Facial Expression Recognition using lightweight deep convolutional networks with Label Distribution Learning on Action Units labels space

Nicolas Mastropasqua[1,2] and Daniel Acevedo[1,2]

[1] Departamento de Computación, Fac. de Cs. Exactas y Naturales, Universidad de Buenos Aires, Ciudad de Buenos Aires, Argentina.
[2] Instituto de Investigación en Cs. de la Computación (ICC). CONICET-UBA. Ciudad de Buenos Aires, Argentina.
{nmastropasqua,dacevedo}@dc.uba.ar

Abstract. 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 areas 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 article 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 an analysis of EfficientFace, a state-of-the-art ligthweight CNN and we analyze the impact of using different approaches to LDL on a variety of in-the-wild datasets: RAF-DB, CAER-S, FER+ and AffectNet.

Keywords: Facial expression recognition · Label distribution learning · Lightweight · Convolutional Neuronal Network · Action unit recognition.

Reconocimiento de expresiones faciales con redes profundas livianas usando Label Distribution Learning y el espacio de Action Units

Resumen 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 la arquitectura lightweight EfficientFace 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 Clave: Reconocimiento de expresiones faciales · Aprendizaje de distribuciones de etiquetas · Redes convolucionales livianas · Reconocimiento de Action Units

1 Introducción

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 [17], el reconocimiento de estados emocionales en robots sociales [23,6] y la clasificación del trastorno del espectro autista [20] entre tantos otros.

Dentro del mundo de la detección automática de expresiones es necesario definir algún tipo de codificación para describir a 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 [7], donde se distinguen seis emociones básicas: Felicidad, tristeza, miedo, enojo, asco y sorpresa. Por otro lado, 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 [8] que codifica cada posible movimiento de 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 por ejemplo EMFACS [28] que define a las categorías de emociones a partir de las AUs.

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 caso se suele alimentar a los modelos a partir de fotogramas extraídos de puntos claves de la secuencia, 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, también llamado 'in-the-wild'), la complejidad de la escena que rodea al sujeto y la presencia de oclusiones.

Fig. 1. En la fila superior, muestras del dataset Cohn-Kanade tomadas en condiciones de laboratorio. 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 éste el obtenido como representante. En la fila inferior, muestras del dataset in-the-wild FER+ cuyas imágenes fueron obtenidas de Internet y anotadas mediante crowd-sourcing. Ambas filas se ordenan por emoción: Enojo, Desprecio, Asco, Miedo, Felicidad, Sorpresa, Tristeza, Neutral.

1.1 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 [18] 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. 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 [31], GoogLeNet [34], ResNet [12] y DenseNet [15] 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 MobileNet [13], ShuffleNet [22] por nombrar algunas. Sin embargo, todas estas redes del estado del arte también se ven perjudicadas cuando se las evalúa en escenarios in-the-wild de mayor complejidad.

Según Gera et al. [10] 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* [40] 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. [37] también surge de arquitecturas lightweight adaptadas para el contexto in-the-wild y logra resultados alentadores en datasets como RAF-DB [21] y CAER-S [19]. Esta arquitectura aprovecha el diseño de ShuffleNet v2 [22] como red base, agregando algunos bloques convolucionales de atención inspirados en BAM [29] y bloques para combinar features locales-globales para ser más robusta ante cambios en las poses y en oclusión 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, emplean Label Distribution Learning (LDL) [38,9], 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 [30]. Relacionado con esto, en Chen et al. [3] se utiliza esta misma estrategia pero incorporando información directamente de la topología de un espacio de etiquetas descriptivas auxiliares, como Action Units [8] o Landmarks, para tratar de abordar el problema del entrenamiento con anotaciones inconsistentes.

1.2 Objetivos

El artículo se centra en el análisis de EfficientFace como arquitectura lightweight para el problema de FER in-the-wild. La primera hipótesis que se tiene es que la

misma puede alcanzar un desempeño equiparable a redes más complejas sobre múltiples datasets. Para ello, como objetivo inicial intentamos replicar los resultados obtenidos en el trabajo original [37] usando una extensión del benchmark presentando allí, considerando los datasets RAF-DB [21], CAER-S [19], FER+ [2] y AffectNet [26].

Luego se estudia el uso Label Distribution Learning (LDL) [38,9] 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 la arquitectura y hacer el entrenamiento más robusto al ruido en las anotaciones. Buscamos obtener resultados similares a los publicados en el trabajo original donde se exhiben mejoras entre 1% y 2% sobre RAF-DB Y CAER-S. A su vez, basándonos en el trabajo de Chen et al. [3], intentamos 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 con una medida más directa que los FLOPs como por ejemplo la velocidad de inferencia.

El resto del artículo se estructura como sigue: En la Sec. 2 se hace una breve revisión de los métodos más relevantes para este trabajo: 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 su aplicación al problema de FER in-the-wild. Por último, en la Sec. 4 se discuten los experimentos ensayados junto con sus resultados y en Sec. 5 se exponen las conclusiones del trabajo junto con posibles líneas a futuro.

2 Trabajos relacionados

2.1 Arquitecturas convolucionales lightweight

Desde la aparición de AlexNet [18] en 2012 comenzó la tendencia de hacer redes convolucionales cada vez más profundas y más anchas convergiendo en arquitecturas como VGG 2014 [31] con aproximadamente 138M parámetros.

Estos acercamientos significaron una forma directa de mejorar el desempeño general de dichas arquitecturas pero a costa de aumentar su complejidad considerablemente. Más aún, el crecimiento desmesurado de la cantidad de parámetros entrenables alentó situaciones indeseables como las de overfitting. 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 [33] o batch normalization [16] y otras técnicas de regularización permitieron mejorar esta situación.

En 2017 Chollet et al. [5] propuso la arquitectura Xception que modificaba los módulos Inception bajo la hipótesis de que las correlaciones cruzadas entre-canales y las espaciales están lo suficientemente desacopladas que pueden ser mapeadas completamente separadas. Esto permitió factorizar las operaciones de convolución 3D y conseguir rendimientos similares pero con una menor cantidad de FLOPs y parámetros. El módulo Xception resulta similar a lo que se conoce como depth-wise separable convolution. La diferencia está, principalmente, en el orden de las operaciones: éstas aplican primero las convoluciones espacialmente y luego las 1×1 para las correlaciones entre-canales. Se puede ver que los FLOPs de esta operación se reducen por un factor $\frac{1}{N} + \frac{1}{k^2}$ siendo N la cantidad de canales de salida y k el tamaño del kernel.

La operación más costosa en las depth-wise separable convolutions son las convoluciones 1×1. En la arquitectura ShuffleNet V1 [36] se hizo especial énfasis en esta cuestión y se propuso modificar los bottleneck modules de las arquitecturas ResNet [12] reemplazando las convoluciones para extracción de features (típicamente con kernels de 3×3 o 5×5) por su versión depth-wise y las convoluciones 1×1 por group convolutions, introducidas en la arquitectura AlexNet para paralelizar el cómputo en múltiples GPUs. Partiendo de la separación en g grupos de un input feature map, se pueden reducir los FLOPs y cantidad de parámetros, comparado contra una convolución tradicional, en un factor que depende linealmente de g. A su vez, para favorecer el flujo de información entre grupos a lo largo de la arquitectura se agregaron operaciones de channel shuffling a cada ShuffleNet module.

Luego en el trabajo de ShuffleNet V2 [22] se observó teóricamente que el costo de acceso a memoria (MAC) de las convoluciones 1×1 se minimiza 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 se acercaba a uno, el MAC se vuelve más chico. De esta forma, abandonaron el concepto de módulo bottleneck para favorecer esta premisa. Además, aunque dadas restricciones de FLOPs el uso de group convolutions da lugar para conseguir mayor profundidad en la red, el incremento en la cantidad de capas también trae aparejado un incremento en MAC. Así, se volvió a la idea original de usar convoluciones 1×1 tradicionales pero se implementó un channel split que separa el input en dos grupos para bajar el costo. La comparación entre los distintos ShuffleNet modules puede observarse en Figura 2. En conjunto, estas revisiones logran una mejoría no solo en el desempeño de su predecesora si no una complejidad menor.

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. [14] donde se introduce 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. La aparición de variantes proliferó y potenció el uso de la atención en este tipo de redes y particularmente en

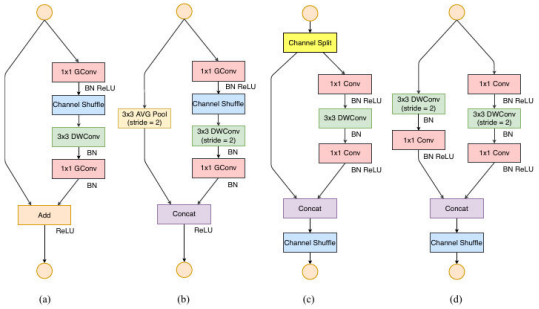

Fig. 2. (a) ShuffleNet Unit V1, (b) ShuffleNet Unit v1 con spatial downsampling, (c) ShuffleNet Unit V2 y (d) ShuffleNet Unit V2 con spatial downsampling. [22]

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. Además, la incorporación de estos módulos puede lograr mejoras significativas en el desempeño de las redes a un bajo costo. Los módulos Squeeze and Excitation [14] fueron uno de los primeros

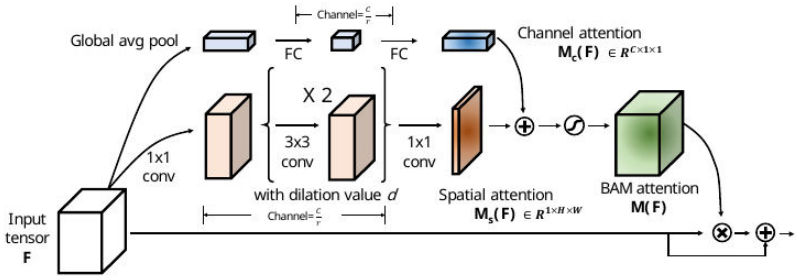

Fig. 3. Dado el feature map intermedio F, el módulo BAM calcula el mapa de atención correspondiente $M(F)$ a través de las dos ramas de atención, espacial y de canales cuyo objetivo es modular los todos los pesos de F.

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. En la etapa de squeeze, a partir de un input feature map con C canales se obtiene un vector de tamaño $1 \times 1 \times C$ mediante Global Average Pooling que agrega la información por canal. Luego en la etapa de excitation se mapea este vector a un vector de pesos por canal a partir de dos capas totalmente conectadas y una última aplicación

de función sigmoidea. Finalmente, este vector de pesos se utiliza para escalar los canales del input feature map. Los Bottleneck Attention Modules (BAM) [29] surgen 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 sobre los módulos Squeeze and Excitation tiene que ver con la falta de extracción de información relevante a la dimensión espacial. Concretamente, BAM descompone el input en dos ramas (ver Figura 3), una que resalta features entre-canales (al estilo Squeeze and Excitation) y otra espacial. Esta última aplica un bloque estilo bottleneck que en última instancia termina por generar un feature map con un solo canal que respeta las dimensiones espaciales originales y representa el peso de cada unidad espacial original.

2.3 Inconsistencias de etiquetas en FER

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 de un dataset, ya que puede sesgar el aprendizaje y potencialmente disminuir el rendimiento final. Este sesgo es evidente entre datasets como se ilustra en la Figura 4, donde dos grupos de instancias de los datasets AffectNet y RAF-DB [21] que podrían ser percibidas como muy similares, tienen etiquetas distintas ('fear' y 'disgust' respectivamente).

En particular para el caso de FER con anotaciones de categorías dadas por las emociones básicas, una de las principales fuentes de inconsistencias tiene que ver con la naturaleza ambigua de las expresiones faciales en sí. Según [30], 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. 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 y crowd-sourcing .

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. Utilizando la distribución de emociones, 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 [39], debería ser más robusta a las inconsistencias que surgen en FER a la vez que permite extraer mayor rendimiento que Multi Label Learning.

El problema de este acercamiento es que, justamente, tener disponible la distribución de emociones no es algo habitual en los datasets usados en FER.

Un enfoque para el mismo podría ser construir un modelo generador que pueda aprender a estimar las distribuciones de emociones como en el trabajo propuesto por Zhao et al. [37] y luego utilizar estas salidas, soft labels, como las etiquetas del ground-truth al momento de entrenar el modelo lightweight. Este acercamiento puede enmarcarse como una estrategia de *knowledge distilattion*, donde se utiliza una red de mayor complejidad que puede aprender mejores representaciones para luego entrenar otra red más chica. Con un enfoque similar a Label Distribution Learning, Chen et al. [3] propuso 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 su vecindad en 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, proponer 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 de 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.

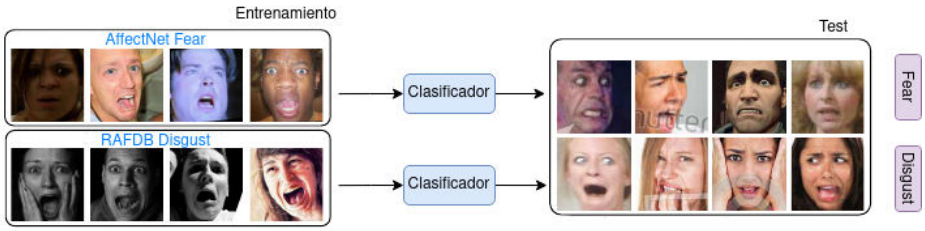

Fig. 4. Casos de sesgo de anotación entre datasets donde el grupo de instancias de la izquierda tienen expresiones muy similares entre sí. Sin embargo, las de la fila superior son etiquetadas como Fear en AffectNet y las de la fila inferior como Disgust en RAF-DB. Esta situación produce un sesgo en la clasificación de los modelos ante instancias no vistas y dificulta el nivel de generalización cross-dataset.

3 Metodología

3.1 El framework de estudio

Habiendo repasado la teoría detrás de algunos diseños ejemplares de arquitecturas lightweight y habiendo considerado las distintas técnicas que suelen acompañar el entrenamiento de las mismas, este trabajo propone el estudio del framework del estado del arte EfficientFace presentado en Zhao et al. [37] cuya arquitectura parte de la ya mencionada ShuffleNet v2 [22]. En definitiva, las ideas

de diseño son las discutidas en la Sec. 2: El uso de arquitecturas eficientes, la incorporación de mecanismos de atención para enfatizar regiones relevantes para FER in-the-wild, el uso de estrategias para capturar y aprovechar la información del contexto local y global y transfer learning [40] a partir de la tarea de reconocimiento facial 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.

El esquema general del framework de estudio puede verse en la Figura 5. En concreto, para adaptarse a escenarios in-the-wild se usa un extractor de features locales colocado al comienzo del pipeline de la arquitectura. 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 [29] sobre el feature-map de salida de la primera etapa. Finalmente, combinando la salida de estos dos módulos se computan features locales-globales que luego siguen su curso por el resto de las etapas. Según el estudio de ablación realizado en [37], 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%

Adicionalmente incorporamos transfer learning [40] tomando el reconocimiento facial como tarea original junto con el dataset MS-CELEB [11] como dominio. De esta forma, para entrenar EfficientFace sobre FER, la tarea objetivo, se inicializan los pesos a partir del modelo entrenado sobre MS-CELEB, exceptuando los de la última capa totalmente conectada ya que estos estarán íntimamente relacionados con la tarea objetivo.

El entrenamiento de la red se logra a través de Label Distribution Learning [38], ya que podría ayudar a alivianar el problema de las inconsistencias de las anotaciones y extraer mayor rendimiento. Para esto se incorpora un generador de emociones modelado con una ResNet50, el cual se mantiene congelado durante el entrenamiento de EfficientFace realizando únicamente inferencia sobre cada input para obtener la distribución de emoción asociada (soft labels). Finalmente la función de pérdida computa la similitud entre las estimaciones de las distribuciones de emociones estimadas por EfficientFace y aquellas dadas por el generador (consideradas como ground-truth).

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 propone el uso de Label Distribution Learning (LDL) [38,9]. 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. El problema principal es que la mayoría de los datasets del dominio cuentan con etiquetas categóricas en lugar de distribuciones de emociones. Por lo tanto, se propone construir un generador de distribuciones de emociones o LDG (por su sigla en inglés, **L**abel**D**istribution **G**enerator) tal como se hace en el trabajo de Zhao et al. [37] para implementar esta estrategia.

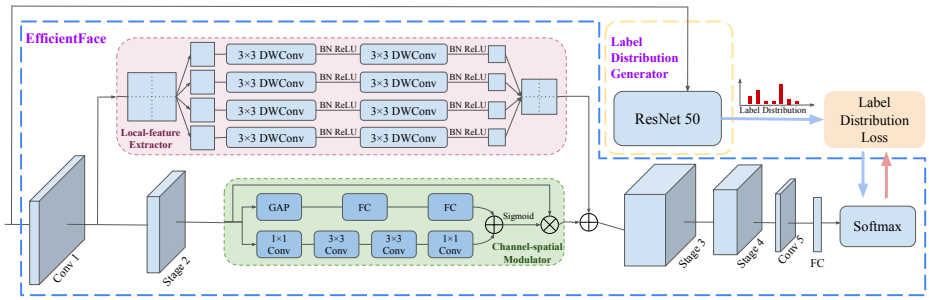

Fig. 5. Esquema completo del framework EfficientFace [37].

Concretamente, el modelo utilizado para el LDG es una ResNet-50 [12] que entrenamos para cada una de los datasets que incluye nuestro benchmark. Al igual que en EfficientFace, 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 para consultar el ground truth de cada instancia durante el entrenamiento con Label Distribution Learning. De esta forma, el LDG permanece haciendo únicamente inferencia asociándole una distribución de emoción estimada a cada instancia como se puede ver esquematizado en la Figura 5.

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. [3], para intentar hacer aún más robusto el aprendizaje. Notar que esto no supone ningún aumento en la complejidad de la red ni tampoco afecta la velocidad de inferencia. Sin embargo, encarece el tiempo de entrenamiento y el trabajo de pre-procesamiento asociado a cada dataset. 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 generados por dos herramientas del estado del arte: OpenFace [1] y MCLTR [27] que utilizan técnicas de Machine Learning tradicionales y end-to-end learning respectivamente.

Para determinar el desempeño de clasificación de los modelos, utilizamos principalmente accuracy rate (tasa de aciertos) que es una medida habitual para estos problemas y como benchmark tomamos los datasets in-the-wild: CAER-S, RAF-DB, AffectNet, FER+. Finalmente, al considerar el costo computacional se tiene en cuenta que la elección de los FLOPs como medida puede ser engañosa. Esto tiene que ver, principalmente, con el hecho de que 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 [22]. Por lo tanto, arquitecturas con FLOPs comparables pueden

tener velocidades distintas. En definitiva, elegimos como medida del costo computacional los lotes por segundo (para el caso GPU) o bien imágenes por segundo en CPU.

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

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.

Para el entrenamiento, capturamos mediante Cross-Entropy la similaridad entre la distribución estimada por el LDG y la del ground-truth. Es decir:

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

donde $N = |S|$ y $\widehat{d_{x_i}} = (\widehat{d_{x_i}^{y_1}}, \widehat{d_{x_i}^{y_2}}, \ldots, \widehat{d_{x_i}^{y_c}})$ 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 definitiva, usamos Cross-Entropy con hard labels.

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. Es decir, siguiendo la misma función de pérdida que en la ecuación 1 y teniendo un dataset construido con etiquetas categóricas.

Habiendo entrenado el LDG de manera independiente, se lo mantiene congelado para auxiliar el entrenamiento de EfficientFace. Se puede pensar que el LDG funciona como '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. 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:

$$d_{x_i}^{y_i\prime} = \frac{\exp(v_{x_i}^{y_i})}{\sum_{j=0}^{c-1} \exp(v_{x_i}^{y_j})} \tag{2}$$

Finalmente, habiendo conseguido S' con soft labels, utilizamos Cross-Entropy como se definió en la ecuación 1 para este caso.

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.

Entrenamiento con LDL y AUs. Siguiendo la idea de Chen et al. [3], se propone utilizar el espacio de etiquetas de una tarea auxiliar para usar información sobre su topología. Para esto consideramos un conjunto de entrenamiento $S = \{(x_1, l'_1, w_1), \ldots, (x_n, l'_n, w_n)\}$ donde $w_i \in W$ siendo $W \subseteq \mathbb{R}^k$ el conjunto de etiquetas asociadas a Action Unit Recognition, la tarea auxiliar, y L' el conjunto de soft labels generados por el LDG. En este caso particular, cada vector del conjunto W representa la intensidad de activación de cada Action Unit. En general los métodos de anotación de Action Units calculan un subconjunto de ellas, en nuestro caso obtenemos 12 y 17 dependiendo del método elegido (MLCR y OpenFace respectivamente).

Dada una instancia del espacio de imágenes, el objetivo es guiar el aprendizaje de la red de manera que considere la vecindad de la instancia pero sobre el espacio de la tarea auxiliar. Para esto 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 ponderadas 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.

Inicialmente, para introducir esta idea consideramos hacerlo de manera indirecta, modificando el entrenamiento del LDG con el objetivo de lograr un "mejor generador" y en última instancia poder destilar ese conocimiento al entrenar EfficientFace. Sin embargo, si pensamos que al optimizar la función de pérdida del LDG estamos minimizando el error entre la estimación del generador, soft

label, contra el ground truth, hard label, la distribución aprendida podría terminar pareciéndose demasiado a la del vector lógico determinado por la emoción categórica del ground-truth. Esta situación sería indeseable, ya que no estaría capturando la ambigüedad propia de las expresiones siendo este el propósito de Label Distribution Learning en primer lugar.

En lugar de ello, vamos a proponer una modificación a la función de pérdida de EfficientFace LDL. La separamos en dos términos: uno que, como antes, computa la pérdida contra la distribución de emociones determinada a partir del ground truth de soft labels (el conjunto L') y otra que capture la pérdida respecto a la distribución de emociones estimada de cada uno de sus vecinos.

$$\mathcal{L}_{ldg-aus}(\theta) = \mathcal{L}(\theta) + \lambda\Omega(\theta). \tag{3}$$

A diferencia de [4], 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 (como en la ecuación 2) y $D_{KL}(P, Q)$ la Divergencia de Kullback-Leibler entre las distribuciones P y Q, definimos:

$$\Omega(f(x|\theta)) = \sum_{ij} a_{ij} D_{KL}(f(x_j)||f(x_i)) \tag{4}$$

Para computar el coeficiente de similitud local a_{ij} siguiendo la misma idea de [4], utilizamos una noción de distancia tal que:

$$a_{ij} = \begin{cases} \exp\left(-\frac{d(w_i,w_j)}{\sigma^2}\right) & \text{si } w_j \in N(i) \\ 0 & \text{en otro caso} \end{cases} \tag{5}$$

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 esta forma, queda definida la estrategia para incorporar la información de la tarea auxiliar cuyo esquema general puede observarse en Figura 6.

Volviendo sobre la ecuación 5, será necesario definir el vecindario de una imagen en el espacio de labels de Action Units utilizando alguna noción de distancia. Para esto, construimos un grafo approximate-KNN utilizando la herramienta NGT[3] como en Chen et al. [4]. De esta forma, se computa offline un índice general para cada dataset que dada la etiqueta de AU 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

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

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).

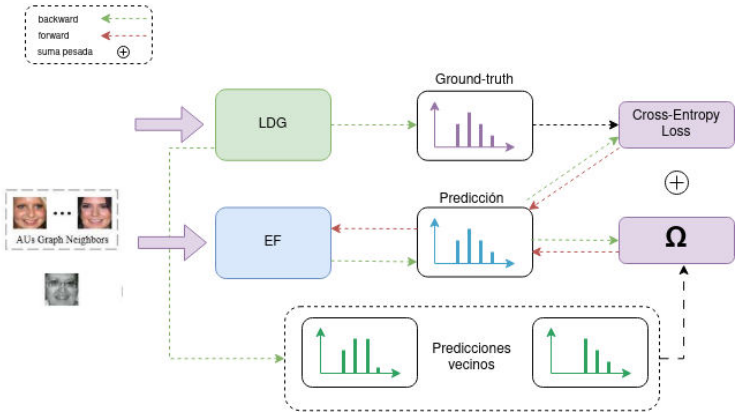

Fig. 6. Esquema para LDL AUS.

3.3 Anotación de Action Units

Como se mencionó en la Sec. 1.1, las Action Units (AUs) codifican movimientos de grupos de músculos faciales según el Facial Action Coding System (FACS) [8]. 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 7. 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 y otras para acciones misceláneas. 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.

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 in-the-wild. 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[25].

Para este trabajo será necesario contar con las anotaciones de algún subconjunto de AUs para así determinar el espacio auxiliar de etiquetas. En particular se decidió experimentar con dos herramientas del estado del arte: al igual que en Chen et al. [3] se usa Openface [1], utilizando técnicas de Machine Learn-

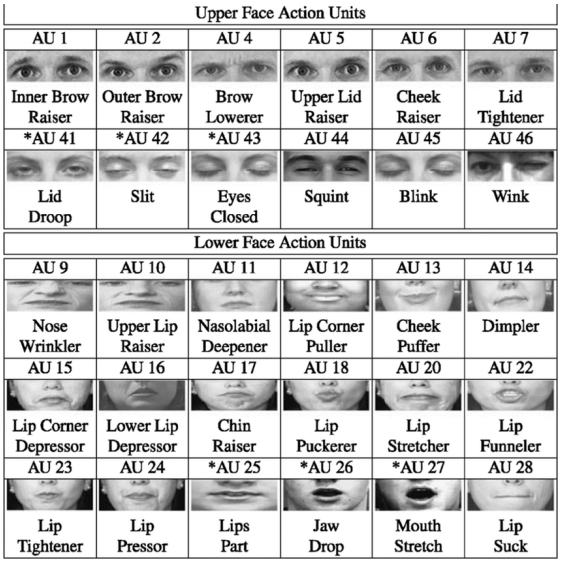

Fig. 7. Codificación de acciones faciales en Action Units según FACS.

ing convencionales, y Mult Label Co-Regularization (MLCR) [27] como método alternativo basado en Deep learning.

4 Experimentos y Resultados

4.1 Datasets

Para evaluar el desempeño de EfficientFace se utilizan datasets de gran escala comúnmente adoptados en FER: RAF-DB [21], FER+ [2], CAER-S [19] y AffectNet [26].

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. Con esto se obtienen 12,271 imágenes para entrenamiento y 3,068 para test.

CAER: Surge 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. Utilizamos 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. Usamos la herramienta MTCNN [35] para conseguir las bounding box, centrar las caras y luego usar los landmarks de los ojos para alinear el rostro.

FER+: Es una versión extendida de FER2013 y consiste de 28,709 imágenes de entrenamiento, 3,589 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 crowdsourcing.

AffectNet: Contiene aproximadamente 450,000 imágenes que fueron manualmente anotadas con un total de 11 categorías de expresiones. 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 la arquitectura propuestas se proponen una serie de experimentos afines. Inicialmente en la Sec. 4.3 se corroboran los resultados obtenidos en trabajos previos [37] para Single Label Learning y se extienden los resultados a nuestro benchmark.

Luego en la Sec. 4.4 se hace un estudio de ablación similar al del trabajo original para evaluar la eficiencia de la técnica de Label Distribution Learning. En la Sec. 4.5 estudiamos la variante de la técnica de LDL empleando Action Unit Recognition como tarea auxiliar. A través de otro estudio de ablación analizaremos la eficiencia de esta técnica sobre los datasets RAF-DB y FER+ únicamente.

En último lugar, en la Sec. 4.6 evaluamos la complejidad de las redes a través de una medida más directa que los FLOPs, utilizando la velocidad (imágenes por segundo en CPU y batches por segundo en GPU).

Para cada modelo estudiado se hizo una exhaustiva búsqueda sobre el espacio de hiperparámetros. En la mayoría de los experimentos, teniendo en cuenta los tiempos que insume el entrenamiento de estos modelos, se realizaron varios 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.

Específicamente, sobre CAER-S se entrenó durante 350 epochs con un batch-size de tamaño 128, 1-cycle 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 "ReduceLrOnplateau", 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 [35] para detectar y centrar los rostros. 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.

4.3 Replicando resultados baseline

Como experimento preliminar de este trabajo buscamos corroborar los resultados obtenidos para la arquitectura de EfficientFace en su versión Single Label Learning (de ahora en más EF SLL) sobre los datasets propuestos en el trabajo de Zhao et al. [37].

Para ajustar los hiperparámetros en el entrenamiento se consideraron los valores sugeridos en el trabajos original y se entrenó usando la misma cantidad de epochs como máximo. En ocasiones los resultados obtenidos de esta forma se acercaban a lo reportado, pero encontramos que con el optimizador RADAM podía extraerse un poco más de perfomance. Lo mismo ocurrió al introducir schedulers alternativos como ReduceLrOnPlateau y 1-cycle [32] con los que logramos aún mejores resultados. Los resultados de esta experimentación preliminar pueden observarse en la fila EF SLL en la Tabla 2.

Encontramos mejorares resultados que el trabajo previo en el dataset RAF-DB, consiguiendo 1.92% por sobre éste (principalmente por el uso del optimizador RADAM), sin embargo el obtenido sobre CAER-S es levemente inferior estando tan solo 0.21% por debajo. No podemos comparar contra los resultados específicos de esta parte sobre AffectNet ya que no se reporta en el trabajo original. 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 y nuestro máximo valor hallado fue de 57.08% sugiriendo algún tipo de sesgo en nuestro entrenamiento más que en el método. Aunque el trabajo original no reporta sobre FER+, está a la vista que es es un buen resultado al compararlo con el resto de las arquitecturas de la Tabla4.3. Una vez más, RADAM admitió mejores resultados que el resto de los optimizadores considerados para este caso.

A su vez, como se esperaba, esta arquitectura diseñada específicamente para lidiar con el problema de FER in-the-wild superan cómodamente a otras redes lightweight del estado del arte como MobileNet-V2. Otro punto destacable es que EfficientFace se desempeña en general de manera equiparable a modelos más complejos del estado del arte. 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.

4.4 Efectividad de Label Distribution Learning

Con el objetivo de evaluar la eficiencia de la estrategia de Label Distribution Learning, se implementó un estudio de ablación. Para ello, se obtuvo en primer lugar un generador de distribuciones (LDG), básicamente una arquitectura ResNet-50, que 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 1-cycle con learning rate máximo de 0.001 durante 300 epochs. Se mantuvieron las mismas técnicas de data-augmentation y preprocesamiento que en EfficientFace.

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 (LDG)	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

Table 1. Accuracy máximo de lightweights del estado del arte, excepto SCAN, según lo reportado en [10] [37] y nuestros valores hallados para EfficientFace SLL al igual que el LDG. Los resultados corresponden los modelos pre-entrenados en MS-CELEB.

A partir de los generadores ajustados a cada dataset, se entrenó a EfficientFace adaptada para tener en cuenta las distribuciones de emociones generadas por el LDG adecuado y se obtuvieron los resultados de la Tabla 2 (ver EF LDL).

Método	Train	RAF-DB			CAER-S			FER+			AffectNet		
		min	med	max	min	med	max	min	med	max	min	med	max
LDG	SLL	88,30	**88,41**	88,69	85,77	**85,97**	86,17	88,72	88,97	89,29	57,94	**58,57**	59,2
EF	SLL	86,50	86,92	87,58	84,05	84,17	84,30	87,96	88,09	88,44	55,60	56,7	57,08
	LDL	87,38	88,03	88,52	84,01	84,20	84,4	88,62	**89,10**	89,32	57,58	57,87	58,17

Table 2. Resultados para el experimento de ablación de la la Sec. 4.4. Se reportan valores de accuracy 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 de cada LDG utilizado para entrenar EfficientFace.

Como primer punto a remarcar, los resultados del estudio son alentadores. Bajo nuestro benchmark, EfficientFace mejora su rendimiento con la estrategia de Label Distribution Learning en todos los datasets, consiguiendo subas en el accuracy rate mediano reportado de aproximadamente 1% en todos los datasets (exceptuando CAER-S donde la suba es menos significativa). 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%).

Si comparamos los resultados contra el LDG encontraremos otro punto destacable: A pesar de tratarse de una red mucho menos compleja que la ResNet-50, EfficientFace logra resultados comparables. Una pregunta que surge es hasta que

punto debería entrenarse el LDG para que su incorporación al entrenamiento de EfficientFace sea conveniente (comparada respecto de su versión SLL). Si pensamos al LDG como un clasificador y lo evaluamos según su accuracy, comprobamos empíricamente sobre el dataset RAF-DB que hasta un 1% de accuracy por debajo de su capacidad de clasificación máxima (88.69%) su incorporación sigue siendo beneficiosa. Más aún, la ganancia que le aporta a EfficientFace contar con un LDG entrenado entre este punto y su capacidad máxima es marginal.

Concluyendo esta parte, EfficientFace se entrenó sobre RAF-DB y FER+ durante 100 épocas 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 épocas. Se usaron las mismas técnicas de data-augmentation y preprocesamiento que las mencionadas en la Sec 4.3.

Observando las matrices de confusión de las figuras Figura 8 queda claro que las emociones positivas (neutral, happy, surprise) son de las más fáciles de reconocer, consiguiendo accuracy rate superiores al 90% y 81% en los datasets RAF-DB y CAERS respectivamente. En particular, happiness es la emoción que prevalece como la mejor clasificada en RAF-DB, FER+ y AffectNet, a la vez que es la que aparece con más frecuencia en estos datasets. Las emociones negativas suelen ser las más difíciles de reconocer: por ejemplo en RAF-DB 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 neutral en RAFDB. Puede pensarse 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. Notamos que sobre CAER-S el modelo clasifica razonablemente bien en general todas las emociones, yendo a contracorriente de lo que se intuye a partir del caso de RAF-DB. Esto puede deberse, por un lado, a que el dataset tiene más del doble de instancias que RAFDB y que, además, las clases están balanceadas.

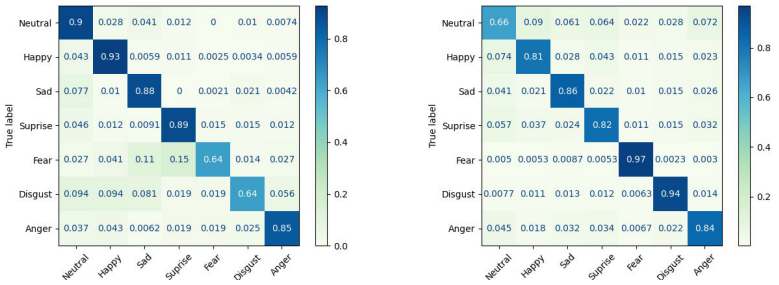

Fig. 8. Matrices de confusión para RAF-DB (izq) y CAERS (der) para EfficientFace.

4.5 Efectividad del Reconocimiento de AUs como Tarea Auxiliar

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 4 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 [24] 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 creo que más o obtenida no siempre refleja la topología original. Sin embargo las vecindades en dimensión alta deberían preservarse.

La forma de las representaciones obtenidas depende fuertemente de la elección del hiperparametro perplexity que puede interpretarse como la medida de la cantidad estimada de vecinos [24]. Por lo tanto se abordó el análisis luego de observar resultados para distintas elecciones de lo anterior. Al considerar algunos datasets in-the-wild relevantes para este trabajo, hay que tener que en cuenta, además, que el balance de clases tampoco es uniforme 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 los movimientos musculares que exhiben estas expresiones. Así, muchas instancias son descartadas al no contar con un grado de confianza en la anotación suficiente.

Analizamos la calidad de las anotaciones obtenidas por las herramientas de MCLTR y OpenFace, nos enfocamos en los resultados de las representaciones t-SNE obtenidas para el dataset RAF-DB como puede verse en la Figura 9.

A simple vista, queda claro que en ambos casos resulta difícil definir una buena clusterización. A pesar de eso, 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.

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 evaluado 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.

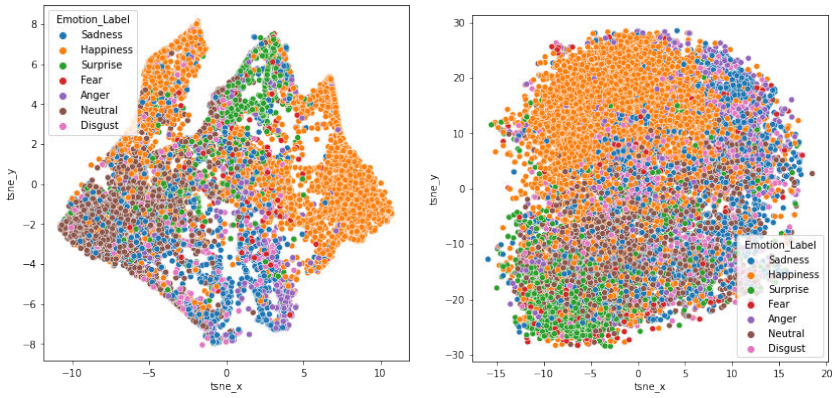

Fig. 9. Visualizaciones de los descriptores t-SNE para el dataset RAF-DB a partir de las anotaciones automáticas de MCLTR (izq) y las anotaciones de OpenFace (der).

Para continuar evaluando la calidad de las anotaciones utilizamos como guía la codificación de EMFACS [28] que define emociones básicas prototípicas a partir de la presencia de Action Units en las expresiones faciales. Nuevamente, centrándonos en el dataset RAF-DB analizamos, para cada emoción, la intensidad promedio, entre todas las instancias, de cada Action Unit normalizadas al rango $[0, 1]$. Entre ambos métodos, MTCLR muestra una característica favorable estimando intensidades de activaciones considerablemente más altas para las AU principales de cada emoción. De hecho en OpenFace las activaciones no suelen superar el umbral de 0.5. A su vez, en la categoría de emociones negativas encontramos mayor cantidad de activaciones que pueden ser inconsistentes: por ejemplo en MCLR, 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 de AU 25 en anger y fear.

Al considerar la topología del espacio de etiquetas de Action Units para hacer más robusto el entrenamiento con Label Distribution Learning se propuso modificar las función de pérdida de EfficientFace agregando un término que capture las relaciones de una instancia con sus vecinos (ver ecuación 4). Los resultados de nuestra experimentación sobre el dataset RAFDB y FER+ mostraron que este método no consigue mejores resultados, cayendo por debajo del baseline reportado en la Tabla 2 (ver EF LDL) por aproximadamente 1% y 0.6% en RAF-DB y FER+ respectivamente.

Teniendo en cuenta lo discutido inicialmente en esta sección, 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 con los margenes de mejora esperados. La situación también amerita replantear la definición de la función de pérdida como otra alternativa a explorar.

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

Table 3. Para todas las corridas el tamaño del batch quedó fijo en 16 para el caso GPU y en 1 para las de CPU. Con memoria se consigna el consumo en la etapa de inferencia según torch profiler.

4.6 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 medida 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 1080Ti 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 imágenes/s.

Los resultados obtenidos en la Tabla 3 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.

5 Conclusiones

En este trabajo se abordó el problema de Facial Expression Recognition (FER) con arquitecturas de redes neuronales convolucionales lightweight. 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 el rendimiento final.

Se estudió la arquitectura lightweight del estado del arte EfficientFace [37] que, considerando el problema de la inconsistencia de anotaciones, propone el

uso de Label Distribution Learning (LDL) para entrenar modelos a partir de etiquetas que representa 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 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, mostrando que EfficientFace puede alcanzar un accuracy rate equiparable a arquitecturas del estado del arte de mayor complejidad, como las ResNet-50, incluso en condiciones de entorno complejas.

El estudio de ablación del experimento 4.4 para evaluar la eficiencia de LDL en este contexto acompañó los resultados de Zhao et al [37] 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 como mucho subas de aproximadamente 1% en la mayoría de los datasets.

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. Sin embargo 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 buena para auxiliar al aprendizaje.

Por último, obtuvimos una comparación más directa de la complejidad de la arquitectura estudiada a partir de las mediciones de velocidad y notamos 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.

Agradecimientos. Este trabajo cuenta con financiamiento de la Agencia I+D+i mediante el proyecto PICT-2017-4316 del FONCYT y de la Universidad de Buenos Aires mediante el proyecto UBACyT 2020 Mod II Cod. 20020190200317BA.

References

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)

3. 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)
4. 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)
5. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Conf. on Comp. Vision and Patt. Recog. (CVPR). pp. 1800–1807. USA (2017)
6. 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)
7. Ekman, P.: Universal and cultural differences in facial expression of emotion. In: Nebraska Symposium on Motivation. vol. 19, pp. 207–283 (1971)
8. Ekman, P., Friesen, W.V.: Facial action coding system: a technique for the measurement of facial movement (1978)
9. Gao, B.B., Xing, C., Xie, C.W., Wu, J., Geng, X.: Deep label distribution learning with label ambiguity. IEEE Trans. on Image Processing **26**(6), 2825–2838 (2017)
10. Gera, D., Balasubramanian, S., Jami, A.: Cern: Compact facial expression recognition net. Pattern Recognition Letters **155**, 9–18 (2022)
11. 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)
12. 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)
13. 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. ArXiv **abs/1704.04861** (2017)
14. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7132–7141 (2018)
15. Huang, G., Liu, Z., Maaten, L.V.D., Weinberger, K.Q.: Densely connected convolutional networks. In: IEEE Conf. on Comp. Vision and Patt. Recognition (CVPR). pp. 2261–2269 (2017)
16. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR **abs/1502.03167** (2015)
17. 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)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (may 2017)
19. Lee, J., Kim, S., Kim, S., Park, J., Sohn, K.: Context-aware emotion recognition networks. In: IEEE/CVF Int. Conf. on Comp. Vision. pp. 10142–10151 (2019)
20. 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)
21. 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)
22. 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)
23. Maat, L., Pantic, M.: Gaze-x: adaptive affective multimodal interface for single-user office scenarios. In: International Conference on Multimodal Interaction (2006)

24. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(86), 2579–2605 (2008)
25. 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)
26. 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)
27. Niu, X., Han, H., Shan, S., Chen, X.: Multi-label co-regularization for semi-supervised facial action unit recognition. CoRR **abs/1910.11012** (2019)
28. P. Ekman, W.V.F., Hager, J.C.: Facial action coding system: The manual on cd rom. a human face (2002)
29. Park, J., Woo, S., Lee, J.Y., Kweon, I.S.: Bam: Bottleneck attention module. ArXiv **abs/1807.06514** (2018)
30. 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)
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd Int. Conf. on Learning Representations (ICLR 2015). pp. 1–14 (2015)
32. Smith, L.N., Topin, N.: Super-convergence: very fast training of neural networks using large learning rates. In: Proc. of SPIE. vol. 11006, pp. 369–386 (2019)
33. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting **15**(1), 1929–1958 (2014)
34. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conf. on Comp. Vision and Patt. Recognition (CVPR). pp. 1–9 (2015)
35. 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)
36. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: IEEE/CVF Conf. on Comp. Vision and Patt. Recognition (CVPR). pp. 6848–6856. IEEE Computer Society, Los Alamitos, CA, USA (jun 2018)
37. 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)
38. 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)
39. Zhou, Y., Xue, H., Geng, X.: Emotion distribution recognition from facial expressions. p. 1247–1250. Assoc. for Computing Machinery, NY, USA (2015)
40. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning. Proceedings of the IEEE **109**(1), 43–76 (2021)