

Redes Convolucionales

Funciona para clasificación de imágenes.

¿Por qué decimos convoluciones? porque hace sumas y multiplicaciones. cuando tienes algo tan complejo como imágenes tendrías que tener ya demasiadas neuronas, no te alcanza con nada, no escala bien, entonces hacemos mejoras/convoluciones para que nuestras redes sean resistentes a más cosas.

Jerarquía de representaciones: los modelos que utilizamos empiezan a extraer características de los datos entonces se empieza a formar una idea de que necesito para resolver ese problema y esa idea que se va formando el modelo a medida que va avanzando es más abstracta. Las últimas capas aprenden cosas más complejas.



En principio voy a aprender features simples, y a partir de ellos construyo features más complejos que uso para resolver el problema.

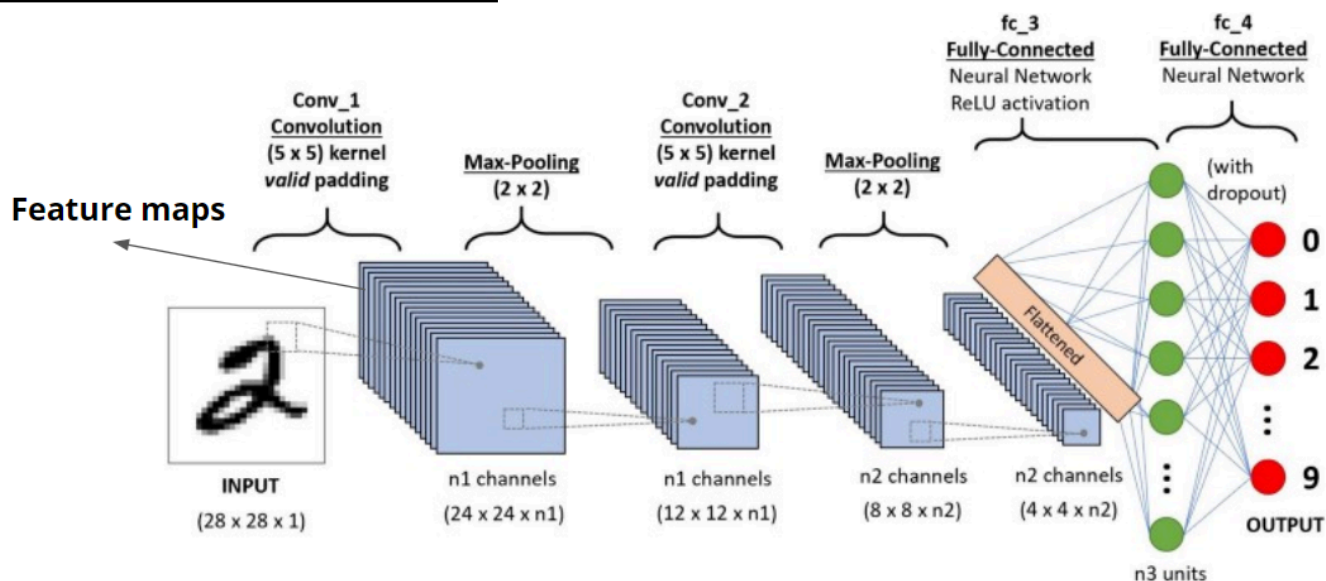
Forma de las redes convolucionales

Las redes neuronales comunes no escalan bien en imágenes (usar fully connected (son las capas que habíamos visto antes) no es una buena opción en imágenes)

Volúmenes 3D de neuronas: Se intenta mantener la arquitectura de las imágenes (altura, ancho, canales).

Las **capas convolucionales** son las que **mantienen la estructura de la imagen**

Arquitectura de las redes convolucionales



Capa de feature extraction: las capas de los cuadraditos. Tienen adentro las capas de convoluciones y las capas de pooling. a medida que avanzo en la capa de feature extraction, la imagen se achica y se agregan más canales.

Capa de clasificacion: neuronas que se conectan con todo el volumen de entrada. Son capas densas comunes. Se intenta que la imagen sea lo más chica posible porque las capas densas son costosas.

Las dos capas del feature extraction son:

capa de convolucion: agrega canales.

capa de pooling: disminuyen la dimensionalidad, la mas usada es **maxPool**. entonces despues de aplicar el filtro te queda algo más chico. pooling NO afecta la cantidad de canales

En las capas de convolución se aplican “**filtros**”, cuyos parámetros son aprendidos durante el aprendizaje. A estos filtros también se los llama “**kernels**”. Son los encargados de buscar features relevantes.

filtros/kernel: El filtro es una matriz de tamaño fijo y el truco de la convolución es agarrar una matriz grande y multiplicar por esa matriz chica, y después se suma o se hace otra operación. Con respecto a la convolución anterior, se va teniendo menos parámetros para aprender. Están en las capas de feature extraction. Puede haber más de un filtro al mismo tiempo, se suman sus resultados. Nos devuelve un **tensor** (vector de matrices)

Propiedades de los filtros (kernels)

- Stride: paso del filtro. No se recomienda más de dos.
- Padding: relleno de la imagen. Hay ceros en los bordes de las imágenes. Para no perder información.
- Profundidad: cantidad de filtros que se aplican en cada capa.

Si paso de un canal a n1 canales, aplique n1 filtros. 5x5 kernel quiere decir que el filtro es de 5x5. Las capas de convolución aumentan la cantidad de canales.

Redes convolucionales

Las redes convolucionales tienen la particularidad de tener ciertas propiedades de **invarianza**, dentro de su estructura, útiles a la hora de trabajar con imágenes. Estas propiedades de invarianza se meten en la arquitectura del modelo:

- **Compartir pesos** utilizando un mismo kernel. Uso un filtro chico y los pesos los voy compartiendo. Se reducen mucho la cantidad de parámetros.
- **Subsampleando**: Imágenes invariantes ante pequeñas rotaciones y traslaciones. las capas de pooling.
- Noción de “campo receptivo local”. **Conectividad local**: neuronas conectadas a una porción del espacio de la entrada (campo receptivo - tamaño del filtro).

Métodos de regularización: se usan los de las redes neuronales que vimos, porque son redes. Además se usa Batch Normalization: mejora el aprendizaje de la red normalizando la entrada de cada capa convolucional.

- Dropout
- L1, L2, (L1+L2)
- Early stopping
- Data augmentation
- Batch normalization



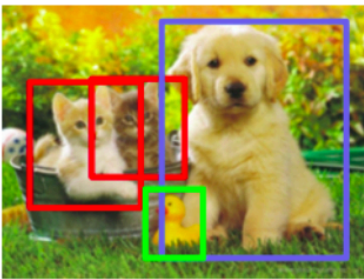

Transferencia de aprendizaje: la idea es usar las feature extraction layers, pudiendo hacer un poco de **fine tuning** (using weights of a previous deep learning algorithm for programming another similar deep learning process). A veces las imágenes se pueden representar de una manera bastante común entre distintos tipos de problemas de clasificación, muchas veces se puede usar una red para hacer feature engineering y después agregamos al final una de clasificación.

Se agregaron conexiones intermedias por el tema de que los gradientes disminuían mucho y las primeras capas no tenían tanto impacto.

Al final se usa un cross entropy como función de error porque clasifica varias clases. Se usa una softmax como activación. No serviría usar varias sigmoideas.

Anda peor en training que en validación. Lo más probable que esté pasando es que el data augmentation esté afectando.

Las tareas que se resuelven son:

Classification	Classification + Localization	Object Detection	Instance Segmentation
			
CAT	CAT	CAT, DOG, DUCK	CAT, DOG, DUCK