Una guía intuitiva de redes neuronales convolucionales.

En este artículo, exploraremos las redes neuronales convolucionales (CNN) y, en un alto nivel, veremos cómo se inspiran en la estructura del cerebro.

Índice

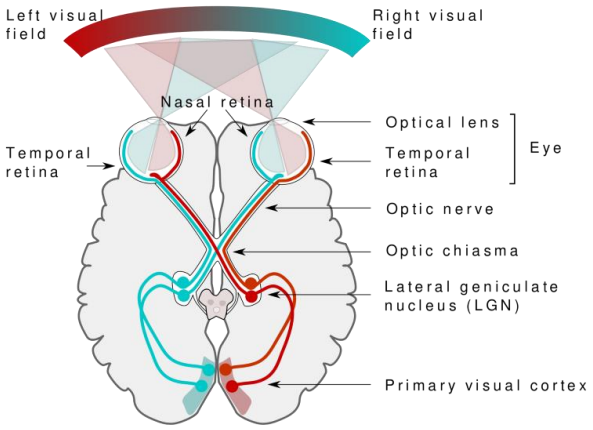
El artículo que viene a continuación es una traducción/resumen hecha por Icia Carro Barallobre: https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050

Otro artículo interesante sobre CNNs es: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

El cerebro

Estamos analizando constantemente el mundo que nos rodea. Sin esfuerzo consciente, hacemos predicciones sobre todo lo que vemos y actuamos sobre ellas.

Vemos, etiquetamos, hacemos predicciones, y reconocemos patrones en función de lo que hemos aprendido en el pasado. ¿Cómo es que podemos interpretar todo lo que vemos?



Mientras que la visión comienza en los ojos, la interpretación de lo que vemos sucede en el cerebro, en la corteza visual primaria.

Cuando vemos un objeto, los receptores de luz en nuestros ojos envían señales a través del nervio óptico a la corteza visual primaria, donde se procesa la entrada.

¿Cómo reconocemos objetos?

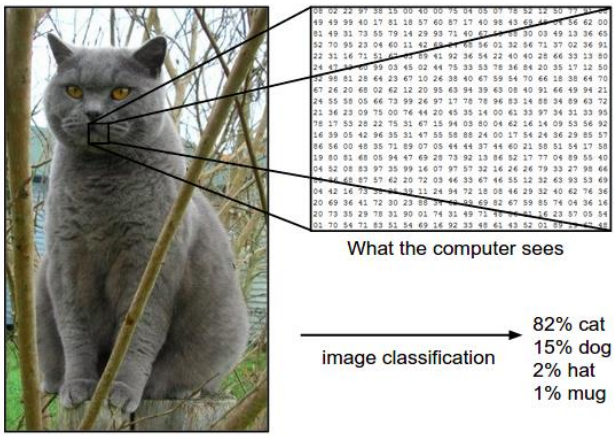
Desde pequeños, **aprendemos por ejemplos** que nos fueron dados a lo largo de nuestra vida, de manera lenta y segura comenzamos a reconocer ciertas cosas al verlas más a menudo en nuestro entorno: volviéndose tan comunes que la próxima vez que los viéramos, sabíamos instantáneamente cuál era su nombre: Se convirtieron en parte de nuestro **modelo** en el mundo.

Convolutional Neural Networks (CNN)

Análogamente al cerebro de un niño, necesitamos mostrar millones de imágenes a un algoritmo antes de que sea capaz de generalizar la entrada y hacer predicciones para imágenes que nunca antes había visto.

Las computadoras "ven" de una manera diferente a la nuestra: sólo números. Cada imagen se puede representar como matrices bidimensionales de números, conocidas como píxeles.

Esto no significa que no podamos entrenarlos para que reconozcan patrones como lo hacemos nosotros, solo tenemos que pensar qué es una imagen de una manera diferente.



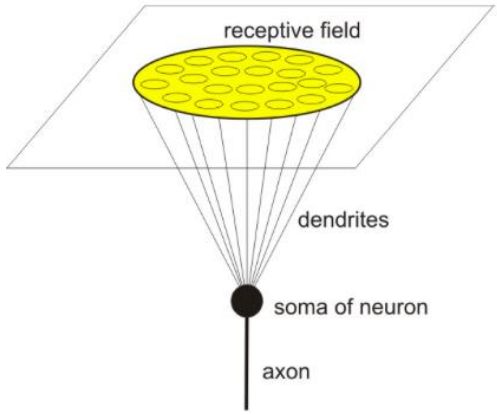
Para enseñar a un algoritmo a reconocer objetos en imágenes, usamos un tipo específico de red neuronal artificial: una red neuronal convolucional (CNN), inspiradas en el cerebro.

En el cerebro hay 2 tipos básicos de células de neuronas visuales que actúan de manera diferente: células simples (células S) y células complejas (células C).

Células simples: Se activan cuando identifican formas básicas (Líneas en área fija, ángulos específico, ...)

Células complejas: Tienen campos receptivos más grandes y su salida no es sensible a la posición específica en el campo ergo, continúan respondiendo a un cierto estímulo, aunque cambie su posición absoluta en la retina. (Complejo se refiere a más flexible, en este caso).

En la visión, un campo receptivo de una sola neurona sensorial es la región específica de la retina en la que algo afectará la activación de esa neurona. Cada célula neuronal sensorial tiene campos receptivos similares:



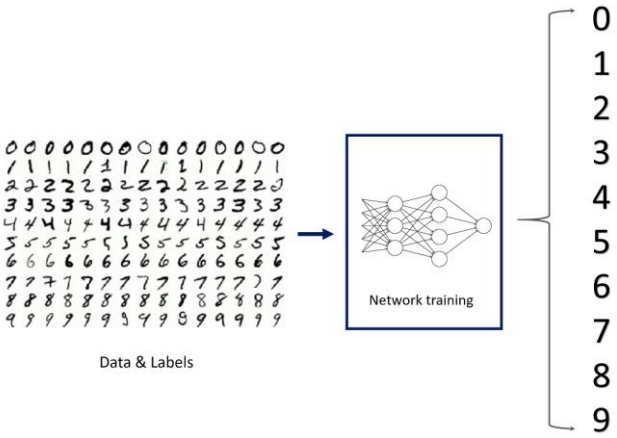
.

Además, el concepto de jerarquía juega un papel importante en el cerebro: La información se almacena en secuencias de patrones, en orden secuencial.

El neocórtex, que es la capa más externa del cerebro, almacena la información jerárquicamente: almacenándose la misma en columnas corticales, o agrupaciones de neuronas uniformemente organizadas.

Antes de llegar a las CNN, hubo otro modelo de red neuronal jerárquico: el neocognitron, inspirado por los conceptos de células simples y complejas, capaz de reconocer patrones al aprender sobre las formas de los objetos.

Más tarde, la primera red neuronal convolucional se llamó LeNet-5 y pudo clasificar los dígitos de los números escritos a mano.

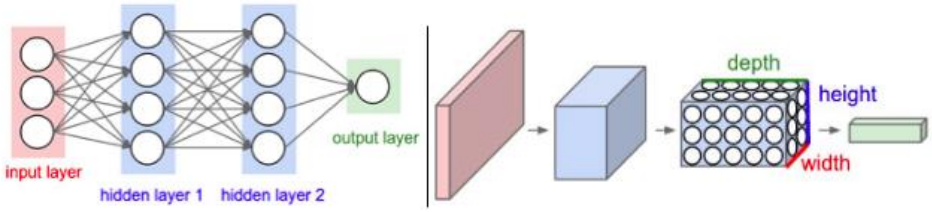


Arquitectura

Las redes neuronales convolucionales tienen una arquitectura diferente a las redes neuronales regulares.

Las redes neuronales regulares transforman una entrada colocándola a través de una serie de capas ocultas, cada capa está formada por un conjunto de neuronas donde cada capa está completamente conectada a todas las neuronas en la capa anterior. Finalmente, hay una última capa totalmente conectada, la capa de salida, que representa las predicciones.

En las CNN, las capas están organizadas en 3 dimensiones: ancho, alto y profundo. Además, las neuronas en una capa no se conectan a todas las neuronas en la siguiente capa sino solo a una pequeña región de la misma. Por último, la salida final se reducirá a un solo vector de puntajes de probabilidad, organizado a lo largo de la dimensión de profundidad.



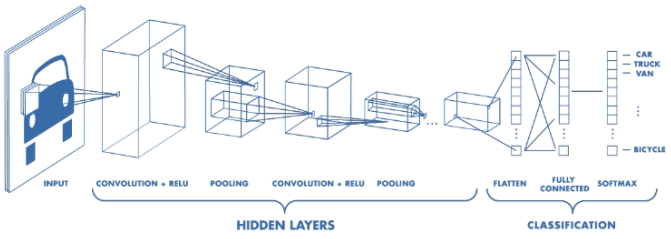
Las CNN tienen dos componentes:

- Las **capas ocultas / parte de extracción de características**: En esta parte, la red realizará una serie de convoluciones y operaciones de agrupación durante las cuales se detectan las características. (En una imagen de una cebra: esta es la parte donde la red reconocería sus franjas, dos orejas y cuatro patas)
- La parte de clasificación: Aquí, las capas totalmente conectadas servirán como un clasificador por encima de estas características extraídas. Asignarán una probabilidad para que el objeto en la imagen sea lo que el algoritmo predice que es.

```
# before we start building we import the libraries

import numpy as np

from keras.layers import Conv2D, Activation, MaxPool2D, Flatten,
Dense
from keras.models import Sequential
```



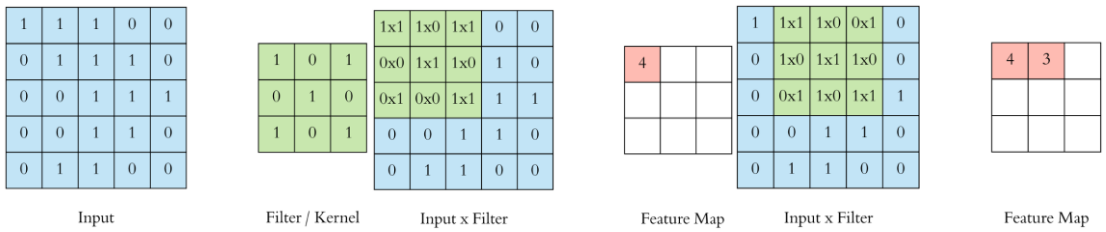
Extracción de características

La convolución es uno de los principales bloques de construcción de una CNN.

El término convolución se refiere a la combinación matemática de dos funciones para producir una tercera función: Fusiona dos conjuntos de información.

En el caso de una CNN, la convolución se realiza en los datos de entrada con el uso de un filtro (también llamado kernel) para producir un mapa de características.

Ejecutamos una convolución deslizando el filtro sobre la entrada:



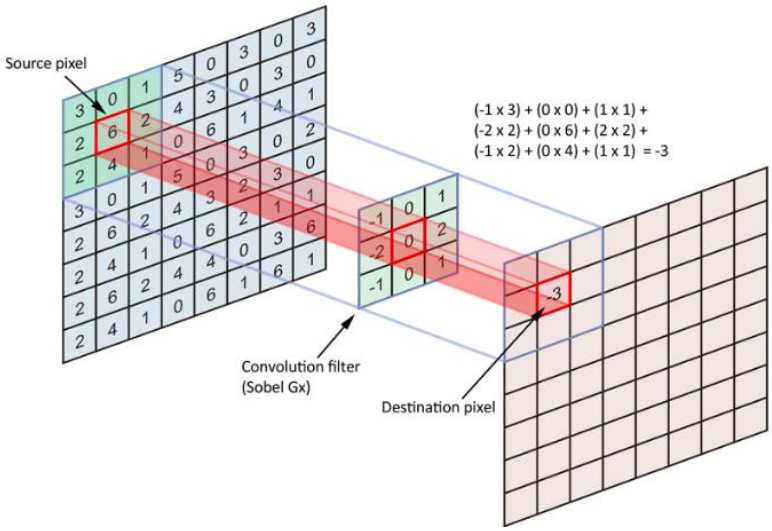
En cada ubicación, se realiza una multiplicación de matrices y suma el resultado en el mapa de características.

El filtro (cuadrado verde) se está deslizando sobre la entrada (cuadrado azul) y la suma de la convolución va al mapa de características (cuadrado rojo).

El área de nuestro filtro también se denomina campo receptivo, que lleva el nombre de las células neuronales: El tamaño de este filtro es 3x3.

Aunque el ejemplo esté en 2D, en realidad las circunvoluciones se realizan en 3D: Cada imagen se representa como una matriz 3D con una dimensión para el ancho, la altura y la profundidad.

La profundidad es una dimensión debido a los canales de colores utilizados en una imagen (RGB).



Realizamos numerosas circunvoluciones en nuestra entrada, donde cada operación utiliza un filtro diferente: Esto resulta en diferentes mapas de características. Al final, tomamos todos estos mapas de características y los juntamos como la salida final de la capa de convolución.

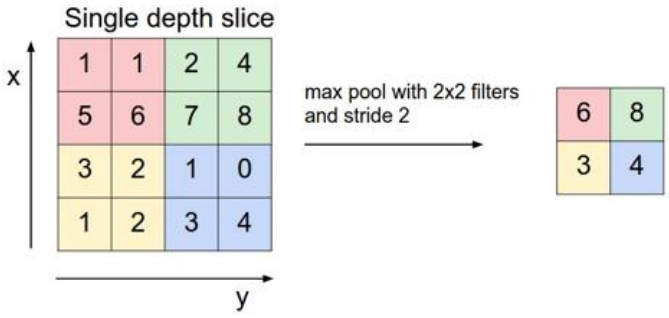
(AQUÍ)

Al igual que cualquier otra red neuronal, usamos una función de activación para hacer que nuestra salida no sea lineal: En el caso de una CNN, la salida de la convolución pasará a través de la función de activación. Esta podría ser la función de activación ReLU.Paso es el tamaño del paso que el filtro de convolución mueve cada vez. Un tamaño de paso suele ser 1, lo que significa que el filtro desliza píxel por píxel. Al aumentar el tamaño de zancada, su filtro se desliza sobre la entrada con un intervalo mayor y por lo tanto tiene menos superposición entre las celdas.La siguiente animación muestra el tamaño de zancada 1 en acción.

https://cdn-images-1.medium.com/max/800/0*iqNdZWyNeCr5tCkc.

Debido a que el tamaño del mapa de características siempre es más pequeño que la entrada, tenemos que hacer algo para evitar que nuestro mapa de características se reduzca: Aquí es donde usamos el relleno.Se agrega una capa de píxeles de valor cero para rodear la entrada con ceros, para que nuestro mapa de características no se reduzca. Además de mantener el tamaño espacial constante después de realizar la convolución, el relleno también mejora el rendimiento y se asegura de que el tamaño del kernel y la zancada encajen en la entrada.

Después de una capa de convolución, es común agregar una capa de agrupación entre las capas CNN. La función de la agrupación es reducir continuamente la dimensionalidad para reducir el número de parámetros y el cálculo en la red: Esto acorta el tiempo de entrenamiento y controla el sobreajuste.El tipo de agrupación más frecuente es la agrupación máxima, que toma el valor máximo en cada ventana. Estos tamaños de ventana deben especificarse de antemano. Esto disminuye el tamaño del mapa de características al tiempo que mantiene la información significativa.



Por lo tanto, al utilizar una CNN, los cuatro hiperparámetros importantes que tenemos que decidir son:

- el tamaño del núcleo
- el recuento de filtros (es decir, cuántos filtros queremos utilizar)
- zancada (qué tan grandes son los pasos del filtro)
- Relleno

```
# Images fed into this model are 512 x 512 pixels with 3 channels

img_shape = (28,28,1)

# Set up the model

model = Sequential()

# Add convolutional layer with 3, 3 by 3 filters and a stride size of 1
# Set padding so that input size equals output size

model.add(Conv2D(6,2,input_shape=img_shape))

# Add relu activation to the layer

model.add(Activation('relu'))

#Pooling

model.add(MaxPool2D(2))
```

Una buena manera de visualizar una capa de convolución se muestra a continuación. Trate de verlo un poco y realmente entienda lo que está sucediendo: https://cdn-images-1.medium.com/max/800/1*_34EtrgYk6cQxlJ2br51HQ.gif

Clasificación

Después de la convolución y la agrupación de capas, nuestra parte de clasificación consta de unas pocas capas totalmente conectadas. Sin embargo, estas capas totalmente conectadas solo pueden aceptar datos de 1 dimensión.

Para convertir nuestros datos 3D a 1D, usamos la función *flatten* en Python. Esto esencialmente arregla nuestro volumen 3D en un vector 1D.Las últimas capas de un NN convolucional son capas totalmente conectadas. Las neuronas en una capa completamente conectada tienen conexiones completas con todas las activaciones en la capa anterior. Esta parte es, en principio, lo mismo que una red neuronal regular.

```
#Fully connected layers

# Use Flatten to convert 3D data to 1D
model.add(Flatten())

# Add dense layer with 10 neurons
model.add(Dense(10))

# we use the softmax activation function for our last layer
model.add(Activation('softmax'))

# give an overview of our model

model.summary
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 27, 27, 6)         30
_________________________________________________________________
activation_1 (Activation)    (None, 27, 27, 6)         0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 6)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1014)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10150
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
=================================================================
Total params: 10,180
Trainable params: 10,180
Non-trainable params: 0
_________________________________________________________________
```

Entrenamiento

Entrenar a una CNN funciona de la misma manera que una red neuronal regular, utilizando la inclinación hacia atrás o el descenso de gradiente. Sin embargo, aquí esto es un poco más complejo matemáticamente debido a las operaciones de convolución.Si desea leer más sobre cómo funcionan las redes neuronales regulares, puede leer el artículo anterior: https://medium.freecodecamp.org/building-a-3-layer-neural-network-from-scratch-99239c4af5d3

```
"""Before the training process, we have to put together a learning process in a particular form. It consists
of 3 elements: an optimiser, a loss function and a metric."""

model.compile(loss='sparse_categorical_crossentropy', optimizer = 'adam', metrics=['acc'])

# dataset with handwritten digits to train the model on
from keras.datasets import mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = np.expand_dims(x_train,-1)

x_test = np.expand_dims(x_test,-1)

# Train the model, iterating on the data in batches of 32 samples
# for 10 epochs

model.fit(x_train, y_train, batch_size=32, epochs=10, validation_data=(x_test,y_test)

# Training...

Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 10s 175us/step - loss: 4.0330 - acc: 0.7424 - val_loss: 3.5352
- val_acc: 0.7746
Epoch 2/10
60000/60000 [==============================] - 10s 169us/step - loss: 3.5208 - acc: 0.7746 - val_loss: 3.4403
- val_acc: 0.7794
Epoch 3/10
60000/60000 [==============================] - 11s 176us/step - loss: 2.4443 - acc: 0.8372 - val_loss: 1.9846
- val_acc: 0.8645
Epoch 4/10
60000/60000 [==============================] - 10s 173us/step - loss: 1.8943 - acc: 0.8691 - val_loss: 1.8478
- val_acc: 0.8713
Epoch 5/10
60000/60000 [==============================] - 10s 174us/step - loss: 1.7726 - acc: 0.8735 - val_loss: 1.7595
- val_acc: 0.8718
Epoch 6/10
60000/60000 [==============================] - 10s 174us/step - loss: 1.6943 - acc: 0.8765 - val_loss: 1.7150
- val_acc: 0.8745
Epoch 7/10
60000/60000 [==============================] - 10s 173us/step - loss: 1.6765 - acc: 0.8777 - val_loss: 1.7268
- val_acc: 0.8688
Epoch 8/10
60000/60000 [==============================] - 10s 173us/step - loss: 1.6676 - acc: 0.8799 - val_loss: 1.7110
- val_acc: 0.8749
Epoch 9/10
60000/60000 [==============================] - 10s 172us/step - loss: 1.4759 - acc: 0.8888 - val_loss: 0.1346
- val_acc: 0.9597
Epoch 10/10
60000/60000 [==============================] - 11s 177us/step - loss: 0.1026 - acc: 0.9681 - val_loss: 0.1144
- val_acc: 0.9693
```

Después de entrenar el primer lote, Keras estima la duración del entrenamiento (ETA: tiempo estimado de llegada) de una época, que es equivalente a una ronda de entrenamiento con todas sus muestras.

Además de eso, obtiene las pérdidas (la diferencia entre la predicción y las etiquetas verdaderas) y su métrica (en su caso, la precisión) tanto para el entrenamiento como para las muestras de validación.

Nota: si su pérdida / precisión de validación comienza a aumentar mientras su pérdida / precisión de entrenamiento aún está disminuyendo, esto es un indicador de sobreajuste.

Nota 2: al final debes probar tu NN contra algún **equipo de prueba** eso es diferente de tu conjunto de entrenamiento y conjunto de validación y, por lo tanto, nunca ha sido tocado durante el proceso de entrenamiento.

Resumen

En resumen, las CNN son especialmente útiles para la clasificación y reconocimiento de imágenes. Tienen dos partes principales:

Una parte de extracción de características y una parte de clasificación.

La principal técnica especial en las CNN es la convolución, donde un filtro se desliza sobre la entrada y combina el valor de entrada + el valor del filtro en el mapa de características.

Al final, nuestro objetivo es alimentar nuevas imágenes a nuestra CNN para que pueda dar una probabilidad para el objeto que cree que ve o describir una imagen con texto.

TEXTO POSTER

Autor/a/s: Icía Carro Barallobre, Lucía Álvarez

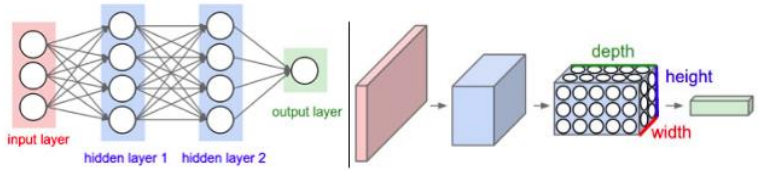
CNN y su analogía con el cerebro

Constantemente estamos analizando el mundo que nos rodea: Vemos, etiquetamos, hacemos predicciones y reconocemos patrones en función de lo que hemos aprendido, análogamente, necesitamos mostrar imágenes a un algoritmo antes de que sea capaz de generalizar la entrada y hacer predicciones para imágenes que nunca antes había visto.

Para enseñar a un algoritmo a reconocer objetos en imágenes, usamos un tipo específico de RNA: las redes neuronales convolucionales (CNN), inspiradas en el cerebro.

En el cerebro hay 2 tipos básicos de células de neuronas visuales que actúan de manera diferente: células simples que se activan cuando identifican formas básicas y, células complejas que tienen campos receptivos más grandes y su salida no es sensible a la posición específica en el campo por lo que continúan respondiendo a un cierto estímulo, aunque cambie su posición absoluta en la retina.

Arquitectura (RNA convencional vs CNN)



En las CNN, las capas están organizadas en 3 dimensiones: ancho, alto y profundo. Además, las neuronas en una capa no se conectan a todas las neuronas en la siguiente capa sino solo a una pequeña región de la misma y, la salida final se reducirá a un solo vector de puntajes de probabilidad, organizado a lo largo de la dimensión de profundidad.

Las CNN tienen dos componentes:

- **Parte de extracción de características (Capa oculta)**: En esta parte, la red realizará una serie de convoluciones y operaciones de agrupación durante las cuales se detectan las características. (En una imagen de una cebra: esta es la parte donde la red reconocería sus franjas, dos orejas y cuatro patas)
- **Parte de clasificación**: Aquí, las capas totalmente conectadas servirán como un clasificador por encima de estas características extraídas. Asignarán una probabilidad para que el objeto en la imagen sea lo que el algoritmo predice que es.

Extracción de características

(TO-DO) Lucia.

Clasificación

Después de la convolución y la agrupación de capas, nuestra parte de clasificación consta de unas pocas capas totalmente conectadas. Sin embargo, estas capas totalmente conectadas solo pueden aceptar datos de 1 dimensión, debemos convertir nuestro volumen 3D en un vector 1D. Las últimas capas de un NN convolucional son capas totalmente conectadas: como en una RNA convencial.

Entrenamiento

Entrenar a una CNN funciona de la misma manera que una red neuronal regular, utilizando la inclinación hacia atrás o el descenso de gradiente. Sin embargo, aquí esto es un poco más complejo matemáticamente debido a las operaciones de convolución.

Conclusión

En resumen, las CNN son especialmente útiles para la clasificación y reconocimiento de imágenes. Contienen dos partes: una parte de extracción de características y una parte de clasificación. La principal técnica especial en las CNN es la convolución: donde un filtro se desliza sobre la entrada y combina el valor de entrada más el valor del filtro en el mapa de características.