Rock-Paper-Scissors game
Deep Learning

Juan Ramón Baeza Borrego

Agosto 2025

Índice

1. Introducción

A continuación desarrollaremos un trabajo basado en Deep Learning[1] a través del dataset Rock-Papers-Scissors. En el presente conjunto de datos nos encontramos varias imagenes de manos, concretamente 2227, realizando gestos. El objetivo es generar un algoritmo de clasificación de dichas imagenes y que sepa reconocer dicho gesto por sus gestos y lo clasifique.
Para ello haremos uso de DL para generar distintas redes neuronales que consigan el mejor rendimientos y hagan la mejor predicción. Para ello generamos varios modelos que posteriormete los evaluaremos y compararemos en fución del *accuracy*.

En cuanto a la justificación de: ¿por qué utilizar DL? Gracias a esta tecnología, podemos utilizar las redes neuronal convolucionales[2], que tras aplicar capas de convolución puede extraer caracteristicas relevante de los datos de entrada. Este tipo de redes son las más recomendadas en este ambito para los problemas de clasificación de imagenes.s

[1] De aquí en adelante lo denotaremos como DL.
[2] De aquí en adelante lo denotaremos como CNN.

2. Metodología

En cuanto la metodología la podemos dividir en tres apartados:

El primero de ellos es el **preprocesamiento de datos**. En el hemos separado el dataset en tres subconjunto de datos: train que tendrá un 70 % de los datos, val que tendrá el 20 % y por último test que tendrá el 10 %. Con estos subconjuntos entrenaremos el modelo, para posteriomente compararlos y saber que nivel de aprendizaje tiene el modelo. Por otro lado hemos escalado las imagenes y las hemos redimensionado a un tamaño de 200x300. Por último destacar en este apartado que hemos utilizado el uso de generadores como *RPSSequence* y *ImagenDataGenerator*.

El segundo de los apartados es **la construcción de modelos**. El primero de ellos ha sido una CNN básica con unos hiperparametros aleatorios, con el fin de tener una base sobre la que trabajar, de esta forma obtenemos el primero modelo.
El segundo modelo ha sido aplicando un ajuste de dichos hiperparámetros con la finalidad de mejorar el *accuracy*. Para ello hemos aplicado la librería *Keras Turner* el cual nos proporciona el mejor conjunto de hiperparametros para nuestra CNN.
Ahora hemos desarrollado distintos modelos bajo la misma idea, aplicar técnicas de regularización. Es por ello que hemos generado tres modelos, uno de ellos con *Ridge*, otro con *Lasso* y por último, con ambos. La finalidad que perseguimos es evitar el sobreajuste y en caso de detectarse cierto sobre ajuste, en función del tipo de técnica sabremos si nuestro modelo penaliza el coeficiente absoluto, en el caso de *Lasso*, o por el contrario se encarga del cuadrado de los coeficientes, en el caso de *Ridge*.
El penultimo de los modelos que hemos generado es aplicando al dataset técnicas de expansión de datos. Esto implica aumentar los dataset y hacerlos más variables, es por ello que conseguimos, valga la redundacia, las variabilidad en los datos. De esta forma, en cada batch se generen imágenes ligeramente modificadas, por lo que el modelo nunca ve dos veces exactamente el mismo dato.
El último de los modelos está basado en la transferencia del conomiento. Esta idea parte de utilizar modelos que han sido ya entrenados, es decir, no parten de 0. Es por ello que están más sensibilizados a capturar las texturas y bordes.

El último y tercero es **la evaluación.** Tras aplicar los distintos conjuntos de entrenamiento, val y test, obtenemos distintas métricas: *accuracy* y *loss*. De ello sacaremos las siguientes conclusiones.

3. Explicabilidad e interpretabilidad

En este último apartado repasaremos de nuevo los modelos y añadiremos los resultados obtenidos buscando una explicación ante dichos resultados. El primero de los modelos es una red CNN básica con hiperparámetros aleatorios. De esta forma hemos obtenido los siguientes resultados:

Modelo CNN				
Epoch	Accuracy (train)	Loss (train)	Accuracy (val)	Loss (val)
1	44 %	14.79	85 %	0.566
2	85 %	0.47	85 %	0.486
3	88 %	0.32	88 %	0.375
4	93 %	0.25	92 %	0.252
5	96 %	0.12	88 %	0.389

El segundo de los modelos realizaremos un ajuste de hiperparametros. Para ello usamos la libreria, mencionada anteirormente, Keras Turner, la cual nos aporta la mejor combinación posible. Obteniendo estos resultados:

Modelo CNN con hiperparametros ajustados				
Epoch	Accuracy (train)	Loss (train)	Accuracy (val)	Loss (val)
1	77 %	1.2027	89 %	0.3784
2	88 %	0.3406	89 %	0.3588
3	92 %	0.2596	75 %	0.7414
4	90 %	0.3000	91 %	0.2668
5	93 %	0.1912	91 %	0.2745

Aqui podemos observar que hasta el tercer epoch sube linealmente, para en el cuarto descender levemente y en el quinto y último epochs tener el máximo de trains accuracy. A su vez la función perdidia en train es al más baja. De la misma forma, respecto a val, obtenemos uno de los mejores resultados y aunque no sea la mejor funcion perdida, es igualmente uno de los mejores resultados. A continuación observaremos cuales son los mejores hiperparametros que nos marca *Keras Turner*.

Los parámetros que hemos ajustado que componen esta última red neuronal son. En la parte convolucional, se mantienen los 32 filtros y una kernel de 3x3. Sin embargo, la función de activación es la tangente. En la regularización este modelo ha optado por no usar el dropout, a diferencia del anterior, posiblemente por la variabilidad de los mismos. Por último, mencionaremos el optimizador que ha pasado de ser adams a rmsprop, puesto que tiende a ir mejor en problemas de imágenes y secuencias.

El tecer tipo de modelo es aplicando técnica de regularización. Como podemos observar, estos son aplicando *Lasso*, *Ridge* y ambos. Los resultados de dicho entrenamiento son:

Modelos con técnicas de regularización		
Models	Accuracy	Loss
Original	32 %	1.1012
Lasso	88 %	0.6883
Ridge	32 %	1.1161
Lasso + Ridge	90 %	0.6591

Elegir una regularización correcta, es crucial, en este casó la mejor opción es utilizar las dos opciones. Aún así, detacamos que la regularización *Lasso* es la que más aporta, por lo que podemos destacar que nuestro dataset se se beneficiaba más de la selección de características que de reducir magnitudes de pesos.

Llegando al final, el cuarto tipo de modelo es aplicando técnicas de expansión de datos. De esta nos hemos encontrado la siguiente tabla:

Modelo CNN con expansión de datos				
Epoch	Accuracy (train)	Loss (train)	Accuracy (val)	Loss (val)
1	37 %	22.6568	67 %	0.8701
2	57 %	1.1442	79 %	0.5884
3	68 %	0.7470	86 %	0.4763
4	80 %	0.5284	85 %	0.4498
5	80 %	0.5273	72 %	0.5309
6	73 %	0.6401	89 %	0.3902
7	79 %	0.5172	92 %	0.3535
8	84 %	0.4418	92 %	0.3284
9	87 %	0.3797	88 %	0.3741
10	83 %	0.4351	90 %	0.3630
11	86 %	0.3731	93 %	0.2832
12	88 %	0.3802	92 %	0.2628
13	86 %	0.4081	94 %	0.2540
14	87 %	0.3592	94 %	0.2240
15	88 %	0.3399	93 %	0.2671
16	91 %	0.2854	95 %	0.1942
17	90 %	0.2988	96 %	0.1929
18	89 %	0.2835	94 %	0.2167
19	93 %	0.2611	90 %	0.3089
20	87 %	0.3544	95 %	0.1788

El modelo CNN entrenado muestra un aprendizaje progresivo y estable a lo largo de las 20 epochs. Desde el inicio, la precisión en entrenamiento era baja (37 %) y la pérdida muy alta (22.65), pero el modelo mejora rápidamente en las primeras epochs, alcanzando un 80 % de accuracy en entrenamiento y 72 % en validación en la quinta epoch. La validación alcanza su punto máximo alrededor de la epoch 16 con 95 % de precisión y una pérdida de 0.1942, indicando que el modelo generaliza bien y no presenta un sobreajuste severo. Al finalizar el entrenamiento, la precisión en validación se mantiene alta (95 %) mientras que la pérdida sigue siendo baja (0.1788), lo que evidencia que el modelo ha aprendido características significativas de los datos y puede realizar predicciones confiables.

El último es la la transferencia del conocimiento. En este caso el modelo termina de mejorarse llegando a una precisión de 99,3 %. Esto son los resultados obtenidos.

Entrenamiento con MobileNetV2				
Epoch	Accuracy (train)	Loss (train)	Accuracy (val)	Loss (val)
1	73 %	0.6087	95 %	0.1571
2	97 %	0.1147	97 %	0.0907
3	97 %	0.0728	99 %	0.0385
4	98 %	0.0563	98 %	0.0599
5	98 %	0.0600	98 %	0.0580
6	99 %	0.0345	98 %	0.0324
7	99 %	0.0299	99 %	0.0259
8	99 %	0.0243	98 %	0.0648
9	100 %	0.0189	98 %	0.0515
10	99 %	0.0278	99 %	0.0175

Por último, visualizaremos una tabla con todos los modelos, y sus mejores resultados:

Comparativa de modelos CNN		
Modelo	Accuracy (val)	Loss (val)
CNN básica	88–93 %	0.176–0.389
CNN + Keras Tuner	91 %	0.2745
L1	88.7 %	0.6883
L2	32.5 %	1.1161
L1+L2	91.4 %	0.6591
CNN + expansión de datos	93.7 %	0.1761
Transferencia de información	99.3 %	0.0344

La CNN básica, el primero modelo, alcanza un rango de precisión en validación de 88–93 %, mientras que la adición de Keras Tuner y técnicas de regularización mejora marginalmente la precisión o controla la pérdida, aunque. La expansión de la información permite una ligera mejora en la generalización, alcanzando 93.7 % de accuracy con pérdida reducida. Finalmente, el Transfer Learning sobresale, logrando un 99.3 % de precisión y una pérdida mínima de 0.0344, lo que indica que reutilizar modelos preentrenados permite capturar características más robustas y acelerar el aprendizaje, superando ampliamente los métodos entrenados desde cero. En general, estos resultados muestran que combinar preentrenamiento con ajustes de hiperparámetros y técnicas de regularización es la estrategia más efectiva para maximizar precisión y minimizar pérdida.

Por último, traemos una matriz de confunción con el fin de observar si hay alguna clase que falle en especial o buscar algún patron.

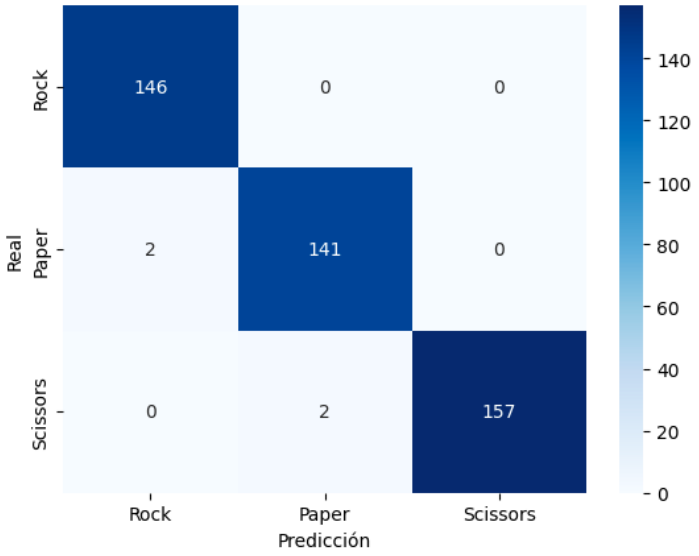

Figura 1: Matriz de confunsión

Sabiamos de antemano que ese patrón de fallo no se daría por la alta predicción. De esta forma confirmamos que el modelo distingue de manera clara las características de cada categoría. La precisión global es del 99 %, lo que evidencia que el modelo generaliza muy bien sobre el conjunto de validación. En resumen, el modelo ha aprendido de forma eficiente los patrones característicos de cada clase, mostrando alta fiabilidad y robustez para la tarea de clasificación de Rock-Paper-Scissors.