TÓM TẮT : DEEP COMPUTER VISION SỬ DỤNG MẠNG NORON TÍCH CHẬP (CNNS)

Mục lục

1 Giới thiệu về Mạng Nơ-ron Tích chập (CNNs)

Mạng Nơ-ron Tích chập (Convolutional Neural Networks - CNNs) là kiến trúc được thiết kế đặc biệt cho dữ liệu dạng lưới, như hình ảnh. Chúng lấy cảm hứng từ vỏ não thị giác của động vật, với các lớp tích chập (Convolutional Layers) đóng vai trò là các bộ lọc học được để phát hiện các mẫu hình cục bộ.

1.1 Các ưu điểm chính của CNN

1. **Chia sẻ Trọng số (Parameter Sharing):** Cùng một tập hợp trọng số (bộ lọc/kernel) được áp dụng trên toàn bộ ảnh, giảm đáng kể số lượng tham số cần học.
2. **Kết nối cục bộ (Local Connectivity):** Mỗi nơ-ron chỉ kết nối với một vùng nhỏ của đầu vào, được gọi là Trường Tiếp nhận Cục bộ (Local Receptive Field).
3. **Bất biến dịch chuyển (Translation Invariance):** Khả năng nhận dạng một đối tượng bất kể vị trí của nó trong hình ảnh.

2 Các Lớp Kiến trúc cơ bản của CNN

Một CNN điển hình bao gồm ba loại lớp chính: Lớp Tích chập (Conv), Lớp Gộp (Pooling), và Lớp Kết nối Đầy đủ (Dense).

2.1 Lớp Tích chập (Convolutional Layer)

Lớp tích chập là cốt lõi của CNN. Nó tính toán tích chập giữa bộ lọc (kernel) và đầu vào.

2.1.1 Phép Tích chập 2D

Phép tích chập 2D được định nghĩa như sau, trong đó I là ma trận đầu vào (ảnh), K là kernel/bộ lọc, và O là ma trận đầu ra (feature map):

$$O(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

Trong thực tế học sâu, một phiên bản biến thể thường được sử dụng là **tích chập chéo (cross-correlation)**, trong đó kernel được lật trước khi tính toán.

2.1.2 Padding và Strides

- **Padding:** Thêm các hàng/cột giá trị 0 xung quanh biên của ảnh.
  - **Valid Padding:** Không sử dụng padding. Kích thước đầu ra giảm.
  - **Same Padding:** Sử dụng padding để đầu ra có kích thước gần bằng đầu vào.
- **Strides:** Bước nhảy của kernel khi di chuyển qua ảnh. S_h là bước nhảy dọc, S_w là bước nhảy ngang.

Kích thước đầu ra $O_h \times O_w$ với đầu vào $I_h \times I_w$, kernel $K_h \times K_w$, stride S_h, S_w, và padding P:

$$O_h = \lfloor \frac{I_h + 2P - K_h}{S_h} \rfloor + 1$$

$$O_w = \lfloor \frac{I_w + 2P - K_w}{S_w} \rfloor + 1$$

2.1.3 Convolutional Layers trong Keras (Ví dụ)

Trong Keras, lớp 'Conv2D' thường được sử dụng.

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    # First convolutional layer: 32 filters, 3x3 kernel, ReLU activation, same
    padding
    layers.Conv2D(32, (3, 3), activation='relu', padding='same',
                  input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),  # Max pooling layer
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(10, activation='softmax')
])
```

Listing 1: Sử dụng Lớp Conv2D trong Keras

2.2 Lớp Gộp (Pooling Layer)

Mục đích của lớp gộp là giảm kích thước không gian (chiều cao và chiều rộng) của feature map, giảm số lượng tham số và tính toán, và làm cho mạng nơ-ron ít nhạy cảm hơn với sự dịch chuyển nhỏ của các đặc trưng.

2.2.1 Max Pooling

Max Pooling là phổ biến nhất, nó chỉ giữ lại giá trị lớn nhất từ vùng gộp.

$$O(i,j) = \max_{m,n \in \text{Vùng gộp}} I(m,n)$$

2.2.2 Global Average Pooling (GAP)

GAP giảm kích thước toàn bộ feature map xuống còn 1×1 bằng cách tính giá trị trung bình của toàn bộ feature map. Nó thường được sử dụng trước lớp Dense cuối cùng để giảm đáng kể số lượng tham số.

2.3 Lớp Kết nối Đầy đủ (Dense Layer)

Các lớp này được đặt ở cuối CNN, sau khi đã trích xuất các đặc trưng. Chúng thực hiện phân loại dựa trên các feature map đã được làm phẳng (Flattened).

3 Các Kiến trúc CNN Nổi tiếng

3.1 LeNet-5 (1998)

Một trong những CNN đầu tiên và là nền tảng của các kiến trúc hiện đại, được sử dụng để nhận dạng chữ số viết tay (MNIST).

Cấu trúc cơ bản: (CONV $\rightarrow$ AVG POOL) $\times 2 \rightarrow$ CONV $\rightarrow$ Flatten $\rightarrow$ Dense $\times 2$.

3.2 AlexNet (2012)

Người tiên phong trong thời đại học sâu, đã giành chiến thắng lớn trong cuộc thi ImageNet ILSVRC 2012.

Đổi mới chính:

1. **Sử dụng ReLU:** Thay thế hàm kích hoạt Sigmoid/Tanh bằng ReLU (Rectified Linear Unit), giúp tăng tốc độ hội tụ.
2. **Sử dụng Dropout:** Giúp điều chỉnh hóa (regularization) và tránh overfitting.
3. **Tăng cường dữ liệu (Data Augmentation):** Tăng cường tính tổng quát.
4. **Local Response Normalization (LRN):** (Không còn được sử dụng rộng rãi).

3.3 GoogLeNet / Inception (2014)

Đã giới thiệu khối **Inception** để giải quyết vấn đề lựa chọn kích thước kernel tối ưu.

3.3.1 Khối Inception

Khối Inception thực hiện nhiều phép tích chập song song (1×1, 3×3, 5×5) và Max Pooling trên cùng một đầu vào, sau đó nối (concatenate) các đầu ra lại.

Đổi mới chính:

1. **Khối Inception:** Cho phép mạng tự động học tập trung vào các đặc trưng ở nhiều quy mô khác nhau.
2. **Tích chập 1×1 (Bottleneck Layer):** Được sử dụng để giảm số lượng kênh (depth) trước các tích chập 3×3 và 5×5 tốn kém, giúp giảm chi phí tính toán đáng kể.

3.4 ResNet (Residual Networks) (2015)

Giải quyết vấn đề suy thoái (degradation problem) khi mạng quá sâu, bằng cách giới thiệu các **Kết nối Tắt (Skip Connections)** hay **Khối Dư (Residual Blocks)**.

3.4.1 Khối Dư (Residual Block)

Một khối dư học một ánh xạ dư $H(x) - x$. Đầu ra của khối là $H(x)$.

$$H(x) = F(x) + x$$

Trong đó $F(x)$ là đầu ra của các lớp tích chập thông thường, và x là đầu vào ban đầu được thêm trực tiếp vào đầu ra (skip connection). Điều này giúp mạng dễ dàng học ánh xạ đồng nhất $H(x) = x$ nếu các lớp bên trong không cần thiết.

```
def residual_block(X, filters, stage, block):
    # Save the original input value
    X_shortcut = X

    # 1. First Conv layer
    X = layers.Conv2D(filters, (3, 3), padding='same')(X)
    X = layers.BatchNormalization(axis=3)(X)
    X = layers.Activation('relu')(X)

    # 2. Second Conv layer
    X = layers.Conv2D(filters, (3, 3), padding='same')(X)
    X = layers.BatchNormalization(axis=3)(X)

    # 3. Add skip connection
    X = layers.Add()([X, X_shortcut])
    X = layers.Activation('relu')(X)
    return X
```

Listing 2: Cấu trúc của Khối Dư cơ bản (TensorFlow/Keras)

3.5 Xception (2017)

Được phát triển bởi Google, dựa trên ý tưởng của Khối Inception, nhưng sử dụng **Separable Convolution** thay thế cho các tích chập tiêu chuẩn.

3.5.1 Depthwise Separable Convolution

Đây là một loại tích chập tách biệt, nó chia tích chập tiêu chuẩn thành hai bước:

1. **Depthwise Convolution:** Áp dụng một kernel $K \times K$ *duy nhất* cho *mỗi* kênh đầu vào. (Ví dụ: 64 kênh đầu vào $\rightarrow$ 64 feature maps, mỗi feature map được tạo bởi 1 kernel).

2. **Pointwise Convolution:** Áp dụng tích chập 1×1 để kết hợp các đầu ra từ bước 1.

Phương pháp này giảm đáng kể số lượng tham số và tính toán, đồng thời cải thiện hiệu suất.

4 Học chuyển giao (Transfer Learning)

Học chuyển giao là phương pháp tận dụng các mạng CNN đã được đào tạo trước (pre-trained models), thường là trên tập dữ liệu rất lớn như ImageNet, và áp dụng chúng cho một tác vụ mới với dữ liệu hạn chế.

4.1 Các bước thực hiện (Ví dụ Keras)

1. **Tải Mô hình Đã được Đào tạo:** Tải mô hình cơ sở (ví dụ: VGG16, ResNet50) và loại bỏ (hoặc thay thế) lớp Dense phân loại cuối cùng.

2. **Đóng băng các lớp cơ sở (Feature Extraction):** Đóng băng trọng số của các lớp tích chập (hoặc chỉ một phần của chúng) để chúng không bị thay đổi trong quá trình huấn luyện.

3. **Thêm lớp phân loại mới:** Thêm các lớp Dense mới cho tác vụ phân loại của riêng bạn.

4. **Huấn luyện mô hình:** Huấn luyện chỉ các lớp Dense mới.

```
# Load pre-trained model on ImageNet
base_model = keras.applications.resnet50.ResNet50(
    weights="imagenet",
    include_top=False,  # Remove the final classification layer
    input_shape=(224, 224, 3)
)

# Freeze the weights of the base model
base_model.trainable = False

# Build a new model (only train Dense layers)
inputs = keras.Input(shape=(224, 224, 3))
x = base_model(inputs, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(1024, activation='relu')(x)
outputs = layers.Dense(10, activation='softmax')(x)  # 10 new classes
model = keras.Model(inputs, outputs)

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model...
# model.fit(...)

```

```
# Fine-tuning: unfreeze and retrain part of the base model layers
# base_model.trainable = True
# model.compile(...)
# model.fit(...)
```

Listing 3: Ví dụ Học Chuyển Giao với ResNet50 trong Keras

5 Các Tác vụ Thị giác Máy tính Sâu

5.1 Phân đoạn Ngữ nghĩa (Semantic Segmentation)

Mục tiêu là phân loại **từng pixel** trong hình ảnh thuộc về lớp đối tượng nào (ví dụ: pixel thuộc về "xe hơi", "đường", "cây cối",...).

5.1.1 Mạng Tích chập Hoàn toàn (Fully Convolutional Networks - FCNs)

Mô hình tiêu biểu cho phân đoạn ngữ nghĩa. FCN thay thế các lớp Dense cuối cùng bằng các lớp Tích chập.

Chuyển đổi Lớp Dense thành Lớp Conv: Một lớp Dense có trọng số W và đầu vào X có thể được coi là một lớp tích chập 1×1 với W là kernel và đầu vào X đã được định hình lại.

Khó khăn kỹ thuật chính: Giảm kích thước (downsampling) qua các lớp Pooling làm mất thông tin chi tiết về vị trí (spatial information), gây khó khăn cho việc phân loại chính xác từng pixel.

5.1.2 Lớp Tích chập Chuyển vị (Transposed Convolutional Layer)

Còn được gọi là **Deconvolution** hoặc **Fractionally Strided Convolution**. Được sử dụng để tăng kích thước không gian (upsampling) của feature maps để khôi phục lại độ phân giải ảnh gốc cho tác vụ phân đoạn.

5.2 Phát hiện Đối tượng (Object Detection)

Tác vụ phức tạp hơn: Vừa phân loại đối tượng, vừa xác định vị trí của chúng bằng cách vẽ các Hộp Giới hạn (Bounding Boxes).

5.2.1 Các Phương pháp Phát hiện Đối tượng

1. **YOLO (You Only Look Once):** Phát hiện đối tượng trong một lần chạy mạng nơ-ron duy nhất, rất nhanh.

2. **Faster R-CNN:** Chia quá trình thành hai bước: (1) Tạo vùng đề xuất (Region Proposal Network - RPN) và (2) Phân loại và tinh chỉnh hộp giới hạn cho từng vùng.

5.3 Phân đoạn Phiên bản (Instance Segmentation)

Thậm chí còn phức tạp hơn Phân đoạn Ngữ nghĩa. Nó không chỉ phân loại pixel mà còn phân biệt các **phiên bản riêng lẻ** của cùng một lớp đối tượng (ví dụ: tách biệt chiếc xe hơi 1 khỏi chiếc xe hơi 2).

6 Thử nghiệm và Khắc phục sự cố CNN

6.1 Xây dựng CNN từ đầu (Ví dụ MNIST)

Khi xây dựng một CNN đơn giản, chúng ta thường làm theo kiến trúc (CONV $\rightarrow$ POOL) lặp lại, theo sau là một phần Dense.

```
model = keras.Sequential([
    # Input: 28x28 grayscale
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dropout(0.5), # Regularization
    layers.Dense(10, activation='softmax')
])
```

Listing 4: Xây dựng CNN đơn giản cho MNIST

6.2 Khắc phục sự cố Mô hình CNN không hội tụ hoặc hiệu suất kém

Nếu CNN không đạt được độ chính xác mong muốn, có thể thử các biện pháp sau:

1. **Tăng cường dữ liệu (Data Augmentation):** Thêm các biến thể của ảnh huấn luyện (xoay, lật, cắt, thay đổi độ sáng,...).
2. **Chuẩn hóa đầu vào (Input Normalization):** Đảm bảo các giá trị pixel được chia tỷ lệ (ví dụ: chia cho 255 để nằm trong $[0, 1]$ hoặc chuẩn hóa theo trung bình/độ lệch chuẩn).
3. **Sử dụng Batch Normalization (BN):** Thêm các lớp BN sau các lớp Conv để ổn định quá trình huấn luyện và cho phép sử dụng tốc độ học cao hơn.
4. **Thay đổi Kiến trúc/Kernel:** Thử các kiến trúc đã được chứng minh (ResNet, VGG) hoặc thay đổi kích thước kernel, số lượng lớp, và lớp gộp.
5. **Tăng cường Regularization:** Sử dụng Dropout, L2 Regularization, hoặc Early Stopping để tránh overfitting.
6. **Sử dụng Học Chuyển Giao (Transfer Learning):** Bắt đầu với một mô hình đã được huấn luyện trước trên tập dữ liệu lớn.

6.3 Tác dụng của Max Pooling so với Conv Layer cùng Stride

- **Max Pooling:** Giảm kích thước không gian và cung cấp một mức độ **bất biến dịch chuyển (translation invariance)** cục bộ (giúp mạng ít nhạy cảm với vị trí chính xác của đặc trưng). Nó không có tham số để học.
- **Conv Layer với Stride lớn:** Giảm kích thước không gian, nhưng **có tham số để học** (kernel weights). Nó giữ lại nhiều thông tin hơn (vì nó là một phép tính có trọng số, không chỉ là phép max). Tuy nhiên, nó có thể không cung cấp sự bất biến dịch chuyển tốt như Max Pooling và thêm chi phí tính toán.

Kết luận: Dùng Max Pooling khi mục đích chính là giảm chiều không gian và tạo sự bất biến cục bộ mà không cần học thêm tham số.

7 Mạng tạo ảnh nghệ thuật (Style Transfer)

Style Transfer là một ứng dụng thú vị của CNNs, cho phép kết hợp nội dung của một bức ảnh (Content Image) với phong cách (màu sắc, họa tiết) của một bức ảnh khác (Style Image).

7.1 Các Thành phần Cơ bản

1. **Mạng đã Huấn luyện trước (Pre-trained Network):** Thường là VGG19, được sử dụng để trích xuất đặc trưng.
2. **Hàm mất mát Nội dung (Content Loss):** Đảm bảo ảnh được tạo ra gần giống với ảnh Nội dung (Content Image).
3. **Hàm mất mát Phong cách (Style Loss):** Đảm bảo ảnh được tạo ra có cùng phong cách với ảnh Phong cách (Style Image).

7.1.1 Ma trận Gram (Gram Matrix) và Style Loss

Style Loss sử dụng Ma trận Gram để đo lường sự tương quan giữa các đặc trưng trên các kênh khác nhau (channels) của feature map. Ma trận Gram G được tính từ feature map F (có C kênh và $H \times W$ không gian) như sau:

$$G_{k,k'} = \sum_{i=1}^{H} \sum_{j=1}^{W} F_{i,j,k} F_{i,j,k'}$$

Trong đó $F_{i,j,k}$ là giá trị của pixel (i, j) ở kênh k. Ma trận Gram G có kích thước $C \times C$.
Hàm mất mát Phong cách (Style Loss) giữa ảnh được tạo G^G và ảnh phong cách G^S tại lớp l:

$$L_{style}^{(l)} = \frac{1}{4C_l^2 M_l^2} \sum_{k} \sum_{k'} (G_{k,k'}^G - G_{k,k'}^S)^2$$

Với C_l là số kênh và M_l là kích thước không gian $(H \times W)$ của lớp l.

7.2 Khởi tạo và Tối ưu hóa

Style Transfer được thực hiện bằng cách **tối ưu hóa trực tiếp các pixel** của ảnh đầu ra (Generated Image) thay vì trọng số của mạng.
Hàm mất mát tổng thể (Total Loss):

$$L_{total} = \alpha L_{content} + \beta L_{style} + \gamma L_{total\_variation}$$

- α và β: Trọng số để cân bằng giữa nội dung và phong cách.
- $L_{total\_variation}$: Một thuật ngữ điều chỉnh hóa bổ sung giúp làm mịn hình ảnh được tạo ra.

8 Các Kiến trúc Hiện đại Khác

8.1 SENet (Squeeze-and-Excitation Networks)

Giới thiệu **Khối SE** (Squeeze-and-Excitation) để cải thiện chất lượng của các feature maps bằng cách cho phép mạng học tập trọng số (importance) cho từng kênh (channel) một cách linh hoạt.

1. **Squeeze (Global Information Embedding):** Sử dụng Global Average Pooling để tổng hợp thông tin không gian (spatial) và thu nhỏ feature map $H \times W \times C$ thành $1 \times 1 \times C$.
2. **Excitation (Adaptive Recalibration):** Sử dụng hai lớp Dense (một lớp giảm chiều, một lớp tăng chiều) và hàm sigmoid để tạo ra một tập hợp các trọng số, mỗi trọng số tương ứng với mức độ quan trọng của một kênh.
3. **Scale (Recalibration):** Nhân trọng số kênh này với feature map ban đầu.

8.2 Mạng Tích chập Dilated (Dilated/Atrous Convolution)

Thay vì sử dụng Pooling để giảm kích thước (downsampling), Tích chập Dilated chèn thêm các **lỗ hổng (gaps)** vào kernel. Điều này giúp tăng trường tiếp nhận hiệu quả (effective receptive field) của kernel mà không mất đi độ phân giải không gian (spatial resolution).

8.2.1 Công thức

Tích chập Dilated được định nghĩa bởi một tham số **tốc độ giãn nở (dilation rate)** r. Khi $r = 1$, nó là tích chập thông thường.

$$O(i, j) = \sum_m \sum_n I(i - r \cdot m, j - r \cdot n)K(m, n)$$

Nó rất hữu ích trong các tác vụ phân đoạn, nơi việc bảo toàn thông tin không gian là tối quan trọng.

9 Tổng kết và Ứng dụng

CNNs đã thay đổi hoàn toàn lĩnh vực Thị giác Máy tính. Các kỹ thuật như Residual Blocks, Inception Modules, Depthwise Separable Convolution, và Transfer Learning là những công cụ không thể thiếu trong xây dựng các mô hình thị giác máy tính hiệu suất cao.

9.1 Các Ứng dụng Chính

1. Nhận dạng và phân loại hình ảnh.
2. Phát hiện và theo dõi đối tượng (ví dụ: xe tự lái).
3. Nhận dạng khuôn mặt.
4. Phân đoạn y tế (ví dụ: phát hiện khối u trong ảnh X-quang/MRI).
5. Tạo ảnh nghệ thuật (Style Transfer, DeepDream).