# Redes Neuronales Profundas

Si recuerda, anteriormente nos encontramos con el teorema de aproximación universal, que establecía que un MLP con una sola capa oculta podría aproximarse a cualquier función. Pero si ese es el caso, ¿por qué necesitamos redes neuronales profundas? En pocas palabras, la capacidad de una red neuronal aumenta con cada capa oculta (y el cerebro tiene una estructura profunda). Lo que esto significa es que las redes más profundas tienen una expresividad mucho mayor que las redes menos profundas. Esto es algo con lo que nos encontramos antes cuando aprendimos sobre MLP. Vimos que al agregar capas ocultas, pudimos crear una red que pudo aprender a resolver un problema que una red neuronal lineal no podía.

Además, las redes más profundas se prefieren a las redes más amplias, no porque mejoren el rendimiento general, sino porque las redes con más capas ocultas (pero menos ancho) tienen muchos menos parámetros que las redes más amplias con menos capas ocultas.

Supongamos que tenemos dos redes: una amplia y otra profunda. Ambas redes tienen 20 entradas y 6 nodos de salida. Calculemos el número total de parámetros para ambas capas; es decir, el número de conexiones entre todas las capas y sesgos.

Nuestra amplia red neuronal tiene dos capas ocultas, cada una con 1.024 neuronas. El número total de parámetros es el siguiente:

page209image24494704

Nuestra red neuronal profunda tiene 12 capas ocultas, cada una con 150 neuronas. El número total de parámetros es el siguiente:

page209image24496992

Como puede ver, la red más profunda tiene menos de la mitad de los parámetros que la red más amplia.

Si bien este capítulo se centra en las redes neuronales profundas; Comenzaremos este capítulo discutiendo una característica importante de Keras llamada **API funcional**. Esta API actúa como un método alternativo para construir redes en tf.keras y nos permite construir redes más complejas que no se pueden lograr con el modelo secuencial API. La razón por la que nos centramos tanto en esta API es que se convertirá en una herramienta muy útil para construir redes profundas como las dos en las que nos enfocamos en este capítulo.

## API funcional

En la API de modelo secuencial se apila una capa encima de otra capa. Generalmente, se accederá al modelo a través de sus capas de entrada y salida. Y no existe un mecanismo simple si queremos agregar una entrada auxiliar en el medio de la red, o incluso extraer una salida auxiliar antes de la última capa.

Ese modelo también tenía sus desventajas; por ejemplo, no admite modelos de tipo gráfico o modelos que se comportan como funciones de Python. Además, también es difícil compartir capas entre los dos modelos. Estas limitaciones son abordadas por la API funcional y son la razón por la que es una herramienta vital para cualquiera que desee trabajar con modelos de aprendizaje profundo.

### La API funcional se rige por los dos conceptos siguientes:

* Una capa es una instancia que acepta un tensor como argumento. La salida de una capa es otro tensor. Para construir un modelo, las instancias de capa son objetos que están encadenados entre sí a través de tensores de entrada y salida. Esto tendrá un resultado final similar al de apilar varias capas en el modelo secuencial. Sin embargo, el uso de instancias de capa facilita que los modelos tengan entradas y salidas auxiliares o múltiples, ya que la entrada/salida de cada capa será fácilmente accesible.
* Un modelo es una función entre uno o más tensores de entrada y tensores de salida. Entre la entrada y la salida del modelo, los tensores son instancias de capa que están encadenadas entre sí por tensores de entrada y salida de capa. Por tanto, un modelo es una función de una o más capas de entrada y una o más capas de salida. La instancia del modelo formaliza el gráfico computacional sobre cómo fluyen los datos desde la (s) entrada (s) a la (s) salida (s).

Una vez que hayas completado la creación del modelo de API funcional, las mismas funciones que se utilizan en el modelo secuencial realizan el entrenamiento y la evaluación. Para ilustrar, en una API funcional, una capa convolucional bidimensional, Conv2D, con 32 filtros y con x como tensor de entrada de capa ey como tensor de salida de capa se puede escribir como:

1. y = Conv2D(32)(x)

También podemos apilar varias capas para construir nuestros modelos. Por ejemplo, podemos reescribir la red neuronal convolucional (CNN) en MNIST cnn-mnist-1.4.1.py usando la API funcional como se muestra en la siguiente lista:

## Crear un modelo de dos entradas y una salida

* Ahora vamos a hacer algo realmente emocionante, creando un modelo avanzado con dos entradas y una salida. Antes de comenzar, es importante saber que la API de modelo secuencial está diseñada para construir modelos de **1 entrada y 1 salida únicamente**.

Supongamos que se inventa un nuevo modelo para la clasificación de dígitos MNIST, y se llama Red-Y, como se muestra en la Figura 2.1.1. Y-Network usa la misma entrada dos veces, tanto en las ramas izquierda como derecha de CNN. La red combina los resultados mediante una capa de concatenación. La operación de fusión concatenar es similar a apilar dos tensores de la misma forma a lo largo del eje de concatenación para formar un tensor. Por ejemplo, la concatenación de dos tensores de forma (3, 3, 16) a lo largo del último eje dará como resultado un tensor de forma (3, 3, 32).

Todo lo demás después de la capa de concatenación seguirá siendo el mismo que el modelo de clasificador CNN MNIST del capítulo anterior: Flatten, luego Dropout y luego Dense:

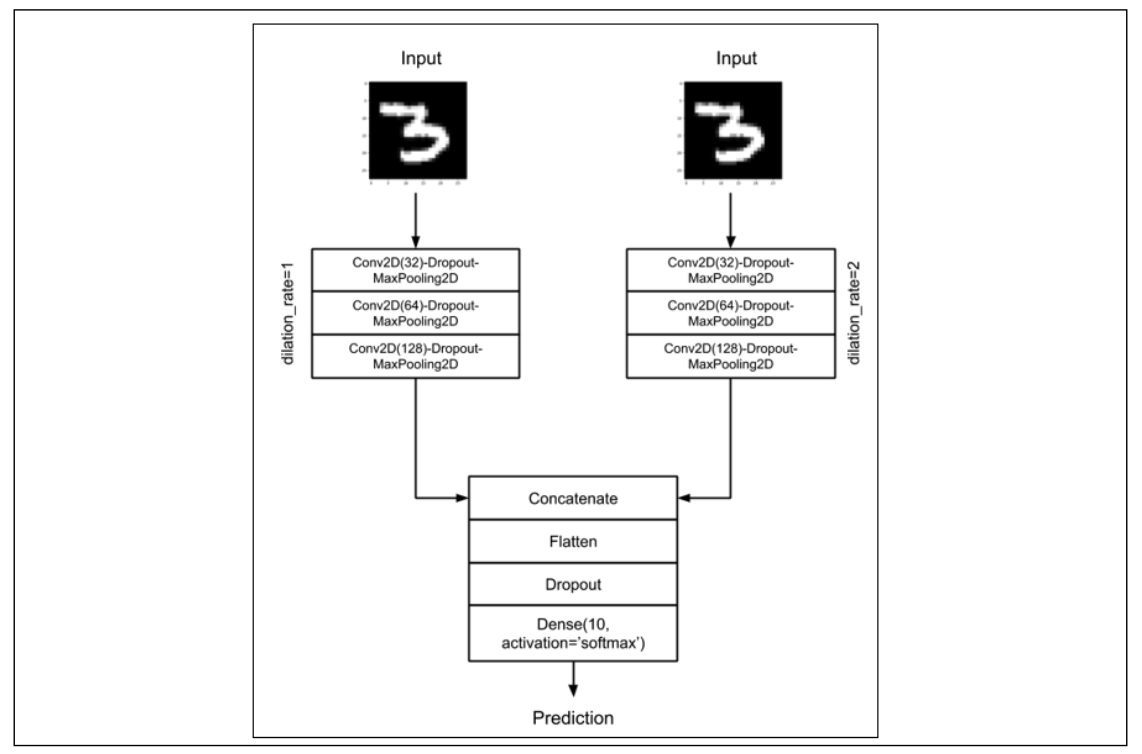
****

Figura 2.1.1: La red Y acepta la misma entrada dos veces pero procesa la entrada en dos ramas de redes convolucionales. Las salidas de las ramas se combinan utilizando la capa de concatenación. La predicción de la última capa será similar al modelo de clasificador CNN MNIST del capítulo anterior.

Para mejorar el rendimiento del modelo del Listado 2.1.1, podemos proponer varios cambios. En primer lugar, las ramas de la red Y están duplicando el número de filtros para compensar la reducción a la mitad del tamaño de los mapas de características después de MaxPooling2D (). Por ejemplo, si la salida de la primera convolución es (28, 28, 32), después de la agrupación máxima, la nueva forma es (14, 14, 32). La siguiente convolución tendrá un tamaño de filtro de 64 y dimensiones de salida de (14, 14, 64).

En segundo lugar, aunque ambas ramas tienen el mismo tamaño de grano de 3, la rama derecha utiliza una tasa de dilatación de 2. La figura 2.1.2 muestra el efecto de diferentes tasas de dilatación en un kernel con tamaño 3. La idea es que al aumentar el tamaño del campo receptivo efectivo del kernel usando la tasa de dilatación, la CNN permitirá que la rama derecha aprenda diferentes mapas de características. El uso de una tasa de dilatación superior a 1 es un método aproximado computacionalmente eficiente para aumentar el tamaño del campo receptivo. Es aproximado ya que el kernel no es en realidad un kernel completo. Es eficiente ya que usamos el mismo número de operaciones que con una tasa de dilatación igual a 1.

Para apreciar el concepto de campo receptivo, observe que cuando el núcleo calcula cada punto de un mapa de características, su entrada es un parche en el mapa de características de la capa anterior que también depende de su mapa de características de la capa anterior. Si continuamos rastreando esta dependencia hasta la imagen de entrada, el kernel depende de un parche de imagen llamado campo receptivo.

Usaremos la opción padding = 'same' para asegurarnos de que no tendremos dimensiones negativas del tensor cuando se use la CNN dilatada. Al usar padding = 'same', mantendremos las dimensiones de la entrada iguales a las de los mapas de características de salida. Esto se logra rellenando la entrada con ceros para asegurarse de que la salida tenga el mismo tamaño.

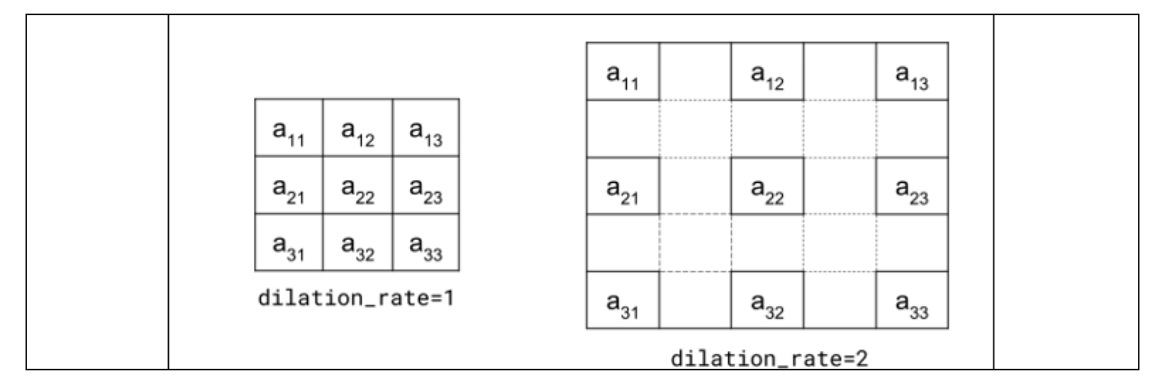


Figura 2.1.2: Al aumentar la tasa de dilatación de 1, el tamaño efectivo del campo receptivo del grano también aumenta

## Encabezado 2

* Utilice los estilos para dar formato a los documentos de Word en un momento. Por ejemplo, este texto usa el estilo Lista con viñetas.
* Vaya al grupo Estilos, que encontrará en la pestaña Inicio de la cinta de opciones, para aplicar el formato que desee en un simple paso.

|  |  |  |
| --- | --- | --- |
|  | Encabezado de columna | Encabezado de columna |
| Encabezado de fila | Texto | 123,45 |
| Encabezado de fila | Texto | 123,45 |
|  |  |  |