Deep Learning: Percepción Computacional, Foundation Models y Transfer Learning

Inteligencia Artificial e Ingeniería del Conocimiento

Constantino Antonio García Martínez

Universidad San Pablo Ceu

Introducción a las ConvNets

Introducción a las ConvNets

Aplicaciones Actuales

Aplicaciones Actuales de las ConvNets

Las ConvNets son fundamentales para las tareas modernas de visión por computador:

- **Imagen Médica**: Detección de enfermedades en rayos X, resonancias magnéticas y TACs
- **Vehículos Autónomos**: Detección de objetos, detección de carriles y comprensión de escenas
- **Control de Calidad**: Detección de defectos de fabricación e inspección de productos
- **Seguridad**: Reconocimiento facial, sistemas de vigilancia y detección de anomalías
- **Aplicaciones Móviles**: Modo retrato, filtros faciales y mejora de imágenes
- **Foundation Models**: Componente clave en transformers de visión (ViT) y modelos multimodales

Aplicaciones en Detalle

- **Imagen Médica**

(a) True positive, p=0.9977 (b) False positive, p=0.9040

(c) True negative, p=0.0506 (d) False negative, p=0.1538

Aplicaciones en Detalle

- **Vehículos Autónomos**

Introducción a las ConvNets

Entendiendo las Imágenes

¿Qué es una imagen?

Las imágenes en escala de grises son simplemente matrices H x W de valores en el rango [0, 255].

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

¿Qué es una imagen?

Las imágenes en color son 3 matrices H x W apiladas con valores en el rango [0, 255]. Cada matriz es un **canal de color**. Alternativamente, las imágenes en color pueden verse como un tensor de tamaño H x W x 3.

Original Color Image

Matlab RGB Matrix

Fundamentos de las Convoluciones

Fundamentos de las Convoluciones

Conceptos Básicos

¿Por qué no usar redes densas?

¿Por qué no usar redes densas?

Nos gustaría que la clasificación fuera **invariante** a la traslación, rotación y escala.

Entendiendo las Convoluciones de Forma Intuitiva

Piensa en la convolución como una forma sistemática de **buscar patrones**:

- El **filtro (kernel)** es como una plantilla del patrón que buscamos
- Valores altos en la salida = fuerte presencia del patrón

Convoluciones

Convoluciones

Los filtros han sido estudiados y diseñados manualmente durante mucho tiempo para detectar patrones en imágenes:

Convoluciones

Por ejemplo, el filtro de Prewitt para detectar bordes verticales es:

$$W = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Con redes neuronales, la idea es que el filtro se aprende

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

Las redes neuronales que usan convoluciones se denominan **redes neuronales convolucionales (ConvNets o CNNs)**.

Fundamentos de las Convoluciones

Capas Convolucionales

Capas Convolucionales

Capas Convolucionales

- Una capa convolucional acepta una "imagen" de tamaño $W_1 \times H_1 \times C_1$
- Requiere cuatro hiperparámetros:
  - Número de filtros F
  - Tamaño del kernel K
  - Stride S
  - Cantidad de padding P
- Produce una nueva "imagen" de tamaño $W_2 \times H_2 \times C_2$ donde:
  - $W_2 = (W_1 - K + 2P)/S + 1$
  - $H_2 = (H_1 - K + 2P)/S + 1$
  - $C_2 = F$
- El canal d-ésimo de salida es el resultado de convolucionar el d-ésimo filtro sobre la "imagen" de entrada
- El número total de parámetros es $K \cdot K \cdot D_1 \cdot F$ pesos y F bias

Construcción de Arquitecturas ConvNet

Construcción de Arquitecturas ConvNet

Componentes Esenciales

¿Por qué Necesitamos Pooling?

Pooling Enables Multi-Scale Pattern Detection

Through hierarchical feature processing

¿Por qué Necesitamos Pooling?

Las capas de pooling sirven para múltiples propósitos:

- **Reducción de Dimensionalidad**:
  - Reduce la carga computacional
  - Disminuye el uso de memoria
  - Hace los mapas de características más manejables
- **Invarianza a la Traslación**:
  - Pequeños desplazamientos en la entrada producen la misma salida
  - Ayuda a reconocer objetos independientemente de su posición exacta
- **Procesamiento Jerárquico**:
  - Ayuda a capturar características a diferentes escalas
  - Crea una jerarquía implícita de características

Capa de Pooling

Construcción de Arquitecturas ConvNet

Implementación Práctica

Implementación Práctica de ConvNets

Las ConvNets modernas típicamente siguen un patrón repetitivo:

1. **Bloque Convolucional**:
   - Capa Conv2D (aprende características)
   - BatchNormalization (estabiliza el entrenamiento)
   - Activación ReLU (añade no-linealidad)
   - MaxPooling (reduce dimensiones)
   - Opcional: Dropout (previene sobreajuste)
2. **Repetir** bloques varias veces
3. **Cabeza de Clasificación (Classification Head)**:
   - Capa Flatten (convierte 2D a 1D)
   - Capas Densas (clasificación final)
   - Activación final (depende del tipo de problema)

Implementación Práctica de ConvNets

```
model = models.Sequential()

# First Convolutional Block
model.add(layers.Conv2D(32, (3, 3), padding='same', input_shape=(28, 28, 1)))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.25))

# Second Convolutional Block
model.add(layers.Conv2D(64, (3, 3), padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.25))

# Other Convolutional Block
# ....

# Classification Head
model.add(layers.Flatten())
model.add(layers.Dense(128))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(10, activation='softmax'))
```

Code Example: MNIST con ConvNets

Técnicas Avanzadas

Técnicas Avanzadas

Data Augmentation

Data Augmentation

¿Por qué?

- Más datos de entrenamiento
- Mejor generalización
- Previene el sobreajuste

Técnicas de Data Augmentation

Code Example: Data Augmentation

Técnicas Avanzadas

Transfer Learning y Foundation Models

Transfer Learning

Tres estrategias:

Feature Extraction | Fine-tuning | Progressive Fine-tuning

Foundation Models

Entrenados con conjuntos de datos masivos

Entrenados en múltiples tareas

Base para transfer learning

Traditional ML

Foundation Models

Estrategias de Fine Tuning

¿Cuándo Usar Cada Estrategia?

- **Feature Extraction**:
  - Dataset pequeño
  - Tarea similar
- **Fine-tuning**:
  - Más datos
  - Tarea diferente
- **Desde Cero**:
  - Dominio único
  - Dataset grande

Code Example: Transfer Learning

Code Exercise: ConvNets Prácticas