Clasificación de actividades humanas y estimación automática de la pose en imágenes mediante técnicas de aprendizaje profundo

Alejandro Torres Rodríguez
torresalex@correo.ugr.es

Javier Moreno Morón
javiermm@correo.ugr.es

Ariel Terrón Rosas
arieltrosas@correo.ugr.es

Iván Salinas López
ivansalinas@correo.ugr.es

Resumen

Este trabajo explora el uso de técnicas de aprendizaje profundo para el análisis de actividades humanas y la estimación de la pose basada en 16 articulaciones del cuerpo humano. Para el análisis de las actividades humanas se compara el rendimiento en clasificación de distintas arquitecturas populares en visión por computador como VGG, ResNet o GoogLeNet. También se incluye una comparativa con técnicas no basadas en redes neuronales, como es el caso de Bag of Visual Words. Para la estimación de la pose se ha optado por la regresión de 16 pares de coordenadas, uno por articulación, así como un acercamiento basado en mapas de calor. Estas dos vertientes son las más usadas en este ámbito y representan dos soluciones a un mismo problema, por lo que se espera observar las ventajas y desventajas de cada método. Las arquitecturas propuestas consiguen una buena tasa de acierto a la hora de clasificar actividades humanas, sin embargo los métodos propuestos para la estimación de la pose no han cumplido con las expectativas y precisan de un mayor refinamiento.

1. Introducción

Las redes neuronales profundas están detrás del gran avance en multitud de campos. En concreto, en el campo de la visión por computador, las CNN (Convolutional Neural Networks) han supuesto una revolución. Problemas clásicos, como la extracción de descriptores para la detección de objetos, han sido reemplazados por las técnicas modernas de aprendizaje profundo, delegando esta responsabilidad en el aprendizaje de la red.

Un aspecto clave en el éxito de las CNN es la reducción en la cantidad de parámetros que estos sistemas deben aprender en contraposición con las redes neuronales clásicas. Por consiguiente, las CNN pueden alcanzar una profundidad mucho mayor que sus predecesores. Sin embargo, esta mayor profundidad plantea a su vez nuevos retos al aumentar el abanico de capas y arquitecturas disponibles. Determinar que arquitecturas y configuraciones de una red resultan más exitosas para una tarea es un problema difícil. Una prueba de esto es la gran cantidad de arquitecturas que han surgido en los últimos años.

En este trabajo se abordan dos problemas sobre imágenes de actividades humanas: la interpretación semántica de dicha actividad y la estimación de la pose de una persona a partir de una imagen. Estos problemas corresponden con clasificación de las actividades y regresión de coordenadas, respectivamente.

Cada uno de estos problemas tiene sus propias aplicaciones. La clasificación de imágenes en función de la actividad desarrollada puede ser útil para la comprensión del comportamiento de personas en aplicaciones como conducción autónoma o realidad aumentada. Mientras que determinar la posición y pose de una persona en una imagen también es útil en estas áreas, ofreciendo información complementaria, por ejemplo a la hora de determinar la localización de un peatón en un sistema de conducción autónoma.

El objetivo de este trabajo es realizar una comparativa de distintas arquitecturas y métodos aplicados a estos problemas para obtener el mejor rendimiento posible. En el caso de la clasificación de imágenes compararemos las arquitecturas VGG, ResNet y GoogLeNet así como Bag of Visual Words. En cuanto a la estimación de la pose estudiaremos la regresión directa de coordenadas y el uso de heatmaps.

2. Fundamentos teóricos

El aprendizaje automático constituye una disciplina en el ámbito de la inteligencia artificial (IA) y las ciencias de la computación que se enfoca en la utilización de datos y algoritmos para emular el proceso de aprendizaje humano. Su objetivo es identificar patrones en los datos con el fin de realizar predicciones. Se distinguen tres tipos de aprendizaje: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. Siendo el primero el más

utilizado y el que encaja en el marco de nuestro trabajo.

Uno de los modelos de aprendizaje más populares y que ha demostrado una mayor efectividad son las redes neuronales. Estas consisten en realizar una serie de operaciones de suma y multiplicación encadenadas que, mediante la introducción de no linearidad (función de activación) consiguen modelar problemas complejos. Cuando aumenta el número de capas de la red, esta se considera una red neuronal profunda.

Las redes neuronales convolucionales o CNN en inglés, son un tipo de red neuronal profunda que trabaja con patrones de cuadrícula, como pueden ser imágenes. Este tipo de redes extraen las características de las imágenes mediante el aprendizaje de filtros en las sucesivas capas. Estos filtros suponen una reducción en el número de pesos que la red tiene que aprender.

Los componentes principales en el diseño de CNNs son:

- Capa de convolución. Este tipo de capas extraen características de la imagen, normalmente reduciendo las dimensiones de esta y aumentando el número de canales.
- Función de activación. Las funciones de activación introducen no linearidad en el modelo. Una de las más utilizadas es la función ReLU.
- Capa de pooling. Las capas de pooling reducen la dimensionalidad del problema conservando la información más relevante. Los más utilizados son *MaxPooling* y *AveragePooling*.
- Capa de normalización. Las capas de normalización ayudan a la generalización evitando el sobreajuste.
- Capa densamente conectada. Normalmente situadas en la parte final de la red, utilizan las características extraidas en los bloques convolucionales para producir la salida, aunque esto puede variar según la arquitectura.

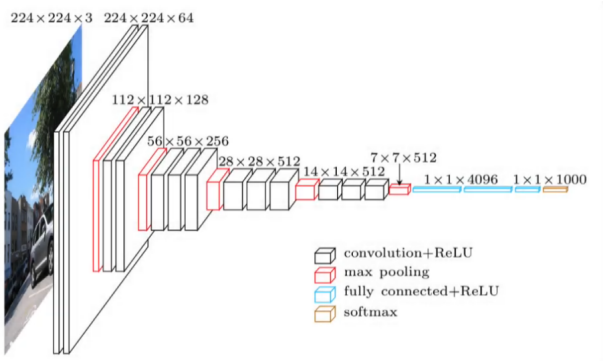

Figure 1. Estructura de una red neuronal convolucional [1]

Una de las técnicas usadas en aprendizaje profundo es la transferencia de aprendizaje (*transfer learning* en inglés). Esta técnica consiste en ajustar un modelo ya entrenado en un problema similar a un problema concreto. Por ejemplo, los modelos entrenados en ImageNet pueden ser utilizados en otros problemas de visión. Esto supone una mayor velocidad de entrenamiento manteniendo un buen rendimiento, ya que entrenar completamente una red compleja puede ser una labor costosa.

La extracción de características no siempre es automática, existen técnicas de extracción manual como SIFT, ORB, Haralick, étc. Se pueden utilizar estas técnicas para conseguir las características de una imagen y utilizarlas para clasificación.

3. Trabajos relacionados

Tal y como se describe en [4], desde los años 2000 se han llevado a cabo trabajos relacionados con la estimación de la pose y el reconocimiento de actividades mediante técnicas de visión por computador y procesamiento de imágenes.

Uno de los trabajos más relevantes en la estimación de la pose a partir de imágenes es *Deep Pose* [8]. En este trabajo, la estimación de la pose se realiza a través de una serie de predictores en cascada que refinan la predicción realizada por el anterior en la cadena. Otro acercamiento popular al problema es la utilización de mapas de calor. El principal problema de este acercamiento es la construcción de dichos mapas. Trabajos como el realizado por *Numerical Coordinate Regression with Convolutional Neural Networks* [5] eliminan la necesidad de procesar estos mapas de calor haciendo que la red neuronal aprenda los mapas por si misma. Para ello, introducen una nueva capa al final del modelo que es capaz de extraer las coordenadas para la pose a partir de un mapa de calor siendo diferenciable y por tanto permitiendo propagar el error por la red. De esta forma, la salida producida por la red son directamente las coordenadas deseadas.

Uno de los trabajos más recientes es [3], donde se introduce una nueva red híbrida que combina una arquitectura de tipo reloj de arena junto con arquitecturas de U-Net, lo que permite minimizar el número de conexiones identidad dentro de la red e incrementa el rendimiento para la misma cantidad de parámetros. Esta arquitectura obtiene los mejores resultados hasta la fecha respecto a ciertos conjuntos de datos como pueden ser MPII Human Pose Dataset y LSP Dataset.

4. Métodos

4.1. Clasificación de actividades

Con el objetivo de alcanzar una estimación automática de las actividades que se están realizando en una imagen, hemos hecho uso de distintas arquitecturas convolucionales

profundas. Estas redes neuronales convolucionales están preentrenadas con el conjunto de imágenes de ImageNet, ampliamente utilizado para entrenar CNNs, ya que contiene millones de imágenes distintas.

VGG16 es una de las arquitecturas más simples y utilizadas para aprovechar la transferencia de aprendizaje. Esta arquitectura consta de 16 capas con pesos y solo realiza convoluciones de 3x3 ya que implícitamente se realiza la agregación de escalas mayores al pasar de capa. Utilizaremos VGG16 en nuestro problema.

Uno de los mayores problemas de las arquitecturas profundas es el desvanecimiento del gradiente. ResNet introduce los bloques residuales para evitar este problema. Para los experimentos utilizaremos ResNet18 ya que es la más simple de este tipo de arquitecturas.

Otro enfoque son las arquitecturas, como GoogLeNet, que utilizan módulos Inception. Estos módulos permiten una mayor profundidad y por tanto un mayor aprendizaje de características pero de manera más eficiente, utilizando menor cantidad de parámetros.

Por otro lado, entre las técnicas no basadas en redes neuronales tenemos *Bag of Visual Words*. Esta técnica consiste en la extracción manual de características en la imagen con métodos como SIFT u ORB. Luego, se crea un diccionario visual agrupando las características en palabras visuales mediante un algoritmo de agrupamiento como puede ser k-medias. Una vez creado el diccionario visual se calculan los histogramas de las palabras visuales de cada imagen y finalmente, mediante técnicas de aprendizaje automático se trata de entrenar un modelo que use los histogramas como entradas. La principal ventaja de este método es que es más ligero computacionalmente que las redes neuronales. Sin embargo, la efectividad de este método depende de la calidad de los descriptores utilizados para la composición del vocabulario visual.

4.2. Estimación de la pose

Nuestro problema puede verse como la regresión de un vector de 32 coordenadas, donde cada articulación se representa como un par de coordenadas (x,y) en la imagen. Una de las primeras consideraciones, tanto a la hora de entrenar como al procesar los datos, es considerar la presencia de varias personas en una sola imagen. Como se explica en el artículo *Deep Learning-Based Human Pose Estimation: A Survey* [9], existen dos alternativas posibles. La primera es entrenar un detector de personas independiente y aplicar el modelo entrenado para una sola persona a los resultados del primer modelo. La segunda sería entrenar directamente un modelo que estime las distintas poses de todas las personas en la imagen, efectivamente entrenando un detector de personas y estimador de pose al mismo tiempo.

El primer método escogido para abordar el problema es la estimación de este vector utilizado como base el modelo preentrenado en ImageNet de ResNet18 y una cabeza con una salida de 32 coordenadas. Este método presenta algunos problemas, ya que, por la naturaleza del dataset es posible que en algunas imágenes algunas de las articulaciones no estén presentes, por ejemplo si se trata de una imagen del tronco superior de una persona. En este caso, las opciones de las que disponemos son eliminar del conjunto de datos dichas imágenes, restringiéndonos a cuerpos completos o asignar un par de coordenadas especial por ejemplo $(-1.0, 1.0)$. Optaremos por descartar dichas imágenes y evaluar el rendimiento del modelo sobre esta versión simplificada del problema.

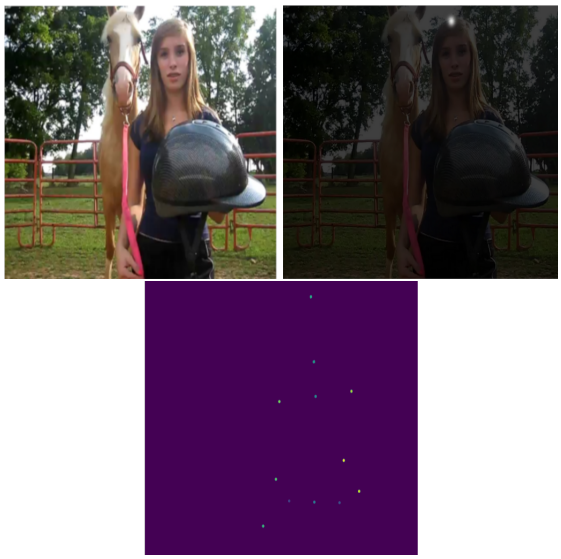

Figure 2. Mapa de calor con mapa de segmentación asociado a todas las articulaciones

La segunda aproximación sería utilizar mapas de calor para cada articulación y la extracción de las coordenadas a partir de mapas de calor estimados. Esta aproximación permite usar arquitecturas de redes como U-Net, permitiendo trasladar la información espacial de las primeras capas de la red a las últimas capas donde se realiza la predicción. El mayor incoveniente de este método es la necesidad de entrenar 16 modelos distintos, uno para cada parte del cuerpo. En este punto, proponemos una solución en la cual se plantea el problema de la estimación de la pose como uno de segmentación, en el cual se consideran 17 clases (16 articulaciones y el fondo) para los píxeles de la imagen. Superponiendo 16 mapas de calor y asignando una clase a cada uno de los mapas usados, creamos un objetivo para el problema de segmentación que usaremos para entrenar nuestro modelo U-Net. La principal ventaja que esperamos obtener con este planteamiento del problema es que en ausencia de una de las articulaciones el modelo siga funcionando correctamente simplemente ignorando la clase correspondiente a dicha articulación.

Aumentación	ρ	Valor
Rotación (º)	0.75	[0, 10.0]
Brillo	0.75	[0, 0.2]
Deformación	0.75	[0, 0.2]
Ampliación	0.75	[1.0, 1.1]
Volteo	0.5	

Figure 3. Margen de distorsión en técnicas de aumento de datos y probabilidades para su aplicación aleatoria en imágenes de entrenamiento

5. Experimentos

El conjunto de datos que se ha utilizado para realizar el trabajo es el *MPI Human Pose Dataset* [2] que es considerado como el punto de referencia para la evaluación de la pose humana en imágenes. El conjunto de datos está formado por alrededor de 25.000 imágenes que contienen aproximadamente 40.000 personas, cubre 410 actividades humanas y todas las imágenes se obtuvieron de vídeos de YouTube.

Tras obtener el conjunto de datos realizamos una exploración de los mismos y descubrimos que las imágenes destinadas al conjunto de test no estaban etiquetadas ya que el propósito original del dataset era realizar una competición en la que los organizadores validaban los modelos de los concursantes y por ende no daban las etiquetas del conjunto de test. Para solucionar este problema lo que hemos hecho es no utilizar estas imágenes y crear nosotros mismos el conjunto de test a partir de las imágenes que están correctamente etiquetadas. Otro de los problemas que encontramos con el dataset es que algunas imágenes que estaban destinadas a entrenamiento, por lo que deberían de estar etiquetadas correctamente, no lo estaban y por tanto tuvimos que eliminarlas. En el archivo matlab con las anotaciones de cada imágen había anotaciones de 3 imágenes inexistentes por lo que estas también fueron eliminadas del dataset. Además, hay clases en el conjunto de datos que solo tenían una imágen por lo que estas también fueron eliminadas ya que no tiene interés aprender a clasificar una clase si solo tenemos una muestra, además que al dividir el conjunto en entrenamiento y test existe la posibilidad de que la imágen se encuentre en el conjunto de test haciendo imposible que el modelo aprenda sobre esa clase. Tras el proceso de limpieza del dataset obtuvimos un total de 18.027 imágenes correctamente clasificadas.

Tras analizar el histograma 4 de frecuencias de las clases de nuestro dataset vimos que existe un gran desbalanceo en el número de ejemplos que tiene cada clase. Esto supone un problema ya que es muy probable que al entrenar los modelos estos aprendan muy bien a clasificar las clases mayoritarias e ignore completamente a las clases minoritarias. Este problema lo afrontaremos con distintas métricas y técnicas

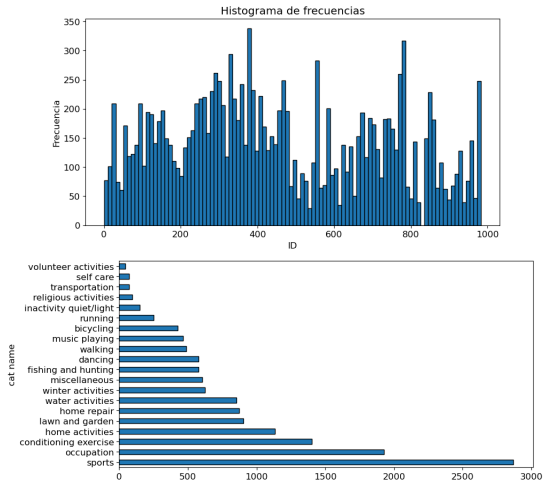

Figure 4. Histograma de la frecuencia de clases en actividades y categorías

que discutimos más adelante.

El protocolo de validación experimental utilizado es *hold-out* que consiste en tener una única partición del conjunto de datos en entrenamiento y test. Este enfoque es útil principalmente cuando el entrenamiento es muy costoso computacionalmente, como es el caso de las redes neuronales, o se tienen muchos datos ya que el conjunto es suficientemente representativo de la realidad. En la aplicación del protocolo de validación a nuestro problema hemos optado por dividir el conjunto de datos inicial en un conjunto de entrenamiento con un 80% de los datos totales y un conjunto de test con el 20% restante. A su vez el conjunto de entrenamiento lo dividimos para sacar un conjunto de validación que está formado por el 10% de los datos totales del conjunto de entrenamiento para verificar que el entrenamiento se está realizando correctamente y el modelo generaliza sin llegar a sobreentrenarlo. Para mitigar un poco el problema del desbalanceo de datos la división en los distintos conjuntos de entrenamiento, validación y test se ha realizado manteniendo la proporción de elementos por clases.

5.1. Clasificación de las actividades

Antes de entrenar los modelos para la clasificación de las actividades hemos realizado una serie de experimentos para determinar ciertos parámetros con el objetivo de obtener el mejor rendimiento posible.

Basándonos en el artículo *Effect of batch size on training dynamics* [6] en el que se discute la efectividad de los tamaños de batch en los resultados del entrenamiento así como de la función de optimización usada decidimos realizar un experimento similar para ver qué tamaño de batch y optimizador es más adecuado para nuestro problema. Para ello, realizamos el entrenamiento de la red VGG con tamaños de mini batch 32, 64, 128 y otros dos entre-

Configuración	Opciones	Prec(%)	Mejor
Tamaño lote	[32, 64, 128]	[70.6, 69.29, 65.77]	32
Optimizador	[Adam, SGD]	[74, 61.34]	Adam
Aumento datos	[Sí, No]	[73.11, 72.84]	Sí
Pesos clases	[Sí, No]	[69.3, 72.49]	No

Figure 5. Configuraciones consideradas a la hora de entrenar el modelo

Modelo	VGG16	ResNet18	GoogLeNet
Pérdida (train)	0.201011	0.200798	0.836987
Pérdida (valid)	0.976770	1.078472	1.211387
Tasa de acierto (%)	79.28	77.06	73.32
Puntuación F1	0.782865	0.754634	0.713459

Figure 6. Clasificación de actividades. Tabla de resultados en entrenamiento

namientos para los optimizadores Adam y SGD.

El artículo concluye con que, en general, es mejor tamaños de mini batch pequeños y usar el optimizador SGD. Respecto al tamaño de batch nuestros resultados fueron igual que en el artículo obteniendo un mejor rendimiento con tamaños pequeños pero respecto al optimizador los resultados fueron mejores al utilizar Adam.

Para afrontar el problema del desbalance de datos planteamos utilizar pesos en la función de pérdidas para dar mayor importancia a las clases minoritarias durante el entrenamiento y aplicar aumento de datos. El experimento se realizó de nuevo mediante el entrenamiento de una red VGG. Aunque parezca poco intuitivo la red que se entrenó sin aplicar pesos a las clases obtuvo un mejor rendimiento, por otro lado, la red que se entrenó con aumento de datos también obtuvo un mejor rendimiento. Un resumen de estos experimentos realizados se puede encontrar en la tabla 5.

Para elegir la tasa de aprendizaje a usar durante el entrenamiento de los modelos utilizamos la técnica introducida en el artículo *Cyclical Learning Rates for Training Neural Networks* [7] que consiste en entrenar el modelo con distintas tasas de aprendizaje y elegir aquella en la que la tasa de decrecimiento del error es máxima. Realizamos el experimento con los tres modelos y obtuvimos valores entre 0.001 y 0.003 por lo que optamos por utilizar un valor intermedio de 0.002 para todos los modelos.

Además, utilizamos dos llamadas de retorno (también conocidas como callbacks). La primera de ellas para guardar el mejor modelo encontrado hasta la época actual, y la segunda utilizada para finalizar el entrenamiento si el modelo no mejora en dos épocas consecutivas. Este último método se denomina *early stopping* y es ampliamente usado para evitar el sobreajuste a los datos de entrenamiento.

Para medir el rendimiento de los modelos normalmente se utiliza la matriz de confusión, en nuestro caso al tener una gran cantidad de etiquetas es inviable, por esta razón hemos optado por utilizar dos métricas: tasa de acierto y la puntuación F1. La tasa de acierto es el número de imágenes cuya etiquetas se han predicho correctamente dividido entre el número de imágenes total. Esta métrica es la más utilizada debido a que nos indica cuánto hemos acertado, sin embargo, en ciertos casos puede dar lugar a errores o confusiones, sobre todo si las clases están desbalanceadas. Para solucionar este problema utilizamos la puntuacion F1 que combina dos métricas: la precisión y el *recall*

$$F1 = \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

En nuestro problema utilizaremos la puntuación F1 ponderada debido a que las clases están desbalanceadas. Esta variante consiste en

$$F1\ ponderada = \sum_{i=1}^{N} w_i \cdot F1_i$$

donde w_i es el peso de la clase i

Con el fin de abordar el problema hemos utilizado las tres arquitecturas mencionadas con anterioridad (VGG, ResNet y GoogleNet). Para ello, hemos realizado un proceso de entrenamiento para decidir cuál es a mejor arquitectura para nuestro problema 6.

Según el rendimiento de los 3 modelos, hemos determinado que VGG16 es el más adecuado a nuestra tarea ya que presenta el mejor porcentaje de acierto y el mejor valor de la puntuación F1.

Una vez elegido el mejor modelo, vamos a calcular el porcentaje de acierto en test. Los resultados son muy similares a los vistos en entrenamiento con un 78,54% de acierto en test frente al 79,28% conseguido en validación, verificando así que el modelo generaliza correctamente.

Aunque no es todo lo bueno que nos gustaría hemos hecho todo lo posible por aumentar el rendimiento del modelo al máximo mediante la experimentación con diversos parámetros. El desbalance de datos y la generalidad de algunas actividades son las causantes de que sea dificil mejorar el rendimiento.

5.1.1 Clasificación de categorías

El conjunto de datos sobre el que trabajamos también está etiquetado por categorías. Existen 20 categorías distintas y hemos decidido utilizar el modelo que nos ha dado mejores resultados en clasificación de actividades (VGG16) para entrenar en clasificación de categorías.

Los resultados obtenidos son bastante favorables con un 79,49% de acierto en validación y un 79.26% en test.

En este caso, sí podemos utilizar la matriz de confusión para ver cómo de bueno es nuestro modelo, ya que solo tiene 20 etiquetas. Se observa como la etiqueta donde más suele fallar es en actividades relacionadas con la ocupación. Esto se debe a que muchas actividades se pueden confundir por estar ocupados haciendo algo.

Sería complicado obtener muchos mejores resultados ya que incluso a un humano le costaría clasificar bien debido a que hay algunas categorías similares que pueden llevar a confusión.

5.2. Estimación de la pose

Para la estimación de las coordenadas de la pose de un individuo se han realizado dos experimentos distintos para cada uno de los cuales se ha entrenado un modelo distinto: estimación usando como base ResNet18 para la estimación de la pose y estimación de la pose usando segmentación, como se ha comentado en el apartado **4.2**. Durante todo este experimento se han descartado las imágenes con más de una persona (según los criterios del dataset) ya que hemos optado por centrarnos en el problema de la estimación de la pose, las cuales suponen unas 11000 imágenes del total mencionando anteriormente. En el caso de la regresión de coordenadas, se han descartado también imágenes con inconsistencias en alguna de las articulaciones, reduciendo su número a unas 7500 imágenes.

5.2.1 Regresión de coordenadas

Figure 7. Estimación de la pose y posición de la cabeza

En este caso se ha optado por hacer *fine tune* y entrenar un modelo basado en ResNet18. Como función de pérdida se ha usado MSE (*Mean Squared Error*) y como métrica MAE (*Mean Absolute Error)*. Se han realizado dos experimentos usando este enfoque, uno tratando de estimar los 16 pares de coordenadas y otro en el que únicamente se ha tratado de estimar la posición de la cabeza en cada imagen. La tabla 8 muestra los resultados obtenidos durante el entrenamiento y el test.

Ambos modelos se comportan de forma similar. Sin observamos el MAE obtenido en ambos casos, sobre un total de 1.0 podríamos pensar que los resultados son aceptables y continuar entrenando el modelo. Sin embargo, una prueba del predictor (figura 7) muestra que los resultados arrojados

Modelo	MSE (Train)	MSE (Valid)	MAE
Pose Completa	0.019	0.012	0.083
Cabeza	0.017	0.013	0.082
Modelo	MSE (Test)	MAE	
Pose Completa	0.013	0.083	
Cabeza	0.013	0.083	

Figure 8. Estimación de la pose. Tabla de resultados en entrenamiento y test

por el predictor no son fiables. Por último, aunque el modelo no cumple su función final, el entrenamiento se ajusta a los resultados de test mostrados, por lo que el problema de la solución propuesta es el enfoque usado para resolverlo.

5.2.2 Estimación mediante segmentación

En este caso, se ha utilizado una arquitectura de U-Net usando como base un modelo preentrenado de ResNet34. Como función de pérdida se ha usado la DiceLoss, la cual es especialmente relevante ya que el desbalanceo en la proporción de fondo respecto al resto de clases es especialmente acusado. Sin embargo, tras entrenar se ha observado que el modelo no consigue aprender ningún patrón relevante y toda la imagen es clasificada como fondo. Los motivos de este fallo pueden ser diversos. El desbalanceo de las clases es evidente. Además, hay que tener en cuenta que los objetivos para la segmentación han sido creados sintéticamente a partir de mapas de calor, por lo que píxeles que no pertenezcan a las articulaciones pueden haber sido asignados a una clase afectando al entrenamiento. Por tanto, este experimento queda descartado como una solución viable.

6. Conclusiones

La clasificación de actividades humanas en imágenes ha resultado ser un problema complejo debido a la ambigüedad y similitud de las mismas impidiendo superar el umbral del 80% de acierto utilizando algunas de las redes neuronales convolucionales más exitosas de la actualidad para la visión artificial. Para poder mejorar el rendimiento de los modelos en este campo es necesario un aumento en el número de imágenes de las clases minoritarias y una mejor taxonomía de las actividades y categorías relacionadas con la pose humana.

Los resultados obtenidos en los experimentos relacionados con la estimación de la pose muestran que se trata de un problema con una complejidad elevada. Las dos soluciones propuestas en este trabajo han resultado insuficientes, si bien la regresión directa es más prometedora. Las líneas para mejorar los resultados obtenidos irían encaminadas a construir un estimador en cascada de forma similar

al método usado en *Deep Pose* [8], siguiendo la regresión directa de las coordenadas. Posteriormente, se podría introducir un nivel más de complejidad añadiendo la detección de múltiples personas en una misma imagen.

References

[1] https://www.codificandobits.com/blog/redes-convolucionales-introduccion/. 2

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 4

[3] Adrian Bulat, Jean Kossaifi, Georgios Tzimiropoulos, and Maja Pantic. Toward fast and accurate human pose estimation via soft-gated skip connections, 2020. 2

[4] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3 SPEC. ISS.):90 – 126, 2006. 2

[5] Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast. Numerical coordinate regression with convolutional neural networks, 2018. 2

[6] Kevin Shen. Effect of batch size on training dynamics. *Medium*, 2018. 4

[7] Leslie N. Smith. Cyclical learning rates for training neural networks. *U.S. Naval Research Laboratory*, 2017. 5

[8] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014. 2, 7

[9] Ce Zheng, Wenhan Wu, Chen Chen, Taojiannan Yang, Sijie Zhu, Ju Shen, Nasser Kehtarnavaz, and Mubarak Shah. Deep learning-based human pose estimation: A survey, 2023. 3