REDES NEURONALES CONVOLUCIONALES Y APLICACIONES

TRABAJO FIN DE GRADO

CURSO 2021-2022

AUTOR

NATALIA CASADO BEINAT

CODIRECTORES

D. ANTONIO LÓPEZ MONTES (UCM)

D. ANTONIO MARTÍNEZ RAYA(UNED/UPM)

D. NICOLÁS ANTEQUERA RODRÍGUEZ (UCM)

GRADO EN INGENIERÍA MATEMÁTICA

FACULTAD DE MATEMÁTICAS

UNIVERSIDAD COMPLUTENSE DE MADRID

REDES NEURONALES CONVOLUCIONALES Y APLICACIONES

CONVOLUTIONAL NEURAL NETWORKS AND APPLICATIONS

TRABAJO DE FIN DE GRADO EN INGENIERÍA MATEMÁTICA

DEPARTAMENTO DE ANÁLISIS Y MATEMÁTICA APLICADOS

AUTOR

NATALIA CASADO BEINAT

CODIRECTORES

D. ANTONIO LÓPEZ MONTES (UCM)

D. ANTONIO MARTÍNEZ RAYA(UNED/UPM)

D. NICOLÁS ANTEQUERA RODRÍGUEZ (UCM)

CONVOCATORIA: JULIO 2022

GRADO EN INGENIERÍA MATEMÁTICA

FACULTAD DE MATEMÁTICAS

UNIVERSIDAD COMPLUTENSE DE MADRID

DEDICATORIA

A mis padres, por darme la oportunidad.

AGRADECIMIENTOS

Me gustaría dar las gracias a todas las personas que han hecho posible el poder llegar a este momento, a Antonia por brindarme la oportunidad de estudiar en esta universidad, a mis hermanas, Adri, Lau, por apoyarme y escucharme. A David por las ideas y el tiempo dedicado a ayudarme, por escuchar atentamente y corregirme para poder mejorar. En especial a mis padres, por enseñarme lo que es la constancia y el apoyo incondicional en todo momento. Gracias por mostrarme que todo esfuerzo tiene su recompensa. Por último, una mención especial a mi tutor por aceptar ser el director de mi TFG y por su implicación en el proceso. Gracias también a mis otros 2 codirectores por sus palabras, sus opiniones y su atenta lectura de este trabajo.

RESUMEN

Las redes neuronales artificiales (RNA) surgen en los años 40 con la intención de simular el funcionamiento de las neuronas del cerebro humano. Entre los modelos implementados en esa época destaca el Perceptrón, conocido como la unidad básica de las redes neuronales.

A este primer periodo, le sigue otro de frustración y desprestigio debido al limitado soporte económico y computacional de la época. Pese a ello, algunos investigadores siguieron trabajando y se desarrolló el método de aprendizaje conocido como *Backpropagation*, actualmente el más utilizado en arquitecturas multicapa.

A raíz de este y otros métodos, a principios de la década de 1980 se produjo un importante resurgimiento del interés en el campo de las redes neuronales. Se introduce el Neocognitrón, una red neuronal artificial jerárquica utilizada para el reconocimiento de caracteres manuscritos japoneses y otras tareas de reconocimiento de patrones. Es el origen de la arquitectura de las redes neuronales convolucionales (CNN). Estas redes tienen aplicaciones en el reconocimiento de vídeos, los sistemas de recomendación, la clasificación y segmentación de imágenes, el análisis de imágenes médicas, el procesamiento del lenguaje natural, y las series temporales financieras.

PALABRAS CLAVE

Backpropagation, Convolucionales, Neocognitrón, Perceptrón, Redes neuronales.

ABSTRACT

Artificial neural networks (ANN) emerged in the 1940s with the intention of simulating the functioning of neurons in the human brain. Among the models implemented at that time, the Perceptron, known as the basic unit of neural networks, stands out.

This first period was followed by one of frustration and discredit due to the limited economic and computational support of the time. Despite this, some researchers continued working and developed the learning method known as Backpropagation, currently the most widely used in multilayer architectures.

As a result of this and other methods, in the early 1980s there was a major resurgence of interest in the field of neural networks. Neocognitrón, a hierarchical artificial neural network used for Japanese handwritten character recognition, is introduced. It is the origin of the convolutional neural network architecture (CNN). These networks have applications in video recognition, recommender systems, image classification and segmentation, medical image analysis, natural language processing, and financial time series.

KEY WORDS

Backpropagation, Convolutional, Neocognitrón, Neural networks, Perceptron.

TABLA DE CONTENIDO

DEDICATORIA.....2

AGRADECIMIENTOS.....3

RESUMEN.....1

ABSTRACT.....2

TABLA DE CONTENIDO.....3

ÍNDICE DE FIGURAS.....5

CAPÍTULO 1. INTRODUCCIÓN.....7

CAPÍTULO 2. MARCO TEÓRICO Y CONCEPTOS MATEMÁTICOS.....10

2.1 MÉTODOS DE OPTIMIZACIÓN. GRADIENTE DESCENDIENTE.....10

2.2 CONVOLUCIÓN.....12

2.3 CONVOLUCIONES DILATADAS.....16

2.4 CONVOLUCIÓN NO LINEAL.....17

2.5 CONVOLUCIONES 1D, 2D Y 3D.....20

2.6 SOBREAJUSTE, WEIGTH DECAY Y DROPOUT.....27

2.9 FUNCIONES DE ACTIVACIÓN NO LINEALES Y UNIDADES LINEALES.....29

2.8 SOFTMAX.....31

2.9 AGRUPAMIENTOS (POOLING).....31

CAPÍTULO 3. ARQUITECTURA DE LAS CNN.....33

3.1 ORGANIZACIÓN DE LAS CAPAS EN LAS CNN.....33

3.2 CAPAS INCEPTION.....33

REFERENCIAS.....37

BIBLIOGRAFÍA.....39

WEBGRAFÍA.....40

APÉNDICES.....42

1. BASE NEUROCIENTÍFICA DE LAS CNN 42

2. GOOGLENET. APLICACIONES 46

APLICACIONES. CLASIFICACIÓN DE IMÁGENES 48

3. RECONOCIMIENTO DE ARRITMIAS PELIGROSAS. CNN 1D Y 2D 50

INTRODUCCIÓN 50

ELECTROCARDIOGRAMA (ECG) 50

DESCRIPCIÓN DE LOS DATOS 51

CLASIFICACIÓN DE ARRITMIAS USANDO CNN 1D 53

RESULTADOS 56

CLASIFICACIÓN DE ARRITMIAS USANDO CNN 2D 57

RESULTADOS 59

4. CONCLUSIONES. FUTURAS LÍNEAS DE INVESTIGACIÓN 60

ÍNDICE DE FIGURAS

Figura 1: Imagen del Algoritmo del descenso del gradiente 10
Figura 2: Ejemplo de convolución en 2D utilizando un filtro 3x3 14
Figura 3: Ejemplo de convolución en 3D utilizando un filtro 3x3x3 14
Figura 4: Dimensión de la capa de convolución con $p = k - 1\ y\ s = 1$ 15
Figura 5: Dimensión de la capa de convolución para i = 1 15
Figura 6: Convoluciones dilatadas con d = 1,2,4 16
Figura 7: Función núcleo del Perceptrón 17
Figura 8: Ejemplo de la configuración de una red convolucional 1D 21
Figura 9: Filtro a través de los datos del acelerómetro 22
Figura 10: Datos de un acelerómetro en función del tiempo 22
Figura 11: Esquema de identificación de los latidos ECG 22
Figura 12: Imagen de salida convolucionada de dos dimensiones 23
Figura 13: Mapa de características de dimensión 2D 23
Figura 14: Convolución 3D con Nk filtros. La salida es un mapa de características de profundidad Nk 24
Figura 15: Convolución por grupos (G = 2) 24
Figura 17: Convolución de profundidad 25
Figura 16: Convolución estándar 25
Figura 18: Convolución punto a punto. Filtro de dimensión 1x1x3 25
Figura 19: Convolución punto a punto. 256 filtros de dimensión 1x1x3 26
Figura 21: Etapa II. Point-wise 26
Figura 20: Etapa I. Depth-wise 26
Figura 22: Mapa de características de una convolución 3D 27
Figura 23: Sobreajuste de una red neuronal 27
Figura 24: Función de activación Sigmoidal 29

Figura 25: Función ReLU (*Rectified Linear Unit*) 29
Figura 26: Función ReLU6 30
Figura 27: MaxPooling 2x2 con distintos tipos de desplazamientos 32
Figura 28: Componentes de una capa de red neuronal convolucional típica 33
Figura 29: Módulo Inception. Naive version 34
Figura 30: Módulo Inception con reducción de dimensión 34
Figura 31: Módulo Inception versión naive para la entrada de dimesiones 28x28x192 35
Figura 32: Módulo Inception con reducción de la dimesión para la entrada de dimesiones 28x28x192 35
Figura 33: Arquitectura por capas del Neocognitrón 43
Figura 34: Filtro de Gabor para la extracción de características de textura de la imagen de destino 45
Figura 35: Propagación RMS 46
Figura 36: Módulo Inception con incorporación de unidades ReLU 46
Figura 37: Modelo completo GoogleNet 47
Figura 38: Modelo simplificado GoogleNet 47
Figura 39: Componentes de un electrocardiograma 50
Figura 40: Opciones de entrenamiento de 1D CNN 55
Figura 41: Capas que forman la red CNN 1D 55
Figura 42: Matriz de confusión de la red CNN 1D 56
Figura 43: Distribución de las clases de arritmia en el dataset del entrenamiento de la red ... 57
Figura 45: Capas que forman la red CNN 2D 58
Figura 46: Matriz de confusión de la red CNN 2D 59

CAPÍTULO 1. INTRODUCCIÓN

Las redes neuronales artificiales (RNA) surgen en los años 40 a raíz de la teoría neuronal de Ramón y Cajal de finales del siglo XIX, en plena época del desarrollo de las teorías biológicas del aprendizaje[3].

Estas teorías se basan en el estudio de los fenómenos biológicos que se dan en el cerebro para que el aprendizaje tenga lugar. A raíz de la tendencia del ser humano a implementar de forma computacional muchas de las funciones realizadas por los seres vivos, las redes neuronales nacen con la intención de simular el funcionamiento de las neuronas del cerebro humano.[1]

En 1943, McCulloch y Pitts[4] desarrollaron modelos de redes neuronales basados en sus propios conocimientos sobre neurología, modelos binarios donde utilizaban funciones lógicas elementales. Entre los modelos que se implementaron en esta época destaca el Perceptrón (Rosenblatt,1985)[5].

El Perceptrón está compuesto por dos niveles, un primer nivel de entrada, cuya función es recibir la información del exterior, y un nivel de salida, que es el encargado de procesar los datos obtenidos en la entrada para generar la salida de la red. Este modelo es conocido como la unidad básica de las redes neuronales.

Otro sistema desarrollado en esta época fue ADALINE (ADAptive LINear Element)[6], un tipo de red neuronal creada en 1960 por Widrow y Hoff (Stanford). Su principal diferencia con respecto al Perceptrón es el método de aprendizaje utilizado, una regla de aprendizaje basada en mínimos cuadrados.

En esta primera fase de las redes neuronales artificiales prevalecen los modelos lineales, es decir, a partir de un conjunto de n valores de entrada, la salida de estas redes neuronales es una combinación lineal de los n valores que se ajustan durante la fase de aprendizaje. A este avance le siguió un periodo de frustración y desprestigio debido al limitado soporte económico y computacional de la época.

Pese a ello, algunos investigadores siguieron trabajando en el desarrollo de métodos computacionales basados en la neuromorfología[7]. Estos métodos fueron aplicados en problemas de identificación y clasificación de patrones. Cabe destacar a Paul Werbos (1982)[8],

quien desarrolló el método de aprendizaje conocido como *Backpropagation*, actualmente el más utilizado en arquitecturas multicapa.

Una red *Backpropagation* es, en esencia, un Perceptrón con múltiples capas, donde las neuronas artificiales que lo componen utilizan diferentes funciones de activación. Se basa en un ciclo de dos fases, propagación – adaptación. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas siguientes de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas. El algoritmo de backpropagation, por tanto, funciona calculando el gradiente de la función de pérdida con respecto a cada peso mediante la regla de la cadena, calculando el gradiente de una capa cada vez, iterando hacia atrás desde la última capa para evitar cálculos redundantes en la regla de la cadena. Es un ejemplo de lo que se conoce como programación dinámica.

A raíz de este y otros métodos, a principios de la década de 1980 se produjo un importante resurgimiento del interés en el campo de las redes neuronales. Uno de los principales logros de esta etapa fue el modelo de retropropagación para el entrenamiento de redes neuronales profundas, capaces de distinguir unos objetos de otros. En 1980 se introdujo el neocognitrón, una red neuronal artificial jerárquica y multicapa propuesta por Kunihiko Fukushima[9]. Se utilizaba para el reconocimiento de caracteres manuscritos japoneses y otras tareas de reconocimiento de patrones. Es el origen de la arquitectura de las redes neuronales convolucionales (Convolutional Neural Network CNN).

En 1997, Hochreiter y Schmidhuber[10] introdujeron la red LSTM (Long Short-Term Memory) para resolver algunas dificultades matemáticas de modelos anteriores. Esta arquitectura de red neuronal puede procesar no solo datos individuales, sino secuencias completas de datos, por lo que es aplicable en ámbitos como el reconocimiento de voz, detección de intrusos o procesamiento de vídeo, entre otros.

Hoy en día, las redes neuronales artificiales se han utilizado en gran variedad de aplicaciones entre las que destaca el reconocimiento de imágenes. Por ejemplo, actualmente se comercializan circuitos integrados basados en RNA y las aplicaciones desarrolladas con estas redes resuelven problemas cada vez más complejos. La tendencia actual es aprovechar su utilidad para procesar grandes conjuntos de datos.

En este trabajo nos centraremos en explicar de forma exhaustiva la arquitectura y el funcionamiento de las redes neuronales convolucionales, así como el concepto de convolución aplicado a este contexto y las aplicaciones de estas redes en el tratamiento de imágenes y la clasificación de arritmias.

CAPÍTULO 2. MARCO TEÓRICO Y CONCEPTOS MATEMÁTICOS

2.1 MÉTODOS DE OPTIMIZACIÓN. GRADIENTE DESCENDIENTE

En términos generales, la optimización se define como la tarea de minimizar o maximizar una función $f(x)$. Se suele hablar de minimización ya que la maximización equivale a la minimización de $-f(x)$. A esta función $f(x)$ se la conoce como función objetivo o función de pérdida, que servirá para poder definir el error a retropropagar durante la etapa de aprendizaje. La función de pérdida es un baremo de cuan bueno es un modelo de predicción en términos de poder aproximarse lo más posible al resultado esperado.

Uno de los métodos más utilizados para encontrar numéricamente el punto mínimo de una función es el conocido como 'descenso de gradiente'. Una manera muy intuitiva de explicar este método es considerar que la función se asemeja a un terreno montañoso. Se parte de una entrada aleatoria y, tantas veces como sea posible, se avanza en dirección contraria al gradiente, es decir, se pretende bajar la montaña. De aquí sale el concepto de optimización mediante el gradiente.

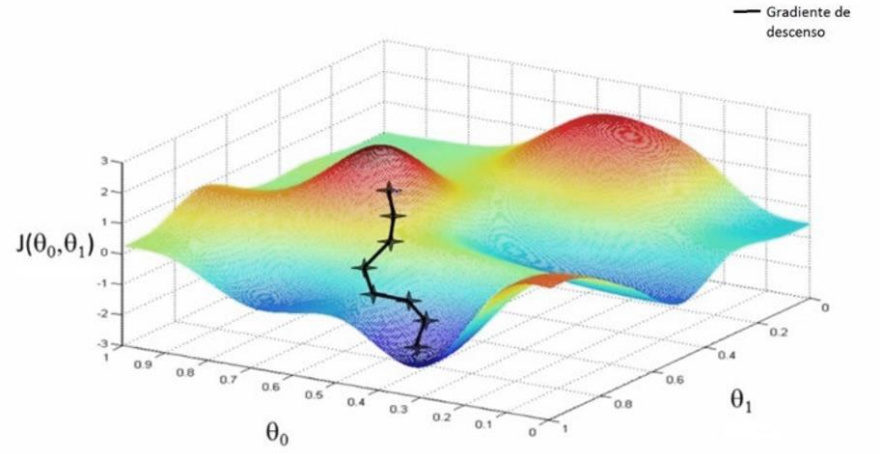


Figura 1[A]: Imagen del algoritmo del descenso del gradiente

El gradiente, de manera no necesariamente formal, puede definirse como una generalización de la derivada, el conjunto de todas las derivadas parciales. Representa la pendiente en el punto que nos encontramos de la función de coste.

Cuando se calcula la pérdida, el objetivo es mejorar la elección de parámetros del modelo. Esto se consigue propagando el error hacia atrás a través de la arquitectura de la red.

Por tanto, en las redes neuronales, al final son los pesos los que terminan ajustándose y minimizando así la función de costes. El proceso de ajuste de parámetros se conoce como aprendizaje.

En el ámbito del aprendizaje con redes neuronales, el problema del Gradiente Descendente Estocástico consiste en minimizar una función objetivo de la forma:

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} J_i(w)$$

Siendo w la variable que minimiza J(w) y que debe estimarse y siendo J_i la i-ésima observación en el conjunto de datos utilizados en el entrenamiento. Se denomina estocástico por la selección aleatoria de las muestras para el ajuste. De modo que $J_i(w)$ es el valor de la función de pérdida en la i-ésima observación y $J\,(w)$ es el riesgo empírico, es decir, el promedio de la diferencia que existe entre los valores deseados y los valores obtenidos con los datos del entrenamiento.

El método del gradiente descendente se usa para minimizar la siguiente función de manera iterativa, con iteraciones en t de la forma:

$$w(t+1) = w(t) - \varepsilon\frac{1}{n}\sum_{i=1}^{n} \nabla J_i(w)$$

De este modo, el método recorre el conjunto de entrenamiento y se produce el ajuste para cada muestra. Analizando la expresión anterior se observa que los gradientes positivos reducen el peso y viceversa, siendo el objetivo minimizar la función de pérdida.

Existen variantes y mejoras de este algoritmo. Una de las más relevantes es fijar la razón de aprendizaje haciéndola depender del número de datos o iteraciones. Si la razón es muy alta, tiende hacía la divergencia de los datos y al contrario hace que la convergencia sea lenta. Así que, si se hace depender de los datos, disminuye a medida que se incorporan nuevas muestras, lo que lleva a un mayor aprendizaje. Esta variante se conoce como el Método del Momento[20] y su ecuación es la que sigue:

$$w(t) = w(t-1) + \alpha\Delta w(t-1) - \varepsilon\nabla J_i(w(t-1))$$

Siendo ε la razón de aprendizaje, $\alpha\epsilon[0,1]$ es el momento constante que controla la velocidad de actualización de $\Delta w(t-1)$. El concepto de momento proviene de la física, de

forma que el vector de pesos w se puede ver como una partícula viajando a través del espacio de parámetros. Adquiere una aceleración a partir de la fuerza que hace que se mantenga en la misma dirección, evitando oscilaciones. Igual que el método del gradiente, w es el vector de parámetros que se pretende estimar para minimizar $J(w)$.

Otra versión de este algoritmo se plantea en caso de que los gradientes incrementen su magnitud exponencialmente, por ejemplo, que los pesos adquieran valores muy altos, en este caso, el entrenamiento se hace inestable y puede llegar a divergir. Esto se conoce como *exploding* del gradiente. A veces se usa lo que se denomina Recorte del Gradiente, se define un umbral T de forma que si $g = \|\nabla J_i(w)\| \geq T$ entonces $\hat{g} = Tg/\|g\|$.

2.2 CONVOLUCIÓN

Las CNN son las redes neuronales que utilizan la convolución en al menos una de sus capas. En general, la convolución es una operación conmutativa que involucra dos funciones con valores reales como argumento.

Para visualizar el concepto de convolución se usa este ejemplo: supongamos que se tiene una fuente de luz variable cuya intensidad se recibe mediante un sensor, que proporciona una salida en una posición x en el tiempo t, es decir x(t). Como x y t son valores reales, en distintos instantes de tiempo, podemos obtener lecturas diferentes. Así que, para conseguir una señal más nítida, podemos realizar un promedio de las salidas de varias medidas. Si además tenemos en cuenta que las medidas más recientes son más relevantes que las que están alejadas en el tiempo, nos planteamos realizar la media ponderada donde las medidas más recientes tienen más peso. Definimos w(a) como la función promedio, siendo a el alejamiento de la medida en el tiempo.

La convolución se define como:

$$s(t) = \int x(a)w(t-a)da$$

En el ámbito de las CNN:

- El primer argumento x para la convolución se conoce como entrada *(input)*
- El segundo w, se conoce como núcleo, filtro o mapa de características *(feature map)*

Cuando se dispone de datos discretos se suele usar una versión también discreta del concepto de convolución:

$$s(t) = (x * w)(t) = \sum_{a=\infty}^{\infty} x(a)w(t-a)$$

La entrada es un vector o matriz multidimensional de datos y el filtro es generalmente un vector o matriz multidimensional de parámetros que se ajustan mediante el proceso de aprendizaje. Estas estructuras multidimensionales se conocen como tensores.

Por otra parte, las convoluciones se pueden realizar sobre más de un eje a la vez, por ejemplo, en el caso de las imágenes I, al ser esencialmente estructuras bidimensionales, necesitan ser tratadas con un núcleo bidimensional K.

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n)$$

Como la convolución es conmutativa, la expresión anterior puede escribirse equivalentemente como sigue:

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} K(i-m, j-n)I(m,n)$$

Aunque esta propiedad no es relevante desde el punto de vista de las CNN, lo que sí que va a jugar un papel importante en este contexto es lo que se conoce como *correlación cruzada*[12]:

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} K(i+m, j+n)I(m,n)$$

Y esta misma expresión puede extenderse fácilmente al ámbito 3D, de forma que en la posición (i,j,k) con un núcleo tridimensional queda como sigue:

$$S(i,j,k) = (I * K)(i,j,k) = \sum_{m}\sum_{n}\sum_{p} K(i+m, j+n, k+p)I(m,n,p)$$

En ocasiones correlación cruzada y convolución son dos términos que se confunden usándose indistintamente. En este contexto, el algoritmo de aprendizaje estima los valores apropiados del filtro en cualquiera de los dos casos.

El propósito de la convolución en las CNN es aplicar un filtro a una imagen para así obtener ciertas características o patrones. La imagen siguiente muestra la convolución del núcleo o filtro, que es la matriz 3x3 que se encuentra en el medio, aplicado sobre un tensor 2D que representa una imagen I_{7x7} en escala de grises. En las convoluciones se define el *campo receptivo* como la región de entrada que contribuye a la salida generada por el filtro. En nuestro ejemplo vemos que es la submatriz de color rojo dentro de la matriz inicial I. Ese filtro se va a ir desplazando por la imagen I y va multiplicando cada una de las celdas del filtro por su correspondiente casilla del campo receptivo. Se genera así otra matriz de menor dimensión que la imagen original, que será la imagen de salida, cuyas celdas son la suma de esas multiplicaciones, es decir, el resultado de aplicarle a la imagen I el filtro.[11]

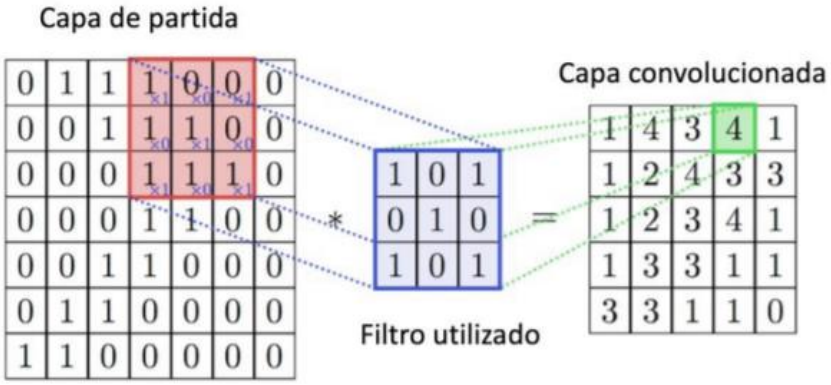


Figura 2[B]: Ejemplo de convolución en 2D utilizando un filtro 3x3

También se dan convoluciones de tipo 3D, simplemente el filtro es un cuboide que se desplaza a través de las dimensiones alto, ancho y alto. Esto puede generalizarse a convoluciones del tipo ND, cuando el número de dimensiones es N.

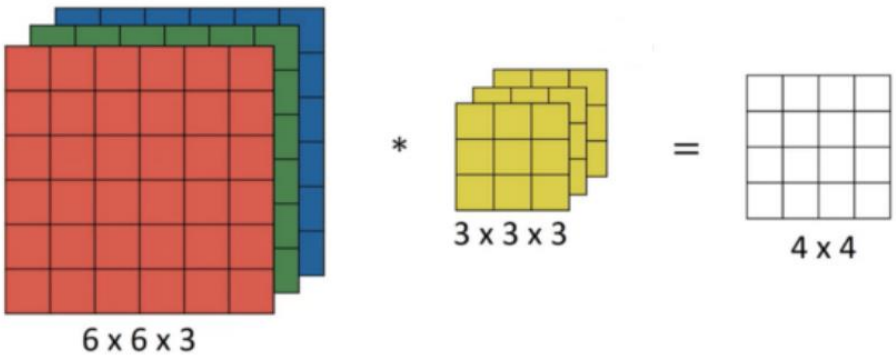


Figura 3[C]: Ejemplo de convolución en 3D utilizando un filtro 3x3x3

En esta imagen se aprecia un núcleo 3D aplicado a una imagen RGB. El proceso es exactamente el mismo, aunque el valor resultante de la imagen salida está en 2D. Se toma como campo receptivo el cubo formando por las celdas 3x3x3 de imagen y se multiplica por las 27

casillas del filtro, es decir, los primeros 9 números del canal rojo con la primera capa del núcleo, luego los 9 que están debajo para el canal verde y después los 9 del azul. Sumando todos esos números y se obtiene la salida. De este modo resulta una matriz 4x4x1.

La colección de núcleos que definen una convolución discreta tienen una forma que se corresponde a alguna permutación del tipo $(n, m, k_1, \ldots\ldots, k_N)$ con:

- n: número de mapas de características de salida
- m: número de mapas de características de entrada
- k_j: dimensión del filtro a lo largo del eje j

La dimensión o_j de una capa de convolución a lo largo del eje j tiene las siguientes propiedades:

- i_j: dimensión de entrada a lo largo del eje j
- k_j: dimensión del filtro a lo largo del eje j
- s_j: distancia entre dos posiciones consecutivas del filtro (*stride*) a lo largo del eje j
- p_j: número de ceros concatenados al comienzo y al final del eje j (*zero-padding*)

Las relaciones que se establecen para obtener la dimensión de la capa de convolución dependen de las variables mencionadas, así, por ejemplo:

- Para i=1 y fijando el valor de s, k, p la dimensión de la capa de convolución es $o = (i - k) + 2p + 1$ (*Zero padding and unit strides*, Relleno con ceros y una unidad de desplazamiento) Figura 5.
- Fijando i, k con $p = k - 1\ y\ s = 1$, la dimensión de la capa de convolución es $o = i + (k - 1)$ (*Full Padding*, Relleno completo para una mayor dimensión de salida) Figura 4.

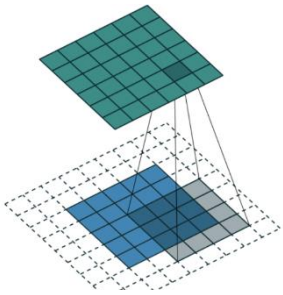


Figura 5[D]: Dimensión de la capa de convolución para i = 1

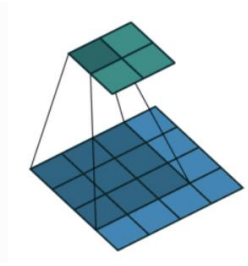


Figura 4[D]: Dimensión de la capa de convolución con $p = k - 1\ y\ s = 1$

2.3 CONVOLUCIONES DILATADAS

Otra operación que se presenta sobre las CNN son las convoluciones dilatadas. Se conocen también como '*atrous convolutions',* proveniente de la terminología francesa *convolutions à trous.* La convolución dilatada, atroz o la convolución con agujeros consiste en inyectar agujeros en el mapa de convolución estándar para aumentar el campo de recepción. En comparación con la convolución original, tiene un hiperparámetro más llamado tasa de dilatación *d*, que se refiere al número de intervalos entre agujeros. Así, por ejemplo, la convolución normal es la tasa de dilatación 1.

La siguiente ecuación define una convolución dilatada:

$$o(x) = \sum_{k=1}^{K} i[x + d * k]w[k]$$

Dada la entrada i[x], obtenemos la salida o(x) con *d* definido previamente y siendo w[k] el filtro de convolución de dimensión k. Este tipo de convolución se utiliza para incrementar el campo receptivo de las unidades de salida sin incrementar la dimensión del filtro.

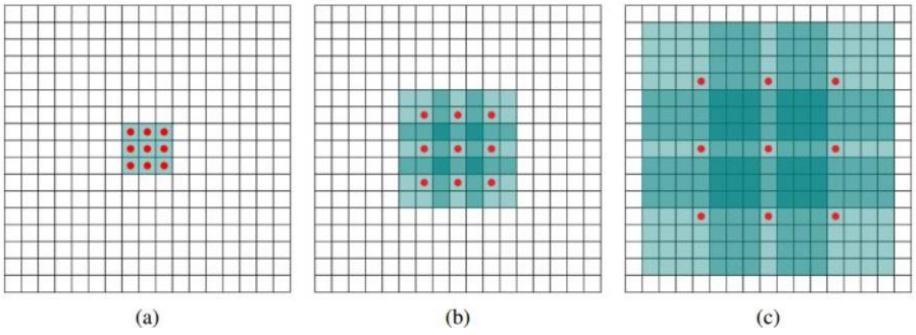


Figura 6[E]: Convoluciones dilatadas con d = 1,2,4

Las 3 imágenes que se ven a continuación muestran un ejemplo ilustrativo con d = 1,2,4. El filtro, k, es de dimensión 3x3, y aplicando la convolución dilatada en cada imagen obtenemos núcleos de dimensión l = 3x3, 7x7 y 15x15. La primera imagen (a), con d = 1 es equivalente a la operación de convolución ordinaria. La imagen (b) se puede entender como que el tamaño del filtro es de 7x7, pero solo el peso de los 9 puntos rojos es distinto de cero. La imagen (c) es una operación de convolución dilatada con d = 4, que alcanza un campo receptivo de 15x15. Es fácil observar que, a diferencia de la convolución tradicional, en esta operación el campo receptivo aumenta exponencialmente.

2.4 CONVOLUCIÓN NO LINEAL

Desde hace varias décadas, algunos estudios se basan en la investigación de la no linealidad en las respuestas de células visuales. De aquí proviene la propuesta de las operaciones de convolución no lineales para las CNN[16] .

En lugar de considerar la convolución como una suma de términos lineales para obtener la respuesta del filtro sobre los datos, se propone también la suma de términos no lineales producidos por interacciones multiplicativas entre todos los pares de elementos sobre los datos de entrada.

De esta forma, transformando las entradas mediante una forma de segundo orden, se consigue una mayor separabilidad entre ellas, igual que ocurre con los núcleos en las máquinas de vectores soporte, un conjunto de algoritmos de aprendizaje supervisado relacionados con problemas de clasificación y regresión. En estos núcleos, la manera más simple de realizar la separación es mediante una línea recta, un plano recto o un hiperplano N-dimensional. Sin embargo, las limitaciones computacionales de las máquinas de aprendizaje lineales hacen que se utilice el recurso de la representación de funciones kernel o núcleo.

Estas funciones proyectan la información a un espacio de características de mayor dimensión y esto hace que aumente la capacidad computacional de la máquina de aprendizaje lineal. Por ejemplo, la función núcleo del Perceptrón se representa de la siguiente forma:

$$k(x_i, x_j) = \|x_i - x_j\|$$

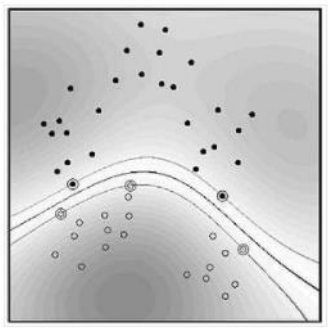


Figura 7[F]: Función núcleo del Perceptrón

El objetivo es estudiar la posibilidad de adoptar un esquema de convolución alternativo para incrementar la capacidad de aprendizaje de las CNN aplicando la teoría de Volterra[17].

El modelo de Volterra plantea una secuencia de aproximaciones mediante funciones continuas, diseñada para representar la relación entrada-salida de los sistemas dinámicos no

lineales, utilizando una expansión funcional polinomial cuyas ecuaciones pueden estar compuestas por términos de órdenes infinitos.

De manera similar a las convoluciones lineales, la convolución de Volterra utiliza núcleos para filtrar los datos. El núcleo de primer orden contiene los coeficientes de la parte lineal. El núcleo de segundo orden representa los coeficientes de las interacciones cuadráticas entre dos elementos de entrada. En general, el núcleo de orden r representa los pesos que las interacciones no lineales entre r elementos de entrada tienen en la respuesta. En Kumar y col. (2009) se han utilizado los núcleos de Volterra para el reconocimiento facial con éxito.[18]

Por tanto, para la convolución se adopta una serie de Volterra de segundo orden. Así dada una región de la imagen de entrada $I \in !^{hxw}$ con *n* elementos ($n = hxw$), redimensionada con un vector $x \in !^{n}$ tal que $x \in \{x_1, x_2, \dots, x_n\}$, la función entrada-salida de un filtro lineal es:

$$y(x) = \sum_{i=1}^{n} w_1^i x_i + b$$

Siendo w_1^i los pesos de los términos lineales de convolución, contenidos en el vector w_1 y b el *bias o sesgo.* La expansión de esta función a una forma cuadrática se hace de la siguiente forma,

$$y(x) = \sum_{i=1}^{n} w_1^i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} w_2^{i,j} x_i x_j + b$$

Donde $w_2^{i,j}$ son los pesos de segundo orden del filtro. Para evitar una doble interacción de cada par de elementos de entrada (x_i, x_j), se adopta una forma triangular superior de la matriz w_2 que contiene sus pesos, de este modo, el número de parámetros a estimar para un núcleo de segundo orden es $\frac{n(n+1)}{2}$. El número de parámetros para un filtro genérico de orden *r* en un modelo de Volterra es, $p = \frac{(n+r)!}{n!\, r!}$.

La forma compacta de la ecuación anterior resulta,

$$y(x) = x^t w_2 x + w_1^t x + b$$

$$w_1 = [w_1^1 \;\; w_1^2 \;\; L \;\; w_1^n]\,; \qquad w_2 = \begin{bmatrix} w_2^{1,1} & w_2^{1,2} & L & w_2^{1,n} \\ 0 & w_2^{2,2} & L & w_2^{2,n} \\ M & M & 0 & M \\ 0 & 0 & L & w_2^{n,n} \end{bmatrix}$$

Los superíndices de la matriz w_2 expresan la correspondencia en las posiciones espaciales de los elementos de entrada $x_i \; y \; x_j$ que interactúan.

La salida resultante de la convolución es:

$$y(x) = \begin{bmatrix} w_2^{1,1} \\ w_2^{1,2} \\ w_2^{1,3} \\ M \\ w_2^{1,n} \end{bmatrix}^t \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ M \\ x_1x_n \end{bmatrix} + \begin{bmatrix} w_1^1 \\ w_1^2 \\ w_1^3 \\ M \\ w_1^n \end{bmatrix}^t \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ M \\ x_n \end{bmatrix} + b$$

La propagación hacia atrás en las ecuaciones de Volterra se realiza adaptando el proceso de propagación a las ecuaciones anteriores, de modo que, si lo aplicamos a la expansión de su forma cuadrática, es necesario obtener los gradientes de las capas de salida y(x) con respecto a los pesos $w_1^i \; y \; w_2^{i,j}$. Pudiendo estimar así el proceso de entrenamiento de los pesos de los núcleos de Volterra. Para propagar el error hacia atrás es necesario obtener los gradientes de la capa de salida y(x) con respecto a las entradas x_i, de modo que, los términos necesarios son:

$$\frac{\partial y}{w_1^i} = x_i \qquad \frac{\partial y}{\partial w_2^{i,j}} = x_i x_j \qquad \frac{\partial y}{\partial x_i} = w_1^i + \sum_{k=1}^{i} w_2^{i,k} x_k + \sum_{k=i}^{n} w_2^{i,k} x_k$$

Una consideración para tener en cuenta radica en el hecho de que $\frac{\partial y}{\partial x_i}$ es una función dependiente de x_i, es decir, en contraposición con los filtros estándar, este término es diferente para cada región del mapa de características, que representa el número de núcleos de convolución en esta capa. Esto significa un coste computacional extra a la hora de propagar el error hacia atrás, proporcional a la dimensión de entrada del mapa de características.

2.5 CONVOLUCIONES 1D, 2D Y 3D

Llegados a este punto, se observa que las convoluciones en CNN constituyen un elemento esencial para este tipo de redes neuronales. En este apartado se explican los conceptos básicos relativos a las convoluciones 1D, 2D y 3D.

Las convoluciones 1D se caracterizan por aplicarse en una única dimensión, en general de naturaleza temporal, como el procesamiento de señales temporales. Un ejemplo de esto es el reconocimiento de voz (Pajares, 2017).[19]

En la actualidad se han desarrollado una gran cantidad de redes neuronales convolucionales 1D[22] y se ha demostrado que para ciertas aplicaciones presentan más ventajas que las convoluciones 2D, algunas de ellas son:

- Diferencias en términos de complejidad computacional. Una imagen de dimensión $n\,x\,n$ que aplica una convolución con un núcleo de dimensión $k\,x\,k$ tiene una complejidad computacional de $O(N^2K^2)$ mientras que para una convolución en 1D, esta se reduce a $O(N\,K)$, que es significativamente mejor.
- La mayoría de las CNN 1D están formadas de 1 o 2 capas ocultas y tienen menos de 10000 parámetros, mientras que las CNN 2D utilizan arquitecturas más complejas con más de 1 millón de parámetros. De modo que, las redes con arquitecturas poco profundas son mucho más fáciles de implementar y de entrenar.
- El entrenamiento de las CNN 2D requiere una configuración de hardware especial, por ejemplo, computación en la nube. Por otro lado, cualquier implementación sobre un ordenador estándar es factible y relativamente rápida para entrenar las CNN 1D con pocas capas ocultas y un pequeño número de neuronas. Son adecuadas para utilizarse en tiempo real y con poco coste, especialmente en móviles o portátiles.

Este tipo de convolución presenta un rendimiento superior para el uso de datos limitados y altas variaciones adquiridas de distintos ámbitos, por ejemplo, electrocardiogramas, estructuras mecánicas o aeroespaciales, circuitos de alta potencia, etc. Como vemos en la siguiente figura, la convolución 1D presenta dos tipos de capas:

1. Las llamadas “Capas de CNN” donde se aplica la convolución 1D, la función de activación y el muestreo (*pooling*).

2. Capas densas completamente conectadas que se asemejan a las capas de un Perceptrón, por eso reciben el nombre de “Capas MLP”.

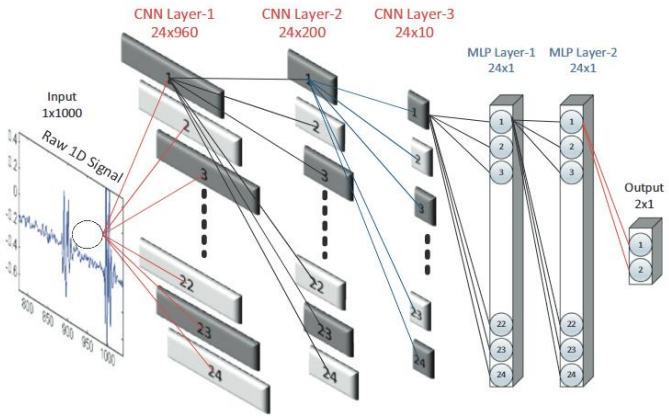


Figura 8[G]: Ejemplo de la configuración de una red convolucional 1D

La configuración de una red convolucional 1D está formada por los siguientes parámetros:

- Número de capas de CNN y MLP ocultas, en este caso, hay 3 capas CNN y 2 MLP.
- Tamaño del núcleo en cada capa de CNN, en la imagen anterior, el núcleo es de dimensión 41 para todas las capas de las CNN ocultas.
- Factor de muestreo en cada capa, que es 4 en este caso.
- Elección de las funciones de agrupación y activación.

La principal ventaja de las CNN 1D, como ya se ha explicado previamente, es su baja complejidad computacional, la única operación con un coste significativo son simplemente sumas ponderadas lineales de dos matrices 1D. Esta operación puede ejecutarse en paralelo durante las operaciones de propagación hacia delante y hacia atrás.

Un ejemplo de estas redes son las tareas de reconocimiento de actividad a partir de datos de un acelerómetro, miden si la persona está de pie, caminando, saltando, etc. Estos datos tienen 2 dimensiones. La primera dimensión es el paso del tiempo y la otra son los valores de la aceleración en 3 ejes. En el siguiente ejemplo se muestran datos que han sido recopilados de un acelerómetro que una persona llevaba en el brazo. Los datos representan la aceleración en los 3 ejes.

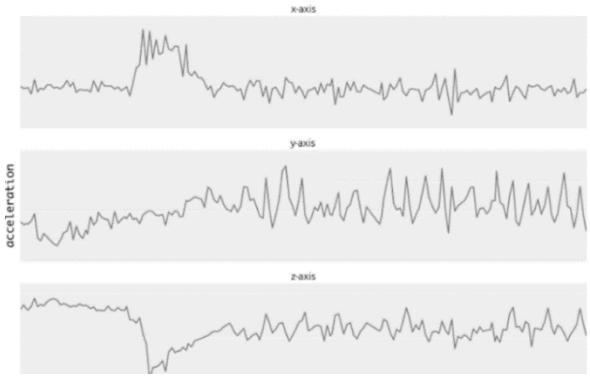


Figura 10[H]: Datos de un acelerómetro en función del

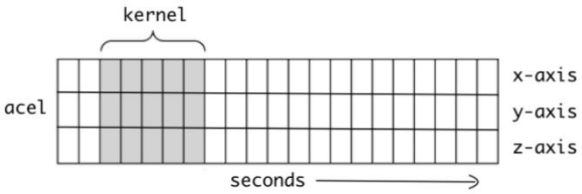


Figura 9[H]: Filtro a través de los datos del acelerómetro

El gráfico de la derecha ilustra cómo se moverá el núcleo en los datos del acelerómetro. Cada fila representa la aceleración de series de tiempo para algún eje. El núcleo solo puede moverse en una dimensión a lo largo del eje del tiempo.

Otro ejemplo es aplicado al monitoreo de electrocardiogramas en tiempo real (ECG). A pesar de todos los métodos propuestos para la clasificación de ECG, las variaciones entre las señales en los distintos pacientes hacen inviable el modelado y aprendizaje de patrones. En otras palabras, un modelo entrenado sobre el ECG es probable que no pueda clasificar con precisión la señal ECG de otra persona. Para abordar este problema, el primer uso que se le da a la CNN 1D es la identificación de latidos de ECG, el objetivo es el de clasificar cada latido de ECG en una de las cinco clases:

- N (Latidos creados del modo sinusal)
- S (Latidos ectópicos supraventriculares)
- V (Latidos ectópicos ventriculares)
- F (Latidos de fusión)
- Q (Latidos inclasificables)

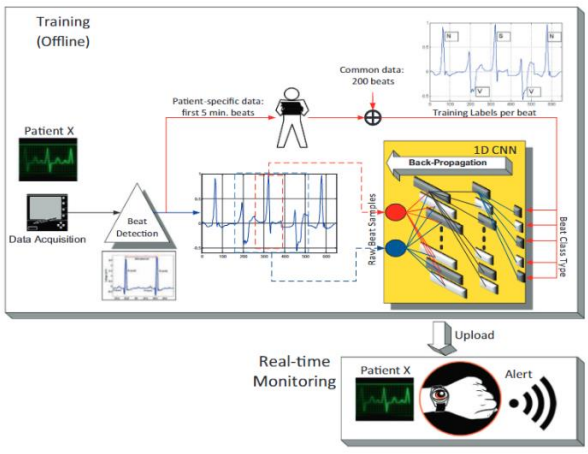


Figura 11[I]: Esquema de identificación de los latidos ECG

La imagen representa como, para cada paciente con arritmia, se entrena una CNN 1D con los datos de entrenamiento específico del paciente. La configuración utilizada en todos los experimentos tiene 3 capas convolucionales ocultas y 2 capas densas, con 32 y 16 neuronas en la primera y segunda capa convolucional oculta y 10 neuronas en la capa densa oculta. La capa de salida es de dimensión 5, que es el número de clases de latidos.

El principal inconveniente de estas redes, que se puede trasladar a las CNN en general es el uso del mismo tipo de neurona en toda la red, es decir, las CNN son redes homogéneas. Sin embargo, estudios recientes han revelado la primera red tipo CNN sin esta limitación. Estas redes se denominan Redes Neuronales Operativas (ONN)[23]. Las ONN pueden ser heterogéneas y encapsular neuronas con cualquier conjunto de operadores, lineales o no lineales, aprender funciones o espacios altamente complejos y multimodales con una complejidad de red y datos de entrenamiento mínimos. Por tanto, se prevé que estas redes reemplacen a las CNN 1D con soluciones más compactas y en particular para patrones complejos.

La convolución 2D se ha explicado exhaustivamente en los apartados anteriores, queda realizar una serie de matizaciones como paso previo a las convoluciones 3D. Ambas son de naturaleza espacial, aplicándose a imágenes, de forma que la salida devuelve una nueva imagen. Cuando la imagen posee un único canal de dimensión HxW y se le aplica un filtro de dimensión $h_k \; x \; w_k$, el resultado es una imagen de salida de dos dimensiones.

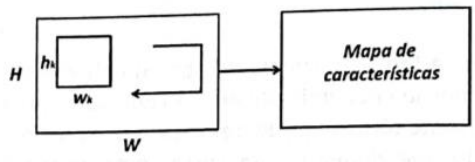


Figura 12[J]: Imagen de salida convolucionada de dos dimensiones

Por otra parte, cuando una imagen de entrada posee C canales, como puede ser una imagen a color con C = 3, el resultado de aplicar un filtrado con un núcleo de convolución de dimensión $h_k \; x \; w_k \; x \; C$ es un mapa de características de dimensión 2D.

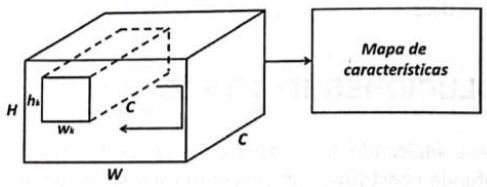


Figura 13[J]: Mapa de características de dimensión 2D

En este mismo caso, si en lugar de considerar un único núcleo se utiliza N_k núcleos (filtros) del tipo 3D se obtiene un mapa de características de salida con una profundidad N_k. Es decir, el número de núcleos de convolución en esta capa es N_k.

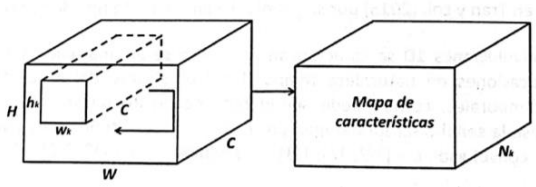


Figura 14[J]: Convolución 3D con N_k filtros. La salida es un mapa de características de profundidad N_k.

Si en lugar de C como número de canales se considera un número de fotogramas o *frames* (T) para el caso de secuencias de video o videoclips, este tipo de convolución se aplicaría de la misma manera. Si además cada *frame* posee C canales se puede seguir considerando una convolución de esta misma naturaleza simplemente tomando la imagen de entrada como un tensor 3D de dimensión $CT\ x\ H\ x\ W$ donde la dimensión CT es el producto de C canales y T *frames*. Por ejemplo, en una secuencia de video de 16 *frames* de color, la dimensión de CT sería $16x3 = 48$.

Ahora bien, si el mapa de características de entrada con C canales se divide en G grupos, cada grupo de dicho mapa constará de C/G canales. En la siguiente figura se muestra un ejemplo de la convolución por grupos donde el mapa de características de la figura anterior se divide en dos grupos. Sobre cada uno de los grupos se aplican $N_k/2$ filtros para obtener un mapa de características de salida con N_k canales.

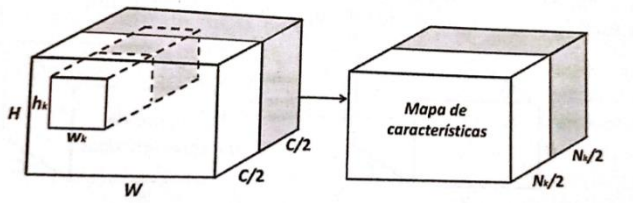


Figura 15[J]: Convolución por grupos (G = 2)

Este tipo de convolución reduce el coste total computacional en un factor 1/G con respecto a la convolución estándar. Siendo G los grupos en los que se divide el mapa de características de entrada con C canales.

Existen dos tipos de convoluciones relevantes en el ámbito de las CNN en su aplicación en el ámbito de las imágenes:

- CONVOLUCIÓN DE PROFUNDIDAD (*depth-wise*)

  A diferencia de las CNN estándar, en las que la convolución se aplica a la vez a todos los canales C, en este tipo se aplica a un único canal a la vez.

  En la figura de la izquierda vemos que se aplica la convolución estándar, mientras que en la figura de la derecha se utiliza cada canal de filtro solo en un canal de entrada. Tenemos un filtro de 3 canales y una imagen de 3 canales. Lo que hacemos es dividir el filtro y la imagen en tres canales diferentes y luego aplicar la convolución estándar a la imagen correspondiente con el canal correspondiente y luego apilarlos nuevamente.

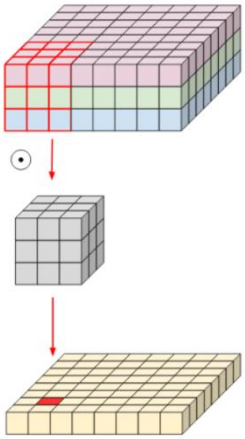


Figura 16[K]: Convolución

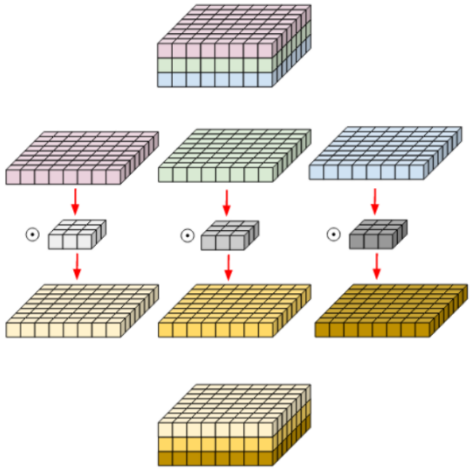


Figura 17[K]: Convolución de profundidad

- CONVOLUCIÓN PUNTO A PUNTO (*point-wise*)

  En esta operación de convolución se aplica la convolución 1x1 en los C canales. En términos de imágenes, esta convolución proyecta un píxel a lo largo de todos los canales a un único valor de salida sin tener en cuenta su vecindad. Se utiliza para aumentar o reducir el número de canales gracias al número de filtros N, de forma que si N < C se produce reducción y si, por el contrario, N > C, una expansión.

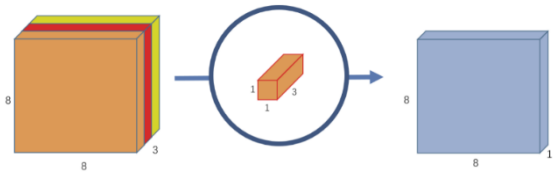


Figura 18[L]: Convolución punto a punto. Filtro de dimensión 1x1x3

Este núcleo tiene la profundidad de cuantos canales tenga la imagen de entrada, en este caso, 3. Por lo tanto, iteramos un núcleo de dimensión 1x1x3 a través de nuestra imagen 8x8x3, para obtener una imagen 8x8x1.

Podemos crear 256 núcleos de dimensión 1x1x3 que generen una imagen de 8x8x1 cada uno para obtener una imagen final de forma 8x8x256. Es decir, $N = 256$ y como N > C, se produce una expansión.

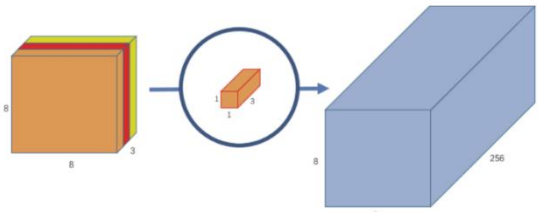


Figura 19[L]: Convolución punto a punto. 256 filtros de dimensión 1x1x3

Existe también lo que se conoce como convolución separable *depth-wise*. Esta convolución se originó a partir de la idea de que la profundidad y la dimensión espacial de un filtro se pueden separar, de ahí el nombre. Esencialmente, constituye una herramienta muy poderosa para reducir el coste computacional y el número de parámetros involucrados.

La convolución se aplica en dos pasos, primero se realiza una operación *depth-wise* mediante un filtro de dimensión 3, sobre el mapa de características para generar una salida; de manera que, al aplicar esta operación sobre los 3 canales, se obtiene un mapa de dimensión 3x8x8 sobre el que se aplica la convolución *point-wise* con un núcleo de dimensión 3x1x1 que genera la correspondiente salida de dimensión 1x8x8.

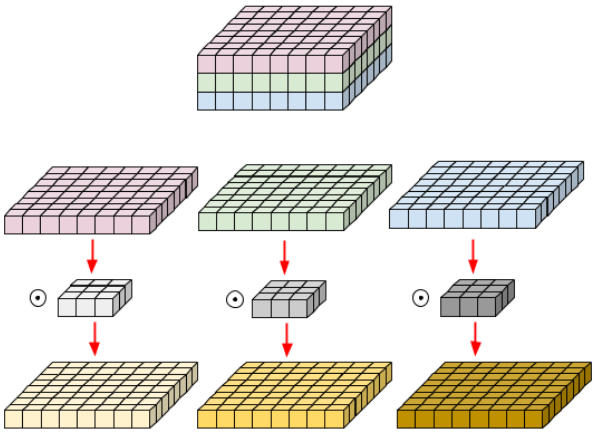


Figura 20[K]: Etapa I. Depth-wise

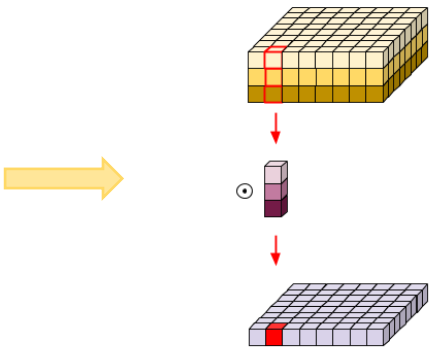


Figura 21[K]: Etapa II. Point-wise

Vamos a finalizar este apartado con las convoluciones 3D, que, a diferencia de las anteriores, su aplicación principal son las secuencias de video. Una secuencia de video está formada por un tensor de dimensiones $CxTxHxW$, siendo C el número de canales, T la longitud del número de *frames* que contiene la secuencia de video, H y W el alto y ancho de cada *frame* respectivamente. En la figura se muestra un esquema de esta convolución, observando a la izquierda el mapa de características 3D de entrada, junto con un núcleo que actúa sobre un entorno de vecindad centrado en el punto que se indica (E) y que genera como resultado el valor del mapa de características indicado en (S).

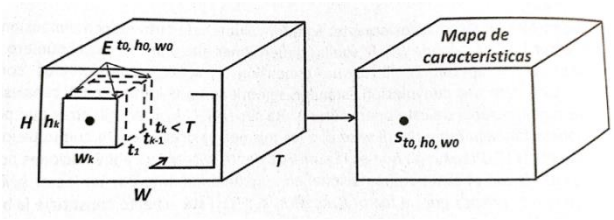


Figura 22[M]: Mapa de características de una convolución 3D

2.6 SOBREAJUSTE, WEIGTH DECAY Y DROPOUT

Un problema común en las redes neuronales es lo que se conoce como sobreajuste o *overfitting*, cuando se ajustan un número elevado de pesos en gran cantidad de neuronas.

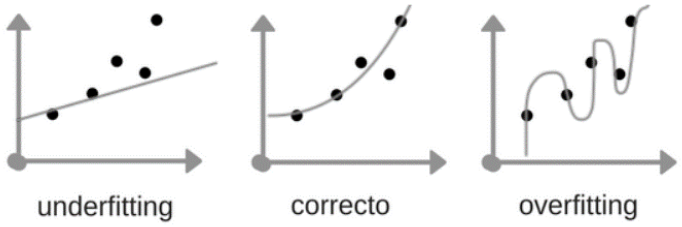


Figura 23[M]: Sobreajuste de una red neuronal

La cuestión es cuál es el mejor ajuste: el que pasa por todos los puntos mediante un polinomio con un grado elevado o una línea recta. Aunque con la primera opción el modelo se ajusta bien a los datos, en el momento de decisión no es el adecuado.

Un método para evitar el sobreajuste es aplicar una regularización, que básicamente consiste en penalizar la pendiente. En general, consiste en modificar la función objetivo a minimizar añadiendo términos adicionales que penalicen los pesos con valores elevados. Por ejemplo, dada la función objetivo Z se le puede añadir el término $\lambda f(\theta)$ de forma que $f(\theta)$ crece a medida que el argumento θ también lo hace, siendo λ el coeficiente de regularización.

El valor de λ determina la protección contra el sobreajuste, de forma que si es cero no se realiza ninguna acción y si es demasiado grande, el modelo dará prioridad a minimizar θ al tratar de encontrar los valores de los parámetros que mejor se ajustan al conjunto de datos de entrenamiento.

El tipo más común de regularización es la regularización L_2. Se implementa aumentando la función error con la media del cuadrado de los coeficientes de todos los pesos de la red neuronal. Es decir, para cada peso *w* en la red neuronal, se agrega $0.5\lambda w^2$ a la función error. Es interesante saber que en la regularización L_2, la aplicación del gradiente descendiente para cada peso tiende a cero. Como consecuencia, se le conoce también como disminución de peso (*weigth decay*).

En el ámbito de las redes neuronales, para mitigar el efecto del sobreajuste se propone eliminar un determinado tipo de neuronas (*dropout*). La elección de estas neuronas se realiza de manera aleatoria. El *dropout* puede aplicarse tras la salida de una determinada capa, como puede ser tras un agrupamiento (*pooling*).

Para que no se produzca el sobreajuste, a parte de los mecanismos de regularización y *dropout* mencionados, podemos considerar algunos más:

- Dividir el conjunto de datos en tres conjuntos: aprendizaje, validación y test
- Validación cruzada: el conjunto de datos se divide en k grupos, uno de ellos es el conjunto de prueba y los demás de entrenamiento y se repite el proceso hasta que cada conjunto haya sido utilizado como conjunto de prueba. De este modo se puede determinar el error del test calculando el error promedio de cada una de las pruebas.
- Aumento de los datos.
- Selección de los datos: seleccionar las características más relevantes de las muestras mediante el Análisis de Componentes Principales.
- Detención temprana: entrenar el modelo para un conjunto grande de datos y trazar gráficamente la evolución de la función de pérdida.

2.9 FUNCIONES DE ACTIVACIÓN NO LINEALES Y UNIDADES LINEALES

Una función de activación clásica es la función sigmoidal:

$$f(a, x, c) = \frac{1}{1 + e^{-a(x-c)}}$$

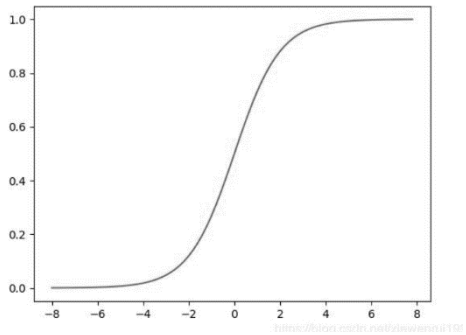


Figura 24[N]: Función de activación Sigmoidal

Dependiendo del signo del parámetro a, la función sigmoide se abre hacia la izquierda o hacia la derecha, lo que hace que sea una buena opción a la hora de representar los conceptos de "muy negativo" o " muy positivo". Sin embargo, esta función tiene varios problemas.

1. Saturación del gradiente: cuando el valor de la función se aproxima a los extremos [0,1], el gradiente de la función tiende a infinito y esto repercute en el ajuste de pesos de la red.
2. Pesos positivos: el valor medio de la función de salida no es 0, esto hace que los pesos tiendan a ser positivos.

Estas cuestiones hacen aumentar el tiempo computacional del entrenamiento y provocan una convolución lenta de los parámetros afectados. Por tanto, para este tipo de redes neuronales se recurre a la función Unidad Lineal Ratificada (ReLU, *Rectified Linear Unit*) :

$$f(x) = \max(0, x)$$

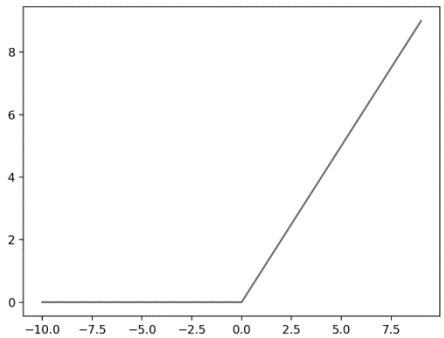


Figura 25[N]: Función ReLU (*Rectified Linear Unit*)

Esta función tiene varias ventajas sobre las funciones sigmoidales:

1. ReLU es muy fácil de calcular ya que se hace una comparación entre su valor de entrada y el valor 0.
2. Su derivada es 0 o 1 dependiendo si su entrada es respectivamente negativa o positiva. Este hecho es muy importante para la retropropagación durante el entrenamiento ya que ayuda a prevenir el crecimiento exponencial del tiempo de computación de este.
3. A diferencia de las funciones sigmoidales, que tienen derivadas que tienden a 0 a medida que se acercan al infinito, las funciones ReLU siempre permanecen en un 1 constante. Esto permite que continúe la propagación hacia atrás del error y el aprendizaje, incluso para valores altos de la entrada.

Algunos tipos de redes definen una variante de esta función como sigue:

$$ReLU6(x) = f(x) = \min(\max(x, 0), 6)$$

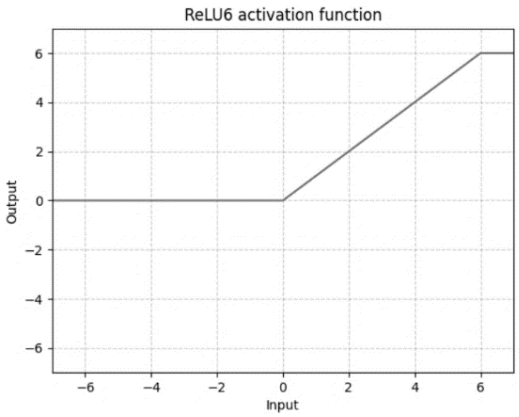


Figura 26[N]: Función ReLU6

La principal diferencia entre ReLU y ReLU6 es que ReLU permite valores muy altos en el lado positivo mientras que ReLU6 se restringe al valor 6 como el máximo que toma la función. Aunque está compuesta de 3 funciones lineales, RELU6 es una función no lineal. Otra función de activación no lineal es la función Softmax que se introduce a continuación.

2.8 SOFTMAX

La función exponencial normalizada o softmax aparece normalmente en las últimas capas ocultas. Se define como:

$$softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} \qquad x = (x_1, x_2, \dots, x_n) \in i^n$$

Se utiliza para proyectar un vector n-dimensional x, de valores reales en un vector n-dimensional de valores reales en el rango [0,1], de forma que cada componente i de $softmax(x)_i$ se obtiene teniendo en cuenta las n componentes del vector x.

Por ejemplo, si tenemos una red neuronal con cuatro neuronas en la capa de salida con un valor x = $(1.2, 2.0, 5.6, 3.1)$, los valores generados por la función $softmax(x)$ serian $(0.011, 0.024, 0.892, 0.072)$.

De manera que, se aplica la función exponencial a cada elemento x_i del vector de entrada *x* y se normalizan esos valores por la suma de las exponenciales. Esto hace que la suma de las componentes del vector salida valga uno.

2.9 AGRUPAMIENTOS (POOLING)

En las redes neuronales, las capas denominadas de agrupamiento o *pooling* proporcionan invariancia a pequeñas translaciones de la entrada. Existen varias operaciones de este tipo, una de ellas es el máximo (*max*), que consiste en dividir la entrada en ventanas, produciendo como salida el máximo de cada una. Otra de las operaciones es la media, aquí la salida es el resultado de esta operación sobre la ventana (Boureau y col.,2010)[14]

Generalmente, la operación de *pooling* se aplica para reducir la dimensionalidad, de forma que, en cada paso, la posición de la ventana se actualiza con los desplazamientos (*strides*). Cuando la ventana se sitúa próxima a los bordes de la imagen, algunos de los elementos de esa ventana pueden quedar fuera de los elementos de entrada. De modo que, para obtener los valores de estas regiones, la entrada puede extenderse con valores de cero, lo que previamente se ha denotado como *zero-padding*.

El *pooling* puede considerarse como una operación de convolución zero-padding, de forma que la relación que describe de manera general y que sirve para cualquier tipo de *pooling* es:

$$o = \left[\frac{i-k}{s}\right] + 1; \quad \text{para cada } i,k,s.$$

Esta operación ayuda a hacer que la representación quede invariante a pequeñas translaciones de la entrada, es decir, si se traslada la entrada mediante un pequeño desplazamiento, los valores de muchas salidas sobre las que se ha aplicado el pooling no cambian.

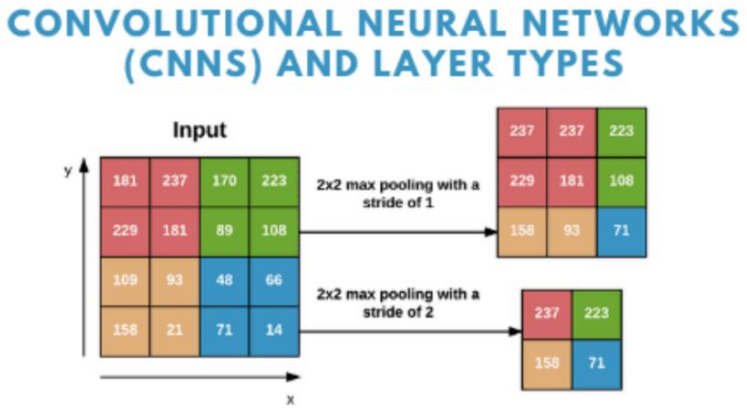


Figura 27[O]: MaxPooling 2x2 con distintos tipos de desplazamientos

En este ejemplo de agrupamiento por el máximo, observamos que la imagen original está dividida en 4 ventanas de distintos colores. En la primera reducción de la imagen original, tenemos una unidad de desplazamiento, es decir, se hace el máximo de las submatrices 2x2 que forma la matriz original, desplazándose cada vez una posición. En la otra matriz, tenemos dos unidades de desplazamiento, luego tomamos el máximo de cada ventana de color.

CAPÍTULO 3. ARQUITECTURA DE LAS CNN

3.1 ORGANIZACIÓN DE LAS CAPAS EN LAS CNN

Existen dos terminologías que se usan normalmente en este tipo de redes para definir el tipo de capa:

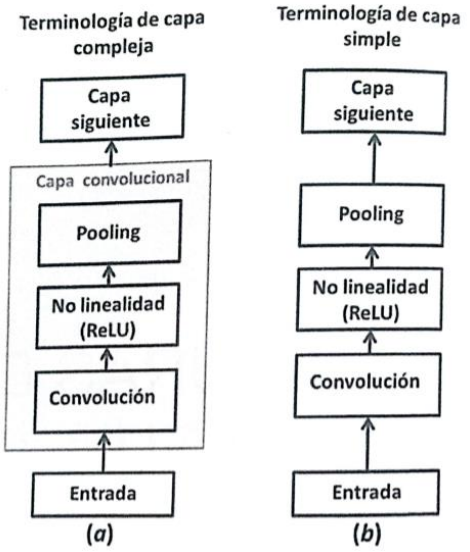


Figura 28[P]: Componentes de una capa de red neuronal convolucional típica

Capa Compleja:
Está formada por varios estados y constituyen la capa convolucional, se encarga de procesar la salida de neuronas que están conectadas en regiones locales de entrada.

Capa Simple:
Cada paso del procesamiento es una capa, pero no todas ellas tienen parámetros para aprender.

3.2 CAPAS INCEPTION

La forma más sencilla de mejorar el rendimiento de las redes neuronales profundas es aumentando su tamaño, tanto en profundidad (número de capas o niveles) como en ancho (el número de unidades en cada capa). Un tamaño más grande significa una mayor cantidad de parámetros, esto hace que la red sea más propensa al sobreajuste. Además, un aumento uniforme del tamaño de la red es también el aumento de recursos computacionales.

Las capas Inception han sido diseñadas para reducir el coste computacional de las redes convolucionales, en particular cuando la entrada tiene un número elevado de mapas de características, es decir, el número de núcleos de convolución en esta capa es muy alto. Se realiza una construcción capa por capa en la que se analiza la correlación de las unidades de la última capa agrupándose las más correlacionadas y conectándose en forma de bancos de filtros. Esto significa que se termina con una gran cantidad de grupos concentrados en una sola región que pueden cubrirse con una capa de convolución 1x1 en la siguiente capa.

Sin embargo, se planteó la posibilidad de tener filtros con múltiples tamaños operando en una misma capa. La red se vuelve un poco más ancha en vez de más profunda. La imagen que aparece a continuación es un módulo de inicio ingenuo (*naive version*). Realiza convolución en una entrada con 3 tamaños diferentes de filtros (1x1, 3x3, 5x5) y también se realiza la agrupación del máximo (*max pooling*). Las salidas se concatenan y se envían al siguiente módulo de inicio.

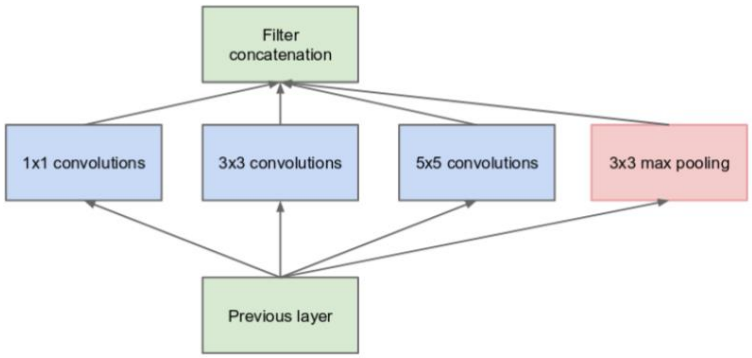


Figura 29[Q]: Módulo Inception. Naive version

Un gran problema con las capas Inception es que incluso con un número reducido de convoluciones con núcleos 5x5 puede llegar ser computacionalmente muy costoso.

Esto lleva a la segunda idea de la arquitectura propuesta: reducir la dimensión de los requisitos computacionales que aumenten excesivamente. Se pueden limitar la cantidad de canales de entrada agregando una convolución adicional de 1x1 antes de las convoluciones 3x3 y 5x5. Aunque este hecho parezca contradictorio se debe a que las convoluciones 1x1 reducen su dimensión antes de realizar la convolución sobre el filtro, además incluyen el uso de funciones ReLU. Esto hace que la dimensión con la que tienen que trabajar las convoluciones más costosas sea menor. El resultado es el siguiente:

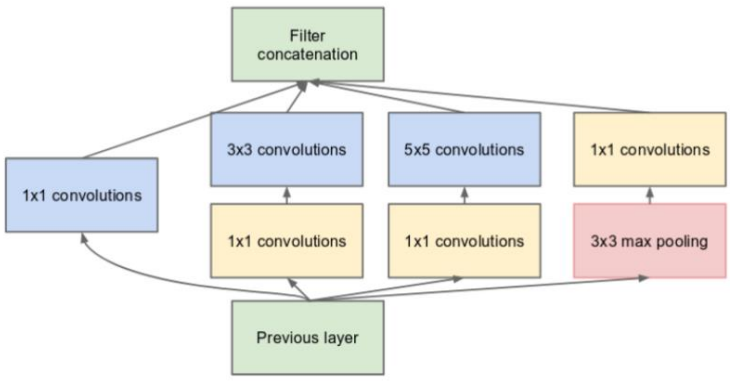


Figura 30[Q]: Módulo Inception con reducción de dimensión

En general, una red Inception consta de módulos, apilados unos sobre otros, con capas de *max pooling* (agrupamientos aplicando el máximo) ocasionales.

Un ejemplo concreto de esta arquitectura de manera más detallada que muestra un módulo Inception en versión *naive* es el siguiente:

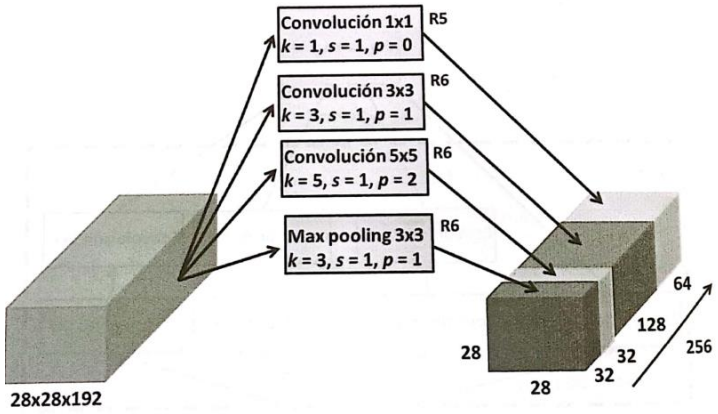


Figura 31[R]: Módulo Inception versión naive para la entrada de dimesiones 28x28x192

La entrada tiene por dimensiones 28x28x192. Se le aplican las operaciones representadas, siendo k la dimensión de los filtros, s el stride (distancia entre dos posiciones consecutivas del núcleo) y p el *padding* (número de ceros concatenados al comienzo y al final del eje). El objetivo es mantener la misma dimensión de entrada en lo relativo al ancho y alto, esto es 28x28, además, el número de filtros aplicados en cada convolución y en el *max pooling* es el que se indica a la salida (64, 128, 32, 32).

La representación equivalente para la capa inception con reducción de dimensionalidad es la siguiente:

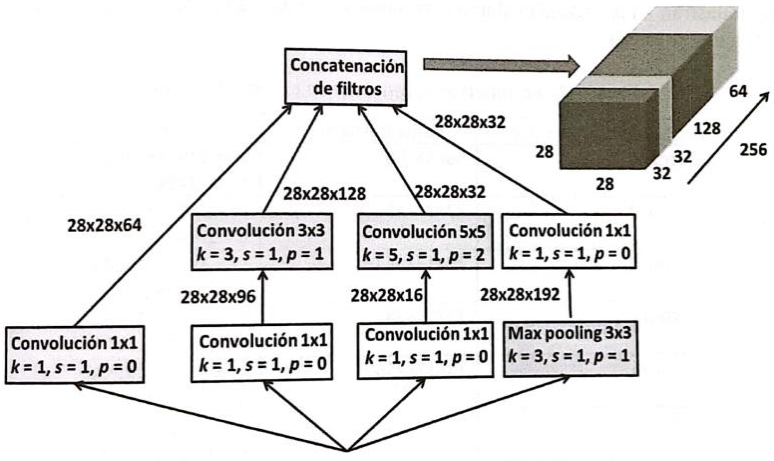


Figura 32[R]: Módulo Inception con reducción de la dimensión para la entrada de dimesiones 28x28x192

En términos de tiempos computacionales, considerando el número de operaciones a realizar en la convolución 5x5 para la entrada de 28x28, resulta ser un total de $5^2 x 192 x 32 x 28^2$, es decir, 120.442.400 operaciones.

Si se considera la propia convolución para es misma entrada, pero aplicando previamente una convolución 1x1 con 16 filtros, el resultado que se obtiene es:

- Para la convolución 1x1: $1^2 x 192 x 16 x 28^2$ = 2.408.448 operaciones
- Para la convolución 5x5: $5^2 x 16 x 32 x 28^2$ = 10.035.200 operaciones

La suma de ambas resulta ser 12.443.648 operaciones, que supone una reducción considerable en torno al 90 por ciento con respecto al valor obtenido con anterioridad.

REFERENCIAS

Información extraída de…/ Para profundizar más…/ Para más información consultar …

[1] Bravo, E. F. C., & Sotelo, J. A. L. (2009). *Una aproximación práctica a las redes neuronales artificiales*. Alianza Editorial.

[2] Martinsanz, G. P., Caro, P. J. H., & Portas, E. B. (2021). *Aprendizaje profundo*. RC Libros.

[3] Glejzer, C., Cicarrelli, A., Maldonado, A., Chomnalez, M., Bulit, F., & Facchinetti, C. (2012). *Las bases biológicas del aprendizaje*. Editorial de la Facultad de Filosofía y Letras UBA. Página 294.

[4] Mcculloch, W., & Pitts, W. (1943). *A logical calculus of the ideas immanent in nervous activity*. https://bit.ly/3dBVkMQ

[5] Ramírez, F. (2018). *Historia de la IA: Frank Rosenblatt y el Mark I Perceptrón, el primer ordenador fabricado específicamente para crear redes neuronales en 1957*. LUCA https://bit.ly/3IHiK1Y

[6] *Adaline Entrenamiento Supervisado*. (2011). Redes de Neuronas Artificiales. https://bit.ly/3lOF9Am

[7] Varona Martínez, P. (1997). *Memoria de Tesis Doctoral .Introducción*. Universidad Autónoma de Madrid. Escuela Técnica Superior de Informática.

[8] Backpropagation through time: what it does and how to do it. (1990, 1 octubre). *IEEE Journals & Magazine | IEEE Xplore*. https://bit.ly/31Dmeli

[9] Fukushima, K. (2007, 19 enero). *Neocognitron*. Scholarpedia. http://scholarpedia.org/article/Neocognitron

[10] Hochreiter, S. (1997, December). *LONG SHORT-TERM MEMORY*. ResearchGate. https://bit.ly/3rRl3cC

[11] Codificando Bits. (2019, 30 marzo). *La Convolución en las Redes Convolucionales* [Vídeo]. YouTube. https://bit.ly/3rVvnjI

[12] *Cross-Correlation*. Wolfram MathWorld. 2021, https://mathworld.wolfram.com/Cross-Correlation.html

[13] HUBEL, D. H., & WIESEL, T. N. (1959, abril). *RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX*. https://bit.ly/3y7LNXw

[14] Boureau, Y., Ponce, J., & LeCun, Y. (2014). *A Theoretical Analysis of Feature Pooling in Visual Recognition*. ResearchGate. https://bit.ly/3y6UeCp

[15] Ringach, D., & Shapley, R. (2003, 11 noviembre). *Reverse correlation in neurophysiology*. ResearchGate. https://bit.ly/3GA4ypc

[16] Niell, C. M. (2008, 23 julio). *Highly Selective Receptive Fields in Mouse Visual Cortex*. Journal of Neuroscience. https://www.jneurosci.org/content/28/30/7520.long

[17] Zoumpourlis, G., Doumanoglou, A., Vretos, N., & Daras, P. (2017). *Non-linear Convolution Filters for CNN-based Learning*. Cornell University. https://arxiv.org/pdf/1708.07038.pdf

[18] Kumar, R., Banerjee, A., & Vemuri, B. C. (2009, 1 junio). *Volterrafaces: Discriminant analysis using Volterra kernels*. IEEE Conference Publication | IEEE Xplore. https://bit.ly/3LvPo7Q

[19] Pajares Martinsanz, G. (2017). *Análisis y reconocimiento de voz*. RC Libros.

[20] Qian, N. (1998, 6 agosto). *On the momentum term in gradient descent learning algorithms*. Columbia University. https://bit.ly/3LR1CbB

[21] DAUGMAN, J. G. (1988, julio). *Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression*. IEEE TRANSACTIONS ON ACOUSTICS. https://bit.ly/3HbiAOk

[22] Kiranyaz, S., Ince, T., & Gabbouj, M. (2016, marzo). *Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks*. Scopus.com. https://bit.ly/3zWlLt5

[23] Kiranyaz, S., Ince, T., Losifidis, A., & Gabbouj, M. (2020, 6 marzo). *Operational neural networks*. SpringerLink. https://bit.ly/3MY0cfS

[24] Manilo, L. A., & Nemirko, A. P. (2016). *Recognition of biomedical signals based on their spectral description data analysis* (Vol. 26). Pattern Recognition and Image Analysis.

[25] Valdez, A. (2018, 7 marzo). *El Neocognitrón*. ACADEMIA. https://www.academia.edu/36104965/El_Neocognitr%C3%B3n

[26] Tuo, S., Chen, T., He, H., Feng, Z., Zhu, Y., Liu, F., & Li, C. (2021, 19 noviembre). *A Regional Industrial Economic Forecasting Model Based on a Deep Convolutional Neural Network and Big Data*. MDPI. https://bit.ly/3OCddvD

[27] Baxter, G., & Srisaeng, P. (2017, 20 octubre). *THE USE OF AN ARTIFICIAL NEURAL NETWORK TO PREDICT AUSTRALIA'S EXPORT AIR CARGO DEMAND*. School of Tourism and Hospitality Management, Suan Dusit University, Huahin Campus, Huahin, Prachaup Khiri Khan, Thailand. https://bit.ly/3OfSyO8

[28] Kurokawa, T., & Takeshita, K. (2004, 1 septiembre). *Air transportation planning using neural networks as an example of the transportation squadron in the Japan Air Self-Defense Force*. Wiley Online Library. https://bit.ly/3zXBQPc

BIBLIOGRAFÍA

Martinsanz, G. P., Caro, P. J. H., & Portas, E. B. (2021). *Aprendizaje profundo*. RC Libros.

Bravo, E. F. C., & Sotelo, J. A. L. (2009). *Una aproximación práctica a las redes neuronales artificiales*. Alianza Editorial.

Shih-kun, L. (s. f.). *Cómo entender la convolución dilatada - programador clic*. Programador Clic. https://programmerclick.com/article/94021688955/

Verma, S. (2021, 11 diciembre). *Understanding 1D and 3D Convolution Neural Network | Keras*. Medium. https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610

1D convolutional neural networks and applications: A survey. (2021, 1 abril). ScienceDirect. https://www.sciencedirect.com/science/article/pii/S0888327020307846

Ioannou, Y. (2017, 10 agosto). *A Tutorial on Filter Groups (Grouped Convolution)*. A Shallow Blog about Deep Learning. https://blog.yani.ai/filter-group-tutorial/

Pandey, A. (2018, 9 septiembre). *Depth-wise Convolution and Depth-wise Separable Convolution*. Medium. https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Colaboradores de Wikipedia. (2021, 2 diciembre). *Máquinas de vectores de soporte*. Wikipedia, la enciclopedia libre. https://bit.ly/3Bv3S3B

Raj, B. (2020, 31 julio). *A Simple Guide to the Versions of the Inception Network*. Medium. https://bit.ly/3HtNXUq

B. (2020, 13 agosto). *How ReLU and Dropout Layers Work in CNNs*. Baeldung on Computer Science. https://www.baeldung.com/cs/ml-relu-dropout-layers

Alake, R. (2021, 25 diciembre). *Deep Learning: GoogLeNet Explained - Towards Data Science*. Medium. https://bit.ly/3LjBsgv

WEBGRAFÍA

[A] https://bit.ly/39M9cGj Consultado en abril 2022

[B] https://bit.ly/3zWE9lN Consultado en diciembre 2021

[C] https://bit.ly/2YXFUsK Consultado en diciembre 2021

[D] https://bit.ly/3yfutkX Consultado en diciembre 2021

[E] https://bit.ly/39MTTNz Consultado en enero 2022

[F] https://bit.ly/2JFvzLN Consultado en enero 2022

[G] https://bit.ly/3ypC9kZ Consultado en enero 2022

[H] https://bit.ly/3xQl1mH Consultado en enero 2022

[I] https://bit.ly/3zWlLt5 Consultado en febrero 2022

[J] Martinsanz, G. P., Caro, P. J. H., & Portas, E. B. (2021). *Aprendizaje profundo*. p.116

[K] https://bit.ly/3ODyzsE Consultado en marzo 2022

[L] https://bit.ly/3ycA5MX Consultado en marzo 2022

[M] Martinsanz, G. P., Caro, P. J. H., & Portas, E. B. (2021). *Aprendizaje profundo*. p.122

[N] https://bit.ly/3OuV6ba Consultado en marzo 2022

[O] https://bit.ly/3ygpUaf Consultado en marzo 2022

[P] Martinsanz, G. P., Caro, P. J. H., & Portas, E. B. (2021). *Aprendizaje profundo*. p.150

[Q] Martinsanz, G. P., Caro, P. J. H., & Portas, E. B. (2021). *Aprendizaje profundo*. p.153

[R] Martinsanz, G. P., Caro, P. J. H., & Portas, E. B. (2021). *Aprendizaje profundo*. p.154

[S] https://bit.ly/3OB6fHo Consultado en diciembre 2021

[T] Martinsanz, G. P., Caro, P. J. H., & Portas, E. B. (2021). *Aprendizaje profundo*. p.145

[U] https://bit.ly/3tYRlTd Consultado en abril 2022

[V] Martinsanz, G. P., Caro, P. J. H., & Portas, E. B. (2021). *Aprendizaje profundo*. p.175

[W] Imagen adaptada de: Martinsanz, G. P., Caro, P. J. H., & Portas, E. B. (2021). *Aprendizaje profundo*. p.175

[X] Martinsanz, G. P., Caro, P. J. H., & Portas, E. B. (2021). *Aprendizaje profundo*. p.176

[Y] https://bit.ly/39RDCXt Consultado en mayo 2022

[Z] Figuras de creación propia

APÉNDICES

1. BASE NEUROCIENTÍFICA DE LAS CNN

Las redes neuronales convolucionales son un exponente claro en la historia de la inteligencia artificial con inspiración biológica en el ámbito de la neurociencia. Su historia comienza con varios experimentos neurocientíficos como paso previo a los modelos computacionales inspirados en el modelo de Hubel y Wiesel [13] (1959). Su trabajo fue importante para la comprensión del funcionamiento la corteza visual, en concreto, las células responsables de la selección de orientación y detección de bordes en los estímulos visuales dentro de la corteza visual primaria.

El cerebro posee una parte denominada corteza visual primaria *o V1(primary visual cortex),* que es la primera área del cerebro, localizada en la parte posterior de la cabeza, que comienza a realizar un procesamiento avanzado de la entrada visual. Una capa de una CNN se diseña para capturar 3 propiedades de la V1:

- V1 se organiza en forma de mapa espacial. Tiene estructura bidimensional, esto quiere decir que recoge en forma simétrica la estructura de la imagen en la retina. Por ejemplo, la luz que llega a la mitad inferior de la retina afecta solo a la correspondiente mitad de V1. Las CNN recogen esta propiedad definiendo las características en términos de mapas bidimensionales.
- V1 contiene muchas células simples. La actividad de una célula simple puede definirse como una función lineal de la imagen en un pequeño y espacialmente localizado campo receptivo. Las unidades del detector en las CNN se diseñan para simular esas propiedades de las células simples.
- V1 posee también células complejas. Éstas presentan características similares a las células simples, pero con la particularidad de que son invariantes a pequeños cambios en la posición de las características. Este hecho inspira las unidades de *pooling (agrupamiento)* de las CNN.

Las CNN (Convolutional neural network) tienen su origen en el Neocognitrón[25], una red neuronal creada para el reconocimiento de caracteres escritos a mano, basados en el estudio de la corteza visual del ojo humano.

El diseño del Neocognitrón adopta esta estructura jerárquica en una arquitectura por capas:

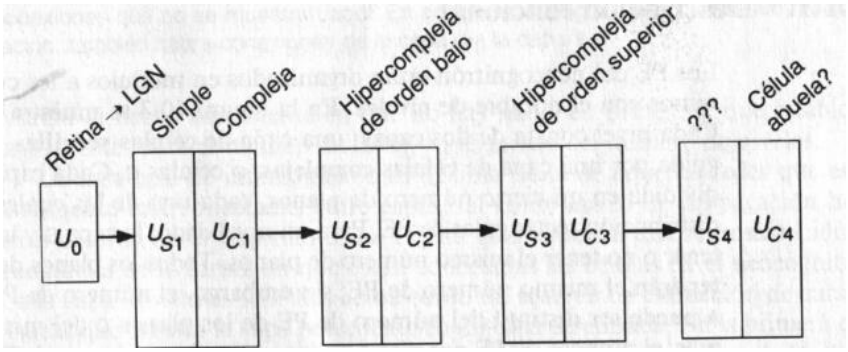


Figura 33[S]: Arquitectura por capas del Neocognitrón

En esta figura se muestra la estructura jerárquica del Neocognitrón. Cada caja representa un nivel formado por una capa de células simples U_{Si}, y una capa de células complejas, en donde *i* es el número de capa. U_0 representa señales que se originan en la retina. La red termina con células individuales que corresponden a estímulos visuales complejos. Esas células finales suelen denominarse células abuelas (*grandmother cells*), debido a la idea de que en nuestro cerebro haya células que responden a estímulos visuales complejos, tales como una imagen de nuestra abuela. De modo que independientemente de sí aparece al lado izquierdo o al derecho de la imagen, si es un primer plano del rostro o una foto alejada del todo el cuerpo, si está iluminada o en sombra, una neurona se activaría.

No obstante, existen diferencias significativas entre las CNN y el sistema de visión de los seres humanos:

- El ojo humano es principalmente de baja resolución. El hecho de que a pesar de ello se reciban amplias escenas con alta resolución es debido a la creación de ilusiones por parte del subconsciente del cerebro. Actualmente, las CNN reciben imágenes de alta resolución, pero la incorporación de mecanismos de aprendizaje constituye aún un reto como línea de investigación.

- El sistema de visión del ser humano está integrado con otros muchos sentidos, como la audición o el estado de ánimo y los pensamientos. Las CNN son relativas a procesamiento de señales temporales, como es el audio o el video.

- El sistema visual humano reconoce mucho más que objetos, es capaz de comprender escenas enteras, incluyendo objetos diversos y sus relaciones. Las CNN se han aplicado para resolver algunos de esos problemas, pero se encuentran aún en niveles prematuros.

- Algunas áreas del cerebro, como la V1, están afectadas por la retroalimentación de niveles superiores. En los modelos de redes neuronales, esta retroalimentación se ha explorado, pero no se han conseguido todavía mejoras convincentes.

Hasta ahora se ha descrito cómo las células simples son más o menos lineales y selectivas para ciertas características y las células complejas, por otra parte, son más bien no lineales y se transforman en invariantes. Pero falta especificar qué detectan estas células individuales. En una red neuronal artificial, las celdas simples en la primera capa son las más fáciles de analizar, porque sus respuestas están dirigidas por una función lineal. Mostrando simplemente una imagen del núcleo de convolución se puede ver a qué responde el canal correspondiente de la capa convolucional.

En cambio, en una red neuronal biológica no se tiene acceso a los pesos en sí mismos, en lugar se coloca un electrodo en la neurona. Así que se puede ajustar un modelo lineal a estas respuestas para obtener de manera aproximada los pesos de la neurona. Este enfoque se conoce como correlación inversa (Ringach y Shapley, 2004)[15].

Los experimentos realizados respecto a la correlación inversa muestran que la mayoría de células V1 poseen pesos que pueden describirse mediante lo que se conoce como funciones de Gabor[21] definidas mediante la ecuación:

$$G_{\alpha,\lambda,\theta,\sigma,\gamma,X_0,Y_0}(x,y) = \alpha \exp\left(-\frac{x_d^2 + \gamma^2 y_d^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x_d}{\lambda} + \varphi\right)$$

Donde,

$$x_d = (x - x_0)\cos(\vartheta) + (y - y_0)\sin(\theta)$$

$$y_d = -(x - x_0)\sin(\vartheta) + (y - y_0)\cos(\theta)$$

Los parámetros $\alpha, \lambda, \theta, \sigma, \gamma, x_0, y_0$ son los que controlan las propiedades de la función Gabor.

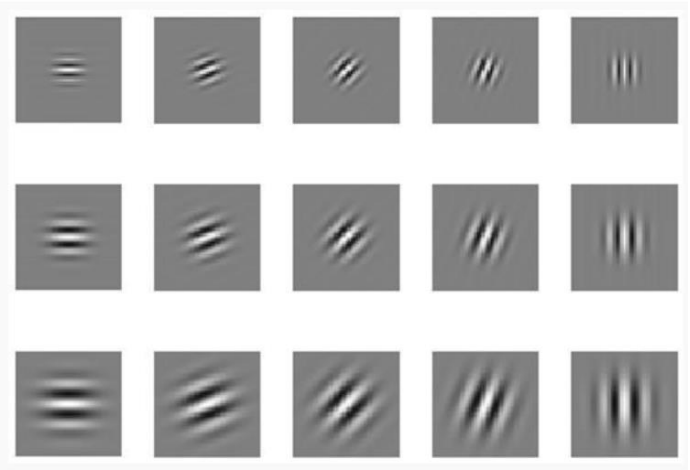


Figura 34[T]: Filtro de Gabor para la extracción de características de textura de la imagen de destino

- x_0, y_0, θ definen la posición espacial (x, y), de forma que el centro de la máscara del filtro se traslada (x_0, y_0) posiciones y rota un ángulo θ con respecto del eje horizontal. Esto se aprecia comparando la primera y última imagen de cada fila, pasa de 0 a 180º.
- α define el rango de valores del filtro.
- La desviación estándar σ del factor gaussiano determina la dimensión del campo receptivo. La dimensión en forma elíptica de dicho campo está gobernada por γ que se denota como razón de aspecto.
- El parámetro λ define la longitud de onda y $1/\lambda$ la frecuencia espacial del coseno en la fórmula anterior. Observando el grosor de la onda, la primera fila presenta una longitud de onda mucho menor que la última, por ejemplo.
- La razón σ/λ determina el ancho de banda de la frecuencia espacial.

2. GOOGLENET. APLICACIONES

GoogleNet es una red neuronal convolucional profunda desarrollada por investigadores de Google compuesta de 22 capas, de la cuales 9 son de tipo inception, por lo que también se la conoce como Inception V1.

La arquitectura de GoogLeNet fue presentada en ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14) donde obtuvo una tasa de error del 6.67%, consiguiendo un desempeño muy cercano al nivel humano.

Como método de optimización utiliza la normalización por lotes (*batch*), distorsiones de imagen y RMSProp (Root Mean Square Propagation), que emplea el concepto de "ventana" para considerar solo los gradientes más recientes en lugar de acumularlos.

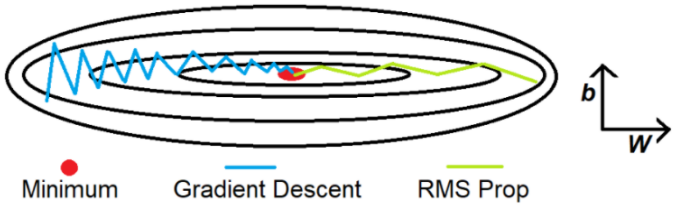
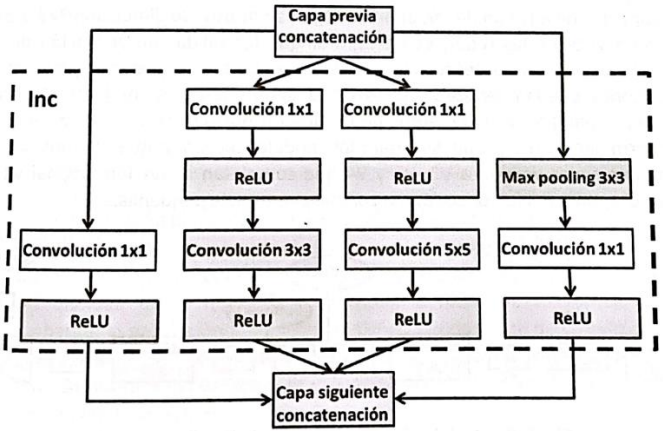


Figura 35[U]: Propagación RMS

La Propagación RMS de raíz cuadrática es una técnica para amortiguar el movimiento en el eje *Y* acelerando así el descenso del gradiente. Para una mejor comprensión, denotemos el eje Y como el sesgo b y el eje X como el peso W. Se llama raíz cuadrática media porque elevamos al cuadrado las derivadas de los parámetros w y b.

Cada una de las 9 capas inception contiene la estructura que se muestran a continuación:



- Convolución
- Funciones de activación ReLU
- Max pooling

Figura 36[V]: Módulo Inception con incorporación de unidades ReLU

El modelo completo es el siguiente:

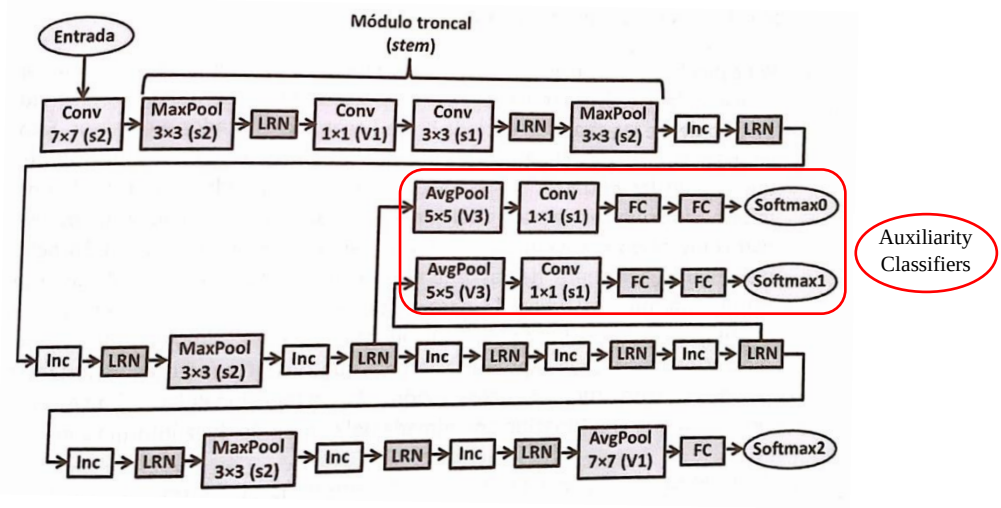


Figura 37[W]: Modelo completo GoogleNet

- Convolución: indicando las dimensiones de los filtros.
- Pooling: tanto average (AvgPool) o máximo (Maxpool) con indicación del tamaño de la ventana.
- Normalización: Local Response Normalization (LRN)
- Capas totalmente conectadas (FC)
- Módulos Inception (Inc): En las V1 constituyen la base de este tipo de redes.

Este último esquema presenta dos redes extra que se conocen como clasificadores auxiliares. Presentan un AvgPool de dimensión 5x5 y s = 3, que proporciona salidas de tamaños 4x4x512 y 4x4x528 respectivamente. A continuación, hay una capa de convolución 1x1 para reducir la dimensionalidad y una unidad ReLU. Le sigue una unidad dropout con capas totalmente conectadas finalizando con unidades softmax. Su objetivo es favorecer a las capas más bajas, aumentar la señal del gradiente que se propaga hacia atrás combatiendo a su vez el problema del desvanecimiento del gradiente y proporcionar una regularización adicional. El esquema del modelo complejo arriba descrito queda simplificado del siguiente modo:

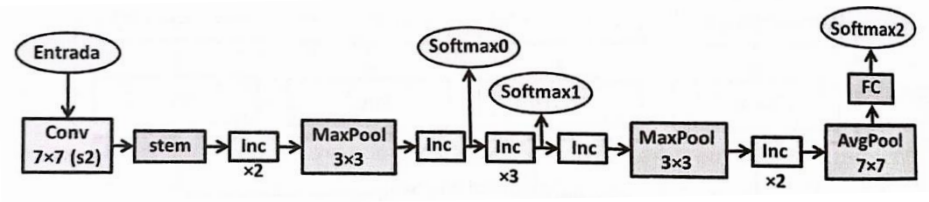


Figura 38[X]: Modelo simplificado GoogleNet

Donde el bloque stem corresponde a todos los elementos indicados arriba y los clasificadores auxiliares salen en forma de flechas que finalizan en las funciones Softmax.

En términos de regularización, se sostiene que el clasificador principal funciona mejor si en las ramas laterales se aplica la normalización por bloques (*batch normalization)* o dropout. Facilitan un aprendizaje más estable y una mejor convergencia.

Una ventaja de este tipo de redes es la reducción de la dimensión, que desde el punto de vista computacional se considera un caso especial de la factorización convolucional. Por ejemplo, considerando una capa de convolución 1x1 seguida de otra de dimensión 3x3, desde el punto de vista de las redes, es de esperar que las salidas de activación de las capas cercanas estén correlacionadas. Luego se podría asumir que sus activaciones pueden reducirse antes de la agregación para mejorar la eficacia computacional. Esta propuesta es la que ha permitido que la versión original V1, se transforme en las denominadas versiones V2, V3 y V4. Para las que se propone una factorización en convoluciones más pequeñas.

APLICACIONES. CLASIFICACIÓN DE IMÁGENES

Como ya hemos visto, GoogleNet es una red neuronal convolucional de 22 capas. Esta red entrenada puede clasificar imágenes hasta en 1000 categorías de objetos, como, por ejemplo, teclado, ratón, lápiz, limón, cesta, escenario y una gran variedad de animales, entre ellos, una cantidad considerable de razas de perros y gatos.

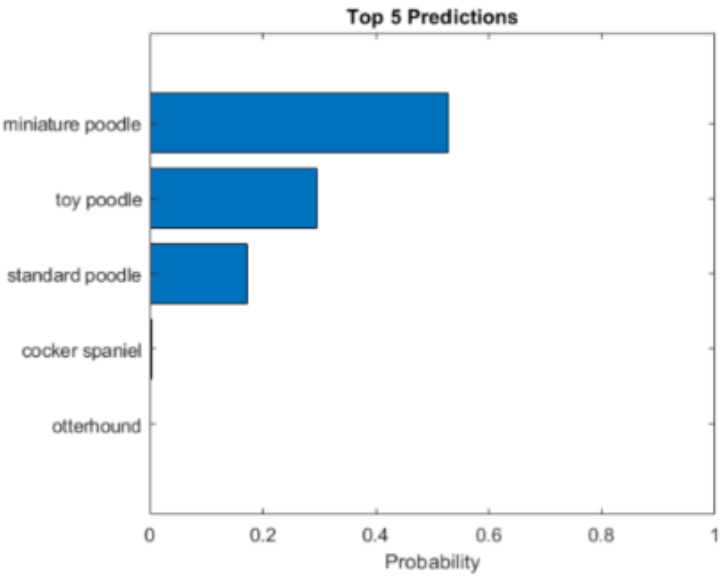
Para este ejemplo he seleccionado un par de imágenes para que la red las clasifique.

En primer lugar, tenemos la imagen de un caniche, la clasificación de la red es: ‘miniature poodle’, con una precisión de casi el 53%. En efecto la red logra acertar lo que aparece en la imagen, aunque llama la atención que la precisión sea tan solo del 52.9%.

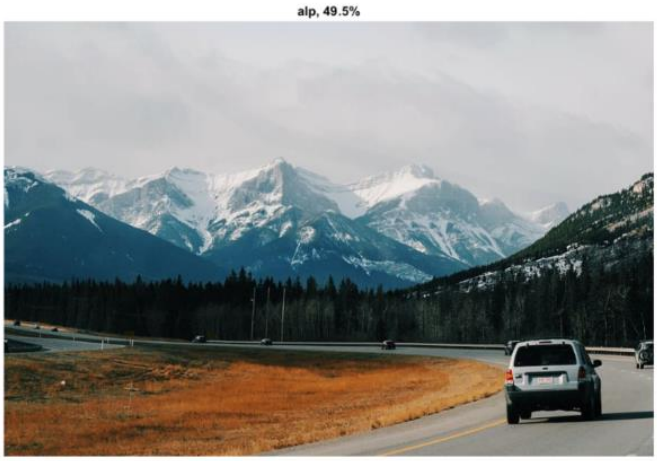
Sin embargo, si visualizamos el Top 5 predicciones observamos que:

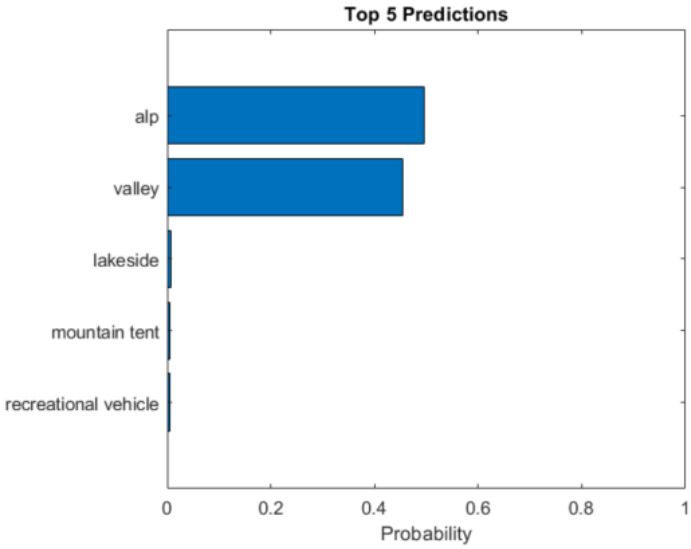
- La segunda clasificación es ‘toy poodle’
- La tercera es `stardar poodle’

Luego realmente esta precisión que aparece no es tan baja si tenemos en cuenta que las 3 primeras opciones de clasificación ya cuentan con que pertenece a la raza poodle.



Veamos otro ejemplo de clasificación:





El objetivo al escoger esta imagen es comprobar cómo reacciona la red cuando existen varias clasificaciones posibles. Efectivamente, lo más característico son las montañas del fondo, que es la clasificación que ha hecho la red con una precisión del 49.5%, sin embargo, observando el Top 5 predicciones la segunda es ’valey’ y en quinto lugar aparece ‘recreational vehicle’. Es decir, GoogleNet ha acertado todos los elementos que componen la fotografía.

3. RECONOCIMIENTO DE ARRITMIAS PELIGROSAS. CNN 1D Y 2D

INTRODUCCIÓN

Una arritmia es un trastorno de la frecuencia cardiaca que provoca que el corazón pueda latir más rápido (taquicardia), demasiado lento (bradicardia) o de manera irregular.

La detección de arritmias potencialmente mortales es uno de los desafíos en el control de enfermedades cardiovasculares. La prueba más sencilla y eficaz para diagnosticar este trastorno es el electrocardiograma (ECG), que se define como la representación visual de la actividad eléctrica del corazón en función del tiempo. Se obtiene colocando unos electrodos pegados en la piel del paciente.

Existen gran variedad de enfoques para la detección automática de arritmias peligrosas, tanto en métodos de reconocimiento utilizados, como en la ventana de tiempo usada para su detección.

Para esta parte del trabajo abordaremos la clasificación de arritmias peligrosas utilizando en primer lugar una red neural convolucional 1D y a continuación se tratará el mismo problema con una CNN 2D.

ELECTROCARDIOGRAMA (ECG)

El trazado típico de un electrocardiograma registrando un latido cardíaco normal consta de los siguientes elementos:

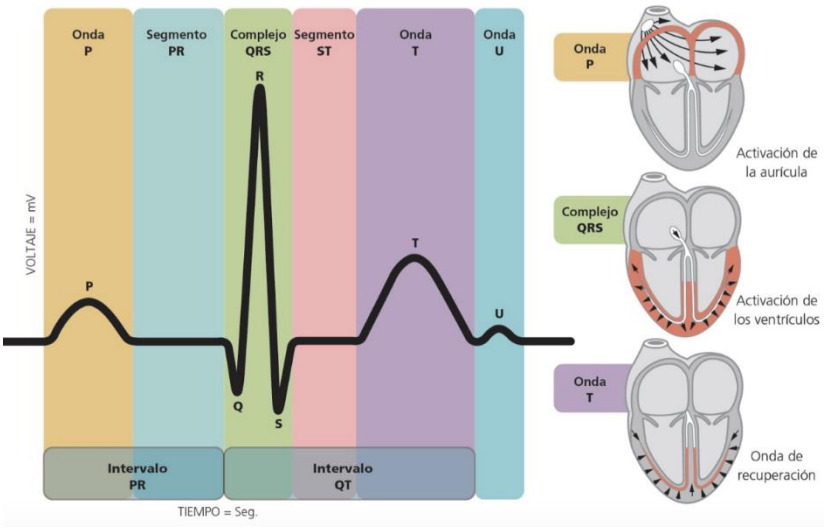


Figura 39[Y]: Componentes de un electrocardiograma

- Una onda P, que es la señal eléctrica que corresponde a la despolarización auricular. Esto significa que se tiene una carga más positiva que el entorno y por tanto activa la aurícula. La despolarización se inicia en el endocardio, es decir, el estímulo eléctrico atraviesa el espesor de la pared cardiaca desde el endocardio al epicardio.
- El complejo QRS corresponde a la corriente eléctrica que causa la contracción de los ventrículos derecho e izquierdo (despolarización ventricular), la cual es mucho más potente que la de las aurículas y incluye a más masa muscular, produciendo de este modo una mayor deflexión (cabio de dirección) en el electrocardiograma. La duración normal es de 60 a 120 milisegundos. Cuando aparece completo, el complejo QRS consta de 3 vectores:
  - Onda Q: Onda negativa, es la más grande las tres.
  - Onda R: Es la primera deflexión positiva del complejo QRS y es la de mayor tamaño.
  - Onda S: Cualquier onda negativa que siga a la onda R.
- La onda T representa la repolarización de los ventrículos. Eléctricamente, las células del músculo cardíaco son como muelles cargados; un pequeño impulso las dispara, despolarizan y se contraen. La recarga del muelle es la repolarización. En la mayoría de las cosas, la onda T es positiva. Si esto no es así puede que sea síntoma de una enfermedad.
- La pequeña onda U normalmente es invisible.

DESCRIPCIÓN DE LOS DATOS

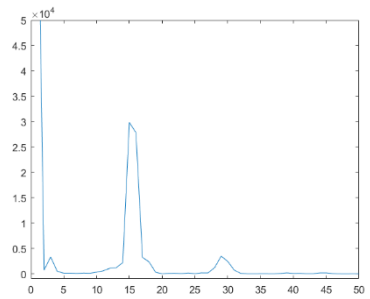
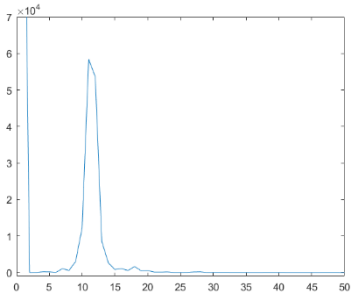
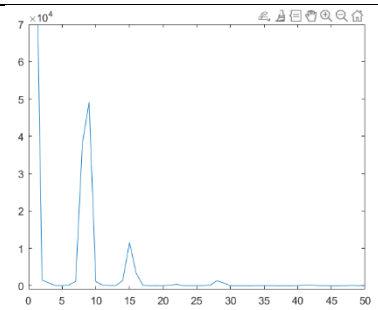
La base de datos utilizada proviene de la página https://physionet.org/ , una web con datos de investigaciones médicas de libre acceso creada por el Laboratorio de Fisiología Computacional del MIT.

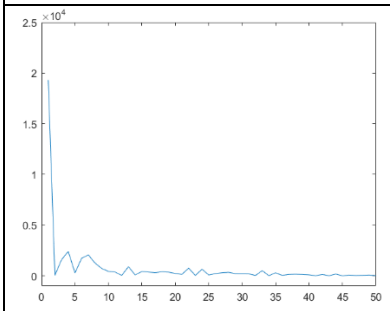
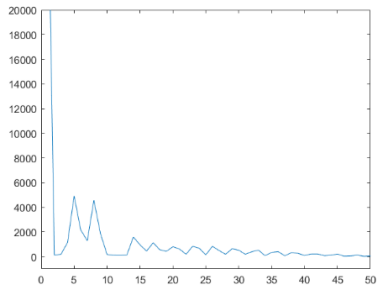
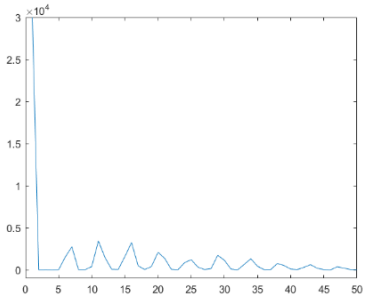
Contiene un conjunto de fragmentos de 2 segundos de señales de ECG con alteraciones del ritmo, que se agrupan en clases separadas según el grado de amenaza para la vida del paciente. Las clases de arritmia incluidas en el conjunto de datos de entrenamiento se presentan a continuación, en orden general decreciente de riesgo para el paciente:

1. Arritmias potencialmente mortales que requieren reanimación urgente:
   - VFL: aleteo ventricular.
   - FV: fibrilación ventricular.

2. Una forma especial de arritmias potencialmente mortales:
    - VTTdP: taquicardia ventricular torsade de pointes.
3. Arritmias ventriculares potencialmente mortales:
    - VTHR: taquicardia ventricular de alta frecuencia (monomorfa y polimorfa).
4. Arritmias ventriculares potencialmente peligrosas:
    - VTLR: taquicardia ventricular de baja frecuencia (monomorfa y polimorfa);
    - B: bigeminismo ventricular;
    - HGEA: alto grado de actividad ectópica ventricular;
    - VER: ritmo de escape ventricular.
5. Arritmias supraventriculares:
    - AFIB: fibrilación auricular;
    - SVTA: taquicardia supraventricular;
    - SBR: bradicardia sinusal;
    - IB: bloqueo cardíaco de primer grado;
    - NOD: ritmo nodal (a-v).
6. Ritmo sinusal:
    - BBB: ritmo sinusal con bloqueo de rama;
    - N: ritmo sinusal normal;
    - Ne: ritmo normal con extrasístole única.

A continuación, se muestra esta clasificación utilizando muestras aleatorias de la base de datos de los fragmentos de ECG de 2 s cada uno:

Arritmias potencialmente mortales	Una forma especial de arritmias potencialmente mortales	Arritmias ventriculares potencialmente mortales
		

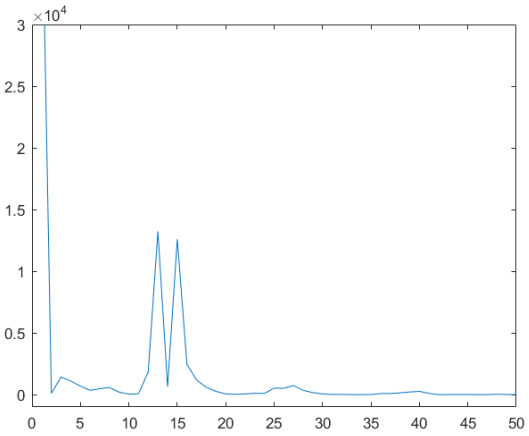
Arritmias ventriculares potencialmente peligrosas	Arritmias supraventriculares	Ritmo sinusal
		

La base de datos contiene además de los fragmentos de ECG de 2s, un espectro completo de un fragmento de ECG (0 - 180 Hz) con un paso de 0,5 Hz. El espectro se calcula para cada fragmento utilizando la transformada rápida de Fourier. El procedimiento de cálculo se describe en detalle en [24].

CLASIFICACIÓN DE ARRITMIAS USANDO CNN 1D

Para este enfoque del problema, se tiene un data set dividido en 6 capetas, cada una de ellas compuesta por subcarpetas con los datos del espectro completo de un fragmento de ECG, estos datos vienen dados en el siguiente formato:

```
A = 361×1
10^4 ×
    4.2463
    0.0107
    0.1432
    0.1106
    0.0692
    0.0352
    0.0481
    0.0590
    0.0205
    0.0044
      ⋮
```



Todos los archivos están compuestos por un vector de 361x1 con datos. Lo primero que se hace con ellos es asignarlos a una matriz de vectores, de este modo, cada celda tiene la información de un array 361x1. En paralelo, se crea un vector que contiene la agrupación de los espectros en las 6 clases de arritmia descritos anteriormente. Estos serán los grupos por clasificar.

Para ello, la matriz se descompone en conjunto de train y test, dejando el 80% de los datos para el entrenamiento y 20% para la clasificación. El siguiente paso es la implementación de la red y posteriormente su entrenamiento. Para implementar estos pasos, van a ser necesarias cuatro partes:

1. Los electrocardiogramas del entrenamiento.
2. Diagnósticos de cada ECG.
3. El conjunto de capas que compondrán la red.
4. Las opciones del entrenamiento.

La red creada con los recursos de Matlab utiliza las opciones de entrenamiento SGDM (*Stochastic Gradient Descent with momentum*). Este método ayuda a acelerar los vectores del gradiente en las direcciones correctas para llegar a una convergencia más rápida. Es uno de los algoritmos de optimización más populares y muchos modelos de última generación se entrenan con él. Algunas de las opciones más importantes de este método son:

InitialLearnRate: Si la tasa de aprendizaje es demasiado baja, el entrenamiento puede llevar mucho tiempo, si es demasiado alta, puede llegar a divergir.

GradientThresholdMethod: Se utiliza para recortar los valores que superan un cierto límite que es el umbral del gradiente (*Gradient Thereshold*), este umbral se puede calcular de varias formas:

- 'l2norm': si la norma L2 del gradiente de un parámetro es mayor que GradientThreshold, entonces se escala el gradiente para que la norma L2 sea igual a GradientThreshold.
- 'global-l2norm': si la norma L2 global, L, es mayor que GradientThreshold, entonces escala todos los gradientes por un factor de GradientThreshold/L
- 'absolute-value': si el valor absoluto de una derivada parcial individual en el gradiente de un parámetro que se puede aprender es mayor que GradientThreshold, entonces escala la derivada parcial para que tenga una magnitud igual a GradientThreshold y conserva el signo de la derivada parcial.

MaxEpoch: Número máximo de etapas que se usarán para el entrenamiento, especificado como un número entero positivo. Una iteración es un paso dado en el algoritmo de descenso de gradiente para minimizar la función de pérdida utilizando un mini lote. Una etapa es el paso completo del algoritmo de entrenamiento sobre todo el conjunto de entrenamiento.

SequencePaddingDirection: Puede ser una de las siguientes:

- "right" : Rellena o trunca secuencias a la derecha. Las secuencias comienzan en el mismo paso de tiempo y el software trunca o agrega relleno al final de las secuencias.
- "left" :Rellena o trunca secuencias a la izquierda. El software agrega relleno al inicio de las secuencias para que las secuencias terminen en el mismo paso de tiempo.

Respecto a la estructura de la red neuronal, está formada por 34 capas, una primera entrada con la dimensión de los datos de entrenamiento, que es de 361 y 6 lotes de 5 capas que se repiten a lo largo de la estructura. Cada lote lo forman:

- Una capa de convolución 1D: con 32 filtros de tamaño 17.
- Función ReLU: Realiza una operación de umbral para cada elemento de la entrada, donde cualquier valor menor que cero se establece en cero.
- Capa de normalización: Normaliza un mini lote de datos en todos los canales para cada observación de forma independiente. Para acelerar el entrenamiento, estas capas se usan después de las capas de aprendizaje.
- MaxPooling 1D: Realiza una reducción de muestreo dividiendo la entrada en regiones de agrupación 1-D y luego calcula el máximo de cada región.
- SpatialDropout: Estable aleatoriamente los elementos de entrada en cero con una probabilidad dada. En este caso, la probabilidad está en 0.005.

```
options =
  TrainingOptionsSGDM with properties:

                     Momentum: 0.9000
                 InitialLearnRate: 1.0000e-04
                LearnRateSchedule: 'piecewise'
              LearnRateDropFactor: 0.1000
              LearnRateDropPeriod: 10
                 L2Regularization: 1.0000e-04
           GradientThresholdMethod: 'l2norm'
                 GradientThreshold: Inf
                         MaxEpochs: 30
                     MiniBatchSize: 128
                           Verbose: 0
                   VerboseFrequency: 50
                     ValidationData: []
                ValidationFrequency: 50
                 ValidationPatience: Inf
                           Shuffle: 'every-epoch'
                    CheckpointPath: ''
              CheckpointFrequency: 1
            CheckpointFrequencyUnit: 'epoch'
               ExecutionEnvironment: 'auto'
                         WorkerLoad: []
                          OutputFcn: []
                              Plots: 'training-progress'
                   SequenceLength: 'longest'
              SequencePaddingValue: 0
           SequencePaddingDirection: 'right'
               DispatchInBackground: 0
            ResetInputNormalization: 1
      BatchNormalizationStatistics: 'moving'
                     OutputNetwork: 'last-iteration'
```

Figura 40[Z]: Opciones de entrenamiento de 1D

```
layers =
  34×1 Layer array with layers:

     1   ''   Sequence Input          Sequence input with 361 dimensions
     2   ''   Convolution             32 17 convolutions with stride 1 and padding 'causal'
     3   ''   ReLU                    ReLU
     4   ''   Layer Normalization     Layer normalization
     5   ''   1-D Max Pooling         Max pooling with pool size 1, stride 1, and padding [0  0]
     6   ''   Spatial Dropout         Spatial dropout with factor 0.005
     7   ''   Convolution             64 17 convolutions with stride 1 and padding 'causal'
     8   ''   Layer Normalization     Layer normalization
     9   ''   ReLU                    ReLU
    10   ''   1-D Max Pooling         Max pooling with pool size 1, stride 1, and padding [0  0]
    11   ''   Spatial Dropout         Spatial dropout with factor 0.005
    12   ''   Convolution             128 17 convolutions with stride 1 and padding 'causal'
    13   ''   Layer Normalization     Layer normalization
    14   ''   ReLU                    ReLU
    15   ''   1-D Max Pooling         Max pooling with pool size 1, stride 1, and padding [0  0]
    16   ''   Spatial Dropout         Spatial dropout with factor 0.005
    17   ''   Convolution             256 17 convolutions with stride 1 and padding 'causal'
    18   ''   Layer Normalization     Layer normalization
    19   ''   ReLU                    ReLU
    20   ''   1-D Max Pooling         Max pooling with pool size 1, stride 1, and padding [0  0]
    21   ''   Spatial Dropout         Spatial dropout with factor 0.005
    22   ''   Convolution             512 17 convolutions with stride 1 and padding 'causal'
    23   ''   Layer Normalization     Layer normalization
    24   ''   ReLU                    ReLU
    25   ''   1-D Max Pooling         Max pooling with pool size 1, stride 1, and padding [0  0]
    26   ''   Spatial Dropout         Spatial dropout with factor 0.005
    27   ''   Convolution             1024 13 convolutions with stride 1 and padding 'causal'
    28   ''   Layer Normalization     Layer normalization
    29   ''   ReLU                    ReLU
    30   ''   1-D Max Pooling         Max pooling with pool size 1, stride 1, and padding [0  0]
    31   ''   Spatial Dropout         Spatial dropout with factor 0.005
    32   ''   Fully Connected         6 fully connected layer
    33   ''   Softmax                 softmax
    34   ''   Classification Output   crossentropyex
```

Figura 41[Z]: Capas que forman la red CNN 1D

RESULTADOS

Tras implementar la red con las características brevemente descritas, los resultados obtenidos se muestran en la siguiente matriz de confusión:

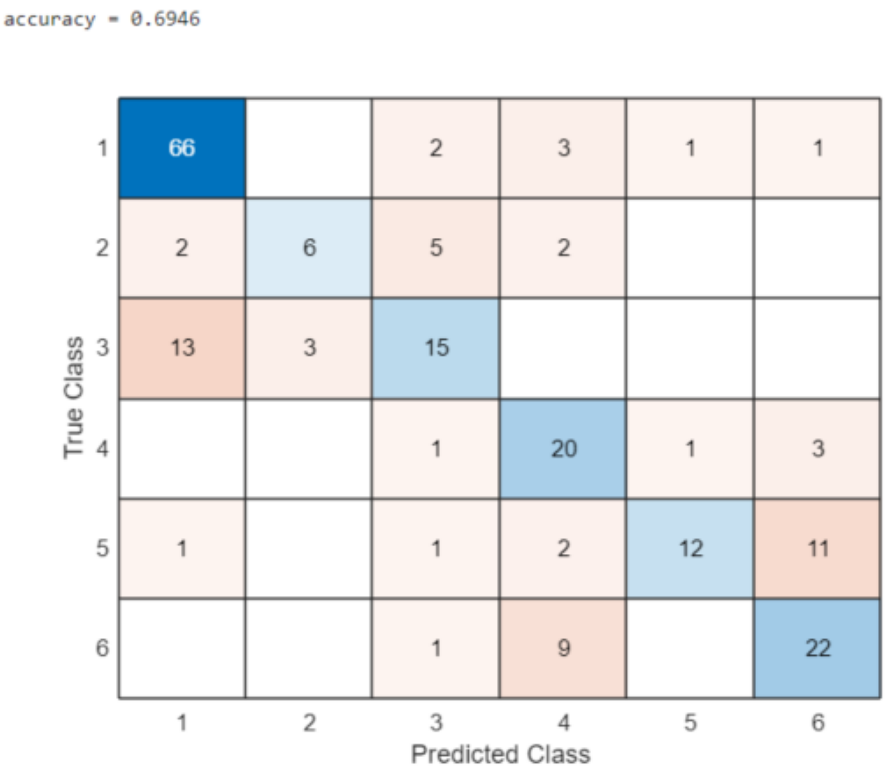


Figura 42[Z]: Matriz de confusión de la red CNN 1D

Cada columna de la matriz representa el número de predicciones de cada clase, esto es las clases que la red ha predicho en el entrenamiento. Cada fila representa las clases reales, es decir, la muestra del 20% de los fragmentos de ECG. Observamos que con una precisión casi del 70%, la clasificación es bastante aceptable. Los valores que aparecen del 1 al 6 se corresponden con la clasificación de las arritmias, siendo:

1. Arritmias potencialmente mortales
2. Una forma especial de arritmias potencialmente mortales
3. Arritmias ventriculares potencialmente mortales
4. Arritmias ventriculares potencialmente peligrosas
5. Arritmias supraventriculares
6. Ritmo sinusal

La cantidad de muestra de cada clase no es equitativa, por lo que no es de sorprender que la red haya sido capaz de predecir una cantidad considerablemente mayor de arritmias del tipo 1 que del 2, siendo la distribución de cada una de las clases la siguiente:

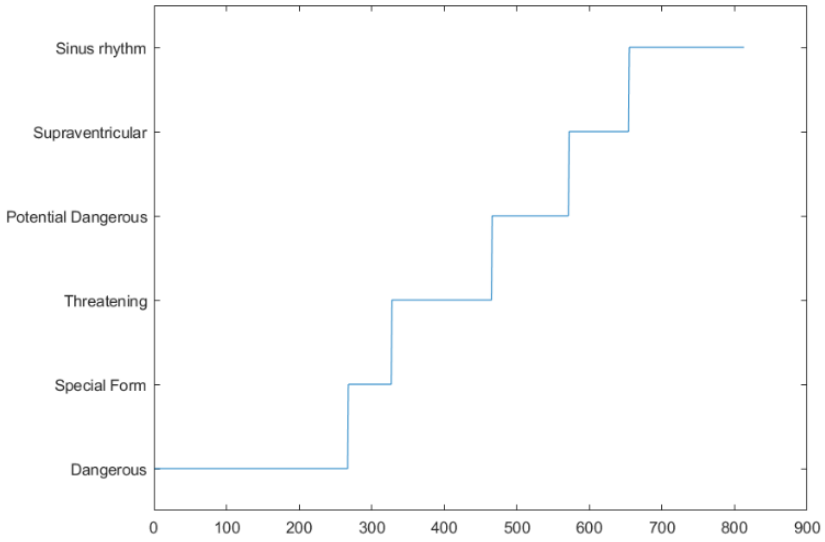


Figura 43[Z]: Distribución de las clases de arritmia en el data set del entrenamiento de la red

CLASIFICACIÓN DE ARRITMIAS USANDO CNN 2D

Para este enfoque del problema se ha optado por usar diagramas que se generan con los espectros de los electrocardiogramas. De este modo se crean imágenes y se almacenan como archivos .png, con lo que se les puede aplicar una red neuronal convolucional en 2 dimensiones.

Matlab tiene muchas librerías para trabajar con imágenes, así como paquetes como Deep Learning Toolbox, que hace más sencillo la implementación de las redes neuronales convolucionales, tanto en 1 como en 2 dimensiones.

En primer lugar, una vez almacenadas todas las imágenes, se procede a segmentar el data set en train y test. De nuevo para este método se ha optado por una división del 80/20. Para el entrenamiento de la red neuronal se necesitan 3 partes:

- Gráficos del ECG.
- Opciones de entrenamiento
- Capas que componen a la red

Las opciones de entrenamiento son las mismas que para la red en 1 dimensión, cambiando alguno de los parámetros, como por ejemplo LearnRateSchedule, que es la opción para reducir la tasa de aprendizaje durante el entrenamiento.

La red creada se compone de 5 bloques compuestos por 4 capas cada uno, estas capas son:

- Convolution2dLayer: Es una capa convolucional bidimensional que aplica filtros convolucionales en este caso a las imágenes, que son la entrada de la red. La capa funciona moviendo los filtros a lo largo de la entrada vertical y horizontalmente y calculando el producto de puntos de los pesos y la entrada, y añadiendo después un término de sesgo.
- BatchNormalizationLayer: Normaliza un minilote de datos en todas las observaciones para cada canal de forma independiente. Para acelerar el entrenamiento de la red neuronal convolucional y reducir la sensibilidad a la inicialización de la red, se utiliza entre las capas convolucionales y las no lineales, como las capas ReLU.
- ReluLayer: Realiza una operación de umbral a cada imagen de la entrada, donde cualquier valor menor que cero se pone a cero.
- MaxPooling2dLayer: Es una capa de agrupación máxima bidimensional, realiza un muestreo descendente dividiendo la entrada en regiones de agrupación rectangulares y, a continuación, calculando el máximo de cada región.

```
options =
  TrainingOptionsSGDM with properties:

                     Momentum: 0.9000
                 InitialLearnRate: 1.0000e-04
                LearnRateSchedule: 'none'
              LearnRateDropFactor: 0.1000
              LearnRateDropPeriod: 10
                 L2Regularization: 1.0000e-04
        GradientThresholdMethod: 'l2norm'
              GradientThreshold: Inf
                      MaxEpochs: 30
                  MiniBatchSize: 128
                        Verbose: 0
                VerboseFrequency: 50
                  ValidationData: []
             ValidationFrequency: 50
              ValidationPatience: Inf
                        Shuffle: 'every-epoch'
                 CheckpointPath: ''
             CheckpointFrequency: 1
        CheckpointFrequencyUnit: 'epoch'
           ExecutionEnvironment: 'auto'
                     WorkerLoad: []
                      OutputFcn: []
                          Plots: 'training-progress'
                 SequenceLength: 'longest'
            SequencePaddingValue: 0
       SequencePaddingDirection: 'right'
           DispatchInBackground: 0
         ResetInputNormalization: 1
   BatchNormalizationStatistics: 'population'
                  OutputNetwork: 'last-iteration'
```

Figura 44[Z]: Opciones de entrenamiento de la red CNN 2D

```
Inputlayers =
  23×1 Layer array with layers:

     1   ''   Image Input             224×224×3 images with 'zerocenter' normalization
     2   ''   Convolution             16 5×5 convolutions with stride [1  1] and padding 'same'
     3   ''   Batch Normalization     Batch normalization
     4   ''   ReLU                    ReLU
     5   ''   Max Pooling             2×2 max pooling with stride [2  2] and padding [0  0  0  0]
     6   ''   Convolution             32 5×5 convolutions with stride [1  1] and padding 'same'
     7   ''   Batch Normalization     Batch normalization
     8   ''   ReLU                    ReLU
     9   ''   Max Pooling             2×2 max pooling with stride [2  2] and padding [0  0  0  0]
    10   ''   Convolution             64 5×5 convolutions with stride [1  1] and padding 'same'
    11   ''   Batch Normalization     Batch normalization
    12   ''   ReLU                    ReLU
    13   ''   Max Pooling             2×2 max pooling with stride [2  2] and padding [0  0  0  0]
    14   ''   Convolution             128 5×5 convolutions with stride [1  1] and padding 'same'
    15   ''   Batch Normalization     Batch normalization
    16   ''   ReLU                    ReLU
    17   ''   Max Pooling             2×2 max pooling with stride [2  2] and padding [0  0  0  0]
    18   ''   Convolution             156 5×5 convolutions with stride [1  1] and padding 'same'
    19   ''   Batch Normalization     Batch normalization
    20   ''   ReLU                    ReLU
    21   ''   Fully Connected         6 fully connected layer
    22   ''   Softmax                 softmax
    23   ''   Classification Output   crossentropyex
```

Figura 45[Z]: Capas que forman la red CNN 2D

RESULTADOS

Al finalizar el proceso de entrenamiento, se le da a la red el 20% de la muestra que ha sido reservada para el test y hace una clasificación. Con estos valores y la muestra de test se calcula la matriz de confusión:

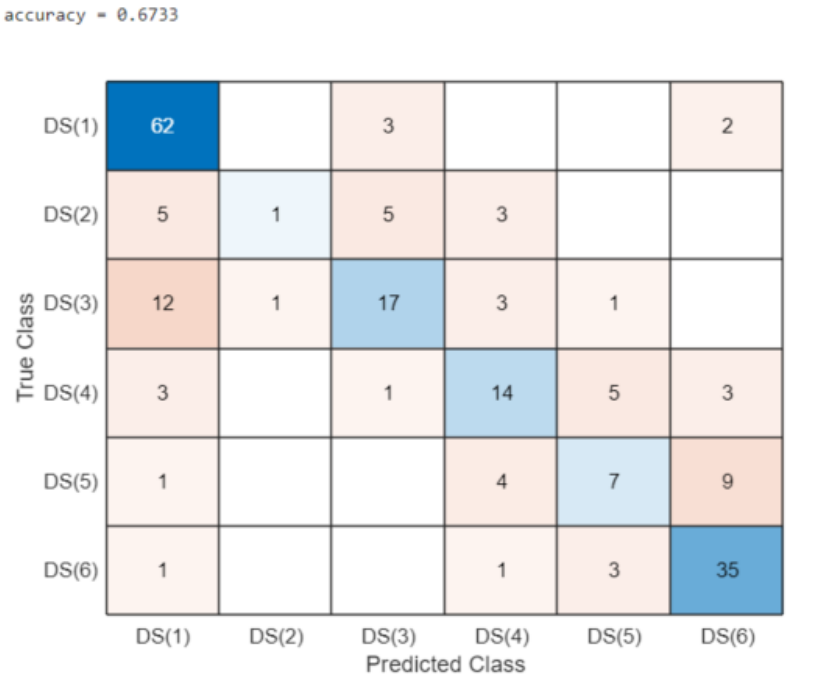


Figura 46[Z]: Matriz de confusión de la red CNN 2D

La precisión de este método es del 67.3%, bastante similar al 69% obtenido con la clasificación 1D. Una gran diferencia entre ambos métodos es la cantidad de tiempo que tarda en entrenarse cada red. Mientras que a la red CNN 1D trabaja con vectores de datos y tarda una media de 3 minutos en entrenarse, la red CNN 2D procesa imágenes, lo que hace que la complejidad temporal aumente considerablemente, llegando a sobrepasar la hora.

4. CONCLUSIONES. FUTURAS LÍNEAS DE INVESTIGACIÓN

Este trabajo de fin de grado recoge una introducción a las redes neuronales convolucionales, así como a su estructura y funcionamiento. En el apéndice se muestran algunas de las muchas aplicaciones que tienen, la detección de imágenes con la red GoogleNet y la clasificación de electrocardiogramas creando una red convolucional desde cero. En concreto para la clasificación de arritmias, una línea de investigación consistiría en optimizar ambas redes y conseguir una precisión superior. Esto puede ser posible, por ejemplo, con un software más potente o con un conjunto de datos más numeroso. Sería interesante, además, estudiar ambas opciones en paralelo (CNN 1D y 2D) para averiguar cuál de las dos es óptima para trabajar este tipo de datos.

Otra forma de enfocar el problema de clasificación de arritmias planteado es mediante fractales. Esta línea de estudio ha sido tratada en la asignatura del Taller de Tecnomatemática aplicada por ejemplo a la música clásica, el lenguaje de las ballenas o incluso el estudio de la estructura fractal de las precipitaciones en España.

Para finalizar, quisiera comentar brevemente un par de aplicaciones de las redes neuronales convolucionales bastante llamativas y que, a pesar de no haber sido incluidas en este trabajo, suponen una línea de investigación muy interesante y poco explotada para futuros trabajos.

En primer lugar, modelos de análisis de variables económicas basados en redes neuronales convolucionales[26], para predecir el desarrollo económico de cada industria en distintas regiones, el PIB anual o los valores de crecimiento de la industria primaria, secundaria y terciaria.

Otro enfoque son las redes neuronales aplicadas a sistemas de transporte, predicción de la demanda aérea[27], ferroviaria o marítima para la exportación a diferentes países o la planificación del transporte de un escuadrón para la defensa aérea de una región[28]. Este último punto está actualmente en desarrollo, con algunos estudios sobre todo en el ámbito del transporte aéreo.