.org/10.1109/lsp.2016.2603342

[51] Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices (2017). https://doi.org/10.48550/ARXIV.1707.01083

[52] Zhao, Z., Liu, Q., Zhou, F.: Robust lightweight facial expression recognition network with label distribution training. Proc. of the AAAI Conf. on Artificial Intelligence **35**(4), 3510–3519 (2021)

[53] Zhou, S., Wu, X., Jiang, F., Huang , Q., Huang, C.: Emotion recognition from large-scale video clips with cross-attention and hybrid feature weighting neural networks. International Journal of Environmental Research and Public Health **20**(2) (2023). https://doi.org/10.3390/ijerph20021400

[54] Zhou, Y., Xue, H., Geng, X.: Emotion distribution recognition from facial expressions. In: Proceedings of the 23rd ACM International Conference on Multimedia. p. 1247–1250. MM '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2733373.2806328

[55] Zhou, Y., Xue, H., Geng, X.: Emotion distribution recognition from facial expressions. p. 1247–1250. Assoc. for Computing Machinery, NY, USA (2015)

[56] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning (2019). https://doi.org/10.48550/ARXIV.1911.02685