HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BỘ MÔN THỰC TẬP CƠ SỞ

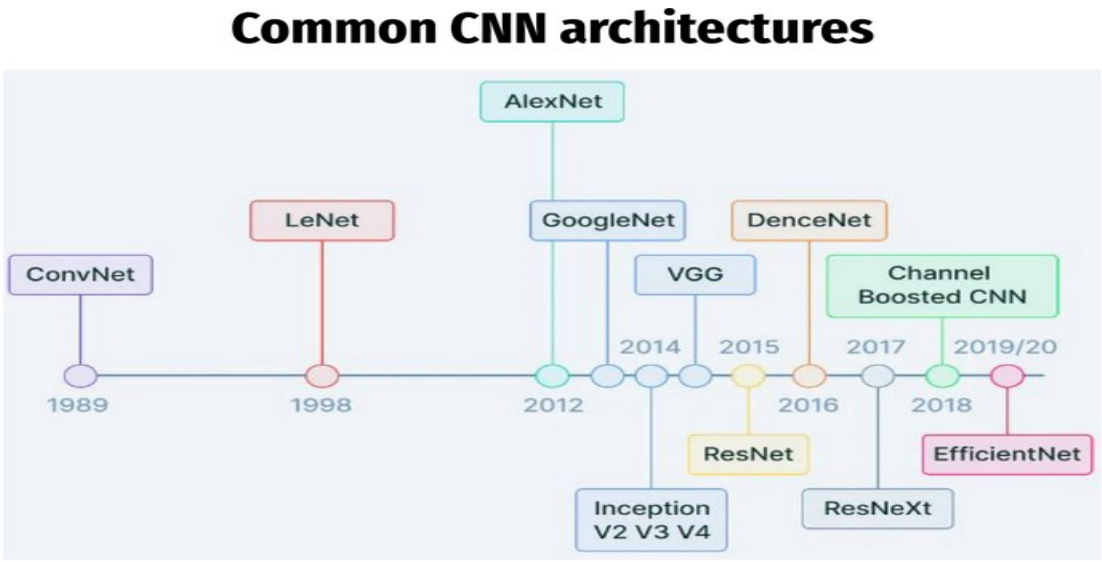
THỰC TẬP CƠ SỞ

Giảng viên hướng dẫn : **KIM NGỌC BÁCH**
Họ và tên sinh viên : **NGUYỄN XUÂN HẢI**
Mã sinh viên : **B22DCCN271**
Lớp : **D22CQCN07-B**
Nhóm : **13**

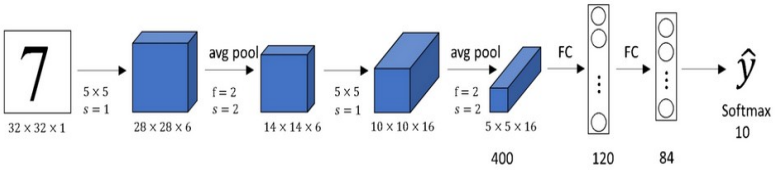
Báo Cáo Hàng Tuần Lần 3 Ngày (23-29/3/2025)

Hà Nội – 2025

BÁO CÁO TIẾN ĐỘ HÀNG TUẦN
I.KIẾN TRÚC MẠNG CNN



1. LeNet(1989)



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

- Để làm cho dữ liệu như vậy tuân theo hồi quy softmax và MLP s, trước tiên chúng ta làm phẳng mỗi hình ảnh từ một ma trận thành một vector chiều dài cố định, và sau đó xử lý chúng với các lớp được kết nối hoàn toàn.
- Là một lợi ích bổ sung của việc thay thế các lớp được kết nối hoàn toàn bằng các lớp kết nối, chúng ta sẽ tận hưởng các mô hình phân tích hơn đòi hỏi ít tham số hơn nhiều.

Các đơn vị cơ bản trong mỗi khối phức tạp là một lớp phức tạp, một chức năng kích hoạt sigmoid, và một hoạt động tổng hợp trung bình tiếp theo. Lưu ý rằng trong khi Relus và max-pooling hoạt động tốt hơn, những khám phá này vẫn chưa được thực hiện trong những năm 1990. Mỗi lớp phức hợp sử dụng một hạt nhân và một hàm kích hoạt sigmoid. Các lớp này ánh xạ các đầu vào sắp xếp không gian cho một số bản đồ tính năng hai chiều, thường làm tăng số lượng kênh. Lớp phức tạp đầu tiên có 6 kênh đầu ra, trong khi thứ hai có 16 kênh. Mỗi hoạt động tổng hợp làm giảm kích thước bằng hệ số thông qua lấy mẫu xuống không gian. Khối phức tạp phát ra một đầu ra với hình dạng được đưa ra bởi (kích thước lô, số kênh, chiều cao, chiều rộng).

- Để truyền đầu ra từ khối phức tạp đến khối dày đặc, chúng ta phải làm phẳng từng ví dụ trong minibatch.

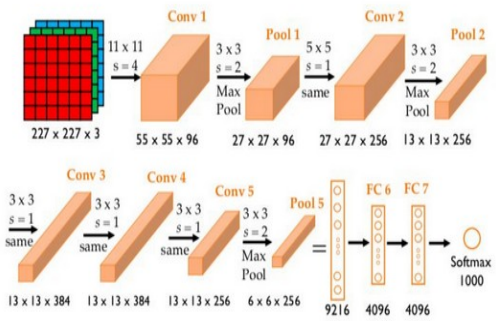
- Nói cách khác, chúng ta lấy đầu vào bốn chiều này và biến nó thành đầu vào hai chiều được mong đợi bởi các lớp được kết nối hoàn toàn: như một lời nhắc nhở, biểu diễn hai chiều mà chúng ta mong muốn sử dụng kích thước đầu tiên để lập chỉ mục các ví dụ trong minibatch và thứ hai để đưa ra biểu diễn vector phẳng of each mỗi example thí dụ. Khối dày đặc của LeNet có ba lớp kết nối hoàn toàn, tương ứng với 120, 84 và 10 đầu ra. Bởi vì chúng ta vẫn đang thực hiện phân loại, lớp đầu ra 10 chiều tương ứng với số lượng lớp đầu ra có thể.

Chúng ta chỉ cần khởi tạo một khối `Sequential` và chuỗi với nhau các lớp thích hợp.

```
import torch
from torch import nn
from d2l import torch as d2l

net = nn.Sequential(
    nn.Conv2d(1, 6, kernel_size=5, padding=2), nn.Sigmoid(),
    nn.AvgPool2d(kernel_size=2, stride=2),
    nn.Conv2d(6, 16, kernel_size=5), nn.Sigmoid(),
    nn.AvgPool2d(kernel_size=2, stride=2),
    nn.Flatten(),
    nn.Linear(16 * 5 * 5, 120), nn.Sigmoid(),
    nn.Linear(120, 84), nn.Sigmoid(),
    nn.Linear(84, 10))
```

2. AlexNet(2012)



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
		-	9216	-	-	
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Những thủ thuật chính để huấn luyện mạng nơ-ron bao gồm khởi tạo tham số dựa trên thực nghiệm, các biến thể tốt hơn của hạ gradient ngẫu nhiên, hàm kích hoạt không ép (*non-squashing activation functions*), và thiếu các kỹ thuật điều chuẩn hiệu quả.

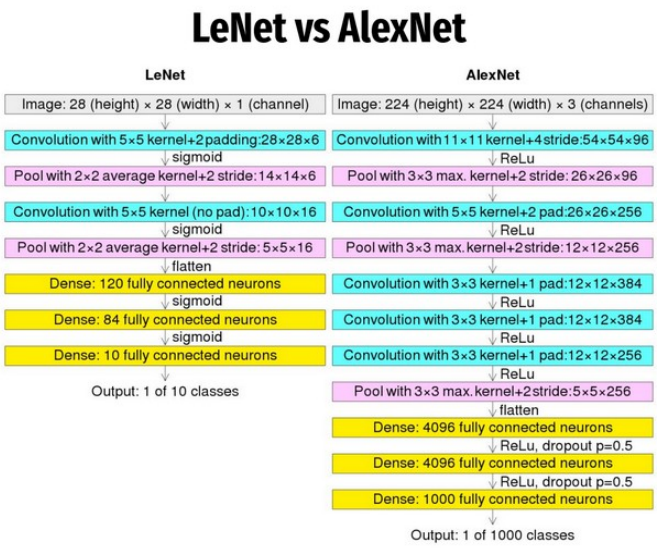
Vì vậy, thay vì huấn luyện các hệ thống *đầu-cuối* (từ điểm ảnh đến phân loại), các pipeline cổ điển sẽ thực hiện các bước sau:

1. Thu thập tập dữ liệu đáng chú ý. Trong những ngày đầu, các tập dữ liệu này đòi hỏi các cảm biến đắt tiền (ảnh có 1 triệu điểm ảnh đã được coi là tối tân nhất vào thời điểm đó).
2. Tiền xử lý tập dữ liệu với các đặc trưng được tạo thủ công dựa trên các kiến thức quang học, hình học, các công cụ phân tích khác và thi thoảng dựa trên các khám phá tình cờ của các nghiên cứu sinh.
3. Đưa dữ liệu qua một bộ trích chọn đặc trưng tiêu chuẩn như SIFT, hoặc SURF, hay bất kỳ pipeline được tinh chỉnh thủ công nào.

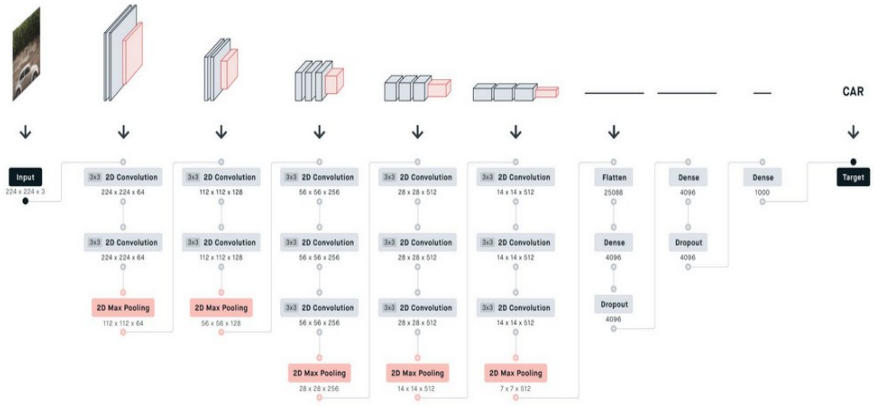
4. Dùng các kết quả biểu diễn để huấn luyện một bộ phân loại ưa thích, có thể là một mô hình tuyến tính hoặc phương pháp hạt nhân.

- Hơn nữa, họ cũng cho rằng để có được độ phức tạp hợp lý, các đặc trưng nên được phân thành thứ lớp với nhiều tầng học cùng nhau, mỗi tầng có các tham số có thể được huấn luyện. Trong trường hợp ảnh, các tầng thấp nhất có thể dùng để phát hiện biên, màu sắc và đường nét. Thật vậy, [Krizhevsky et al., 2012] giới thiệu một biến thể mới của mạng nơ-ron tích chập đã đạt được hiệu năng xuất sắc trong cuộc thi ImageNet.

- Một điều thú vị là ở các tầng thấp nhất của mạng, mô hình đã học được cách trích xuất đặc trưng giống như các bộ lọc truyền thống. Fig. 7.1.1 được tái tạo từ bài báo khoa học trên mô tả các đặc trưng cấp thấp của hình ảnh.



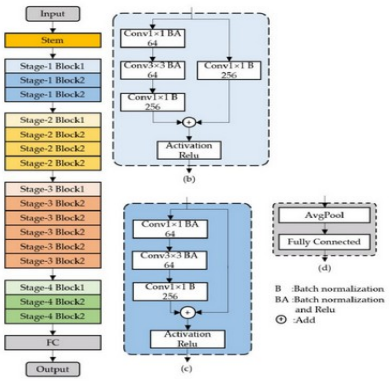
3. VGGNet(2014)

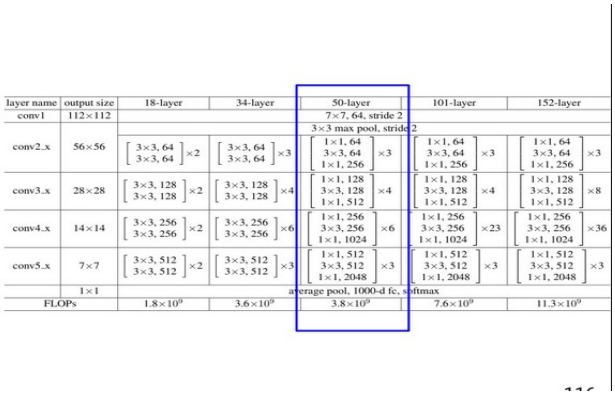


- Khối cơ bản của mạng tích chập cổ điển là một chuỗi các tầng sau đây: (i) một tầng tích chập (với phần đệm để duy trì độ phân giải), (ii) một tầng phi tuyến như ReLU, (iii) một tầng gộp như tầng gộp cực đại. Một khối VGG gồm một chuỗi các tầng tích chập, tiếp nối bởi một tầng gộp cực đại để giảm chiều không gian. Trong bài báo gốc của VGG [Simonyan & Zisserman, 2014], tác giả sử dụng tích chập với các hạt nhân 3×3 và tầng gộp cực đại 2×2 với sải bước bằng 2 (giảm một nửa độ phân giải

sau mỗi khối). Trong mã nguồn dưới đây, ta định nghĩa một hàm tên `vgg_block` để tạo một khối VGG. Hàm này nhận hai đối số `num_convs` và `num_channels` tương ứng lần lượt với số tầng tích chập và số kênh đầu ra.
- Giống như AlexNet và LeNet, mạng VGG có thể được phân chia thành hai phần: phần đầu tiên bao gồm chủ yếu các tầng tích chập và tầng gộp, còn phần thứ hai bao gồm các tầng kết nối đầy đủ. Phần tích chập của mạng gồm các mô-đun `vgg_block` kết nối liên tiếp với nhau. Trong Fig. 7.2.1, biến `conv_arch` bao gồm một danh sách các tuples (một tuple cho mỗi khối), trong đó mỗi tuple chứa hai giá trị: số lượng tầng tích chập và số kênh đầu ra, cũng chính là những đối số cần thiết để gọi hàm `vgg_block`. Mô-đun kết nối đầy đủ giống hệt với mô-đun tương ứng trong AlexNet.
Mạng VGG gốc có 5 khối tích chập, trong đó hai khối đầu tiên bao gồm một tầng tích chập ở mỗi khối, ba khối còn lại chứa hai tầng tích chập ở mỗi khối. Khối đầu tiên có 64 kênh đầu ra, mỗi khối tiếp theo nhân đôi số kênh đầu ra cho tới khi đạt giá trị 512. Vì mạng này sử dụng 8 tầng tích chập và 3 tầng kết nối đầy đủ nên nó thường được gọi là VGG-11.

4.ResNet(2015)

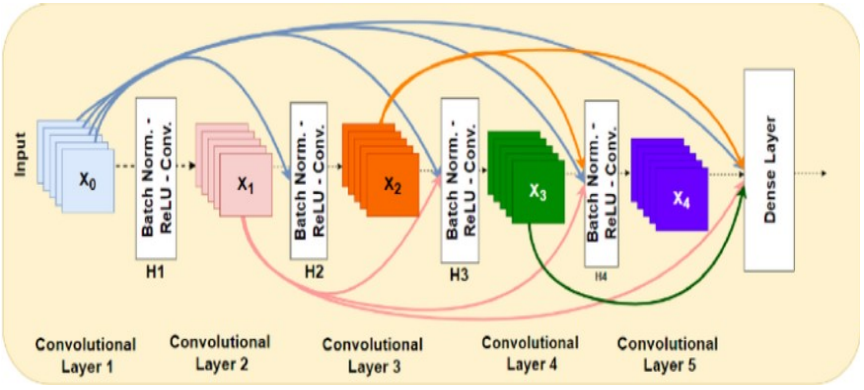




layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Coi F là một lớp các hàm mà một kiến trúc mạng cụ thể (cùng với tốc độ học và các siêu tham số khác) có thể đạt được. Nói cách khác, với mọi hàm số $f \in F$, luôn tồn tại một số tập tham số W có thể tìm được bằng việc huấn luyện trên một tập dữ liệu phù hợp. Giả sử f^* là hàm cần tìm. Sẽ rất thuận lợi nếu hàm này thuộc tập F, nhưng thường không may mắn như vậy. Thay vào đó, ta sẽ cố gắng tìm các hàm số f^* F tốt nhất có thể trong tập F.
Bây giờ, hãy tập trung vào mạng nơ-ron dưới đây. Ký hiệu đầu vào là X. Giả sử ánh xạ lý tưởng muốn học được là *f(x)* , và được dùng làm đầu vào của hàm kích hoạt. Phần nằm trong viền nét đứt bên trái phải khớp trực tiếp với ánh xạ *f(x)* . Điều này có thể không đơn giản nếu chúng ta không cần khối đó và muốn giữ lại đầu vào **x**. Khi đó, phần nằm trong viền nét đứt bên phải chỉ cần tham số hoá *độ lệch* khỏi giá trị **x**, bởi vì ta đã trả về **x** + *f(x)* Trên thực tế, ánh xạ phần dư thường dễ tối ưu hơn, vì chỉ cần đặt *f(x)* =0. Nửa bên phải Fig. 7.6.2 mô tả khối phần dư cơ bản của ResNet. Về sau, những kiến trúc tương tự đã được đề xuất cho các mô hình chuỗi (*sequence model*), sẽ đề cập ở chương sau. Hai tầng đầu tiên của ResNet giống hai tầng đầu tiên của GoogLeNet: tầng tích chập 7 x 7 với 64 kênh đầu ra và sải bước 2, theo sau bởi tầng gộp cực đại 3 x 3 với sải bước 2. Sự khác biệt là trong ResNet, mỗi tầng tích chập theo sau bởi tầng chuẩn hóa theo batch.

5. DenseNet(2016)



Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

- Điểm mấu chốt là khai triển Taylor phân tách hàm số thành các số hạng có bậc tăng dần. Tương tự, ResNet phân tách các hàm số thành $f(x) = x + g(x)$

Cụ thể, ResNet tách hàm số f thành một số hạng tuyến tính đơn giản và một số hạng phi tuyến phức tạp hơn. Nếu ta muốn tách ra thành nhiều hơn hai số hạng thì sao? Một giải pháp đã được đề xuất bởi [Huang et al., 2017] trong kiến trúc DenseNet.

a, Khối Dày Đặc

DenseNet sử dụng kiến trúc "chuẩn hóa theo batch, hàm kích hoạt và phép tích chập" đã qua sửa đổi của ResNet (xem phần bài tập trong Section 7.6). Đầu tiên, ta sẽ lập trình kiến trúc này trong hàm `conv_block`.

```
from d2l import mxnet as d2l
from mxnet import np, npx
from mxnet.gluon import nn
npx.set_np()

def conv_block(num_channels):
    blk = nn.Sequential()
    blk.add(nn.BatchNorm(),
            nn.Activation('relu'),
            nn.Conv2D(num_channels, kernel_size=3, padding=1))
    return blk
```

Một khối dày đặc bao gồm nhiều khối `conv_block` với cùng số lượng kênh đầu ra. Tuy nhiên, ta sẽ nối đầu vào và đầu ra của từng khối theo chiều kênh khi tính toán lượt truyền xuôi.

```
class DenseBlock(nn.Block):
    def __init__(self, num_convs, num_channels, **kwargs):
        super(DenseBlock, self).__init__(**kwargs)
        self.net = nn.Sequential()
        for _ in range(num_convs):
            self.net.add(conv_block(num_channels))

    def forward(self, X):
        for blk in self.net:
            Y = blk(X)
            # Concatenate the input and output of each block on the channel
            # dimension
```

```
            X = np.concatenate((X, Y), axis=1)
        return X
```

Trong ví dụ sau, ta sẽ định nghĩa một khối dày đặc gồm hai khối tích chập với 10 kênh đầu ra. Với một đầu vào gồm 3 kênh, ta sẽ nhận được một đầu ra với $3 + 3 \times 10 = 23$ kênh. Số lượng kênh của khối tích chập kiểm soát sự gia tăng của số lượng kênh đầu ra so với số lượng kênh đầu vào. Số lượng kênh này còn được gọi là tốc độ tăng trưởng (*growth rate*).

b, Tầng Chuyển Tiếp

Mỗi khối dày đặc sẽ làm tăng thêm số lượng kênh. Nhưng việc thêm quá nhiều kênh sẽ tạo nên một mô hình phức tạp quá mức. Do đó, một tầng chuyển tiếp sẽ được sử dụng để kiểm soát độ phức tạp của mô hình. Tầng này dùng một tầng tích chập 1 x 1 để giảm số lượng kênh, theo sau là một tầng gộp trung bình với sải bước bằng 2 để giảm một nửa chiều cao và chiều rộng, từ đó giảm độ phức tạp của mô hình hơn nữa.

Ta sẽ áp dụng một tầng chuyển tiếp với 10 kênh lên đầu ra của khối dày đặc trong ví dụ trước. Việc này sẽ làm giảm số lượng kênh đầu ra xuống còn 10, đồng thời làm giảm đi một nửa chiều cao và chiều rộng.

c, Mô hình DenseNet

Tiếp theo, ta sẽ xây dựng một mô hình DenseNet. Đầu tiên, DenseNet sử dụng một tầng tích chập và một tầng gộp cực đại như trong ResNet.

```
net = nn.Sequential()
net.add(nn.Conv2D(64, kernel_size=7, strides=2, padding=3),
        nn.BatchNorm(), nn.Activation('relu'),
        nn.MaxPool2D(pool_size=3, strides=2, padding=1))
```

Sau đó, tương tự như cách ResNet sử dụng bốn khối phần dư, DenseNet cũng dùng bốn khối dày đặc. Và cũng giống như ResNet, ta có thể tùy chỉnh số lượng tầng tích chập được sử dụng trong mỗi khối dày đặc. Ở đây, ta sẽ đặt số lượng khối tích chập bằng 4 để giống với kiến trúc ResNet-18 trong phần trước. Ngoài ra, ta đặt số lượng kênh (tức tốc độ tăng trưởng) của các tầng tích chập trong khối dày đặc là 32, vì vậy 128 kênh sẽ được thêm vào trong mỗi khối dày đặc.

- Trong ResNet, chiều cao và chiều rộng được giảm sau mỗi khối bằng cách sử dụng một khối phần dư với sải bước bằng 2. Ở đây, ta sẽ sử dụng tầng chuyển tiếp để làm giảm đi một nửa chiều cao, chiều rộng và số kênh.

```
# Num_channels: the current number of channels
num_channels, growth_rate = 64, 32
num_convs_in_dense_blocks = [4, 4, 4, 4]

for i, num_convs in enumerate(num_convs_in_dense_blocks):
    net.add(DenseBlock(num_convs, growth_rate))
    # This is the number of output channels in the previous dense block
    num_channels += num_convs * growth_rate
    # A transition layer that haves the number of channels is added between
    # the dense blocks
    if i != len(num_convs_in_dense_blocks) - 1:
        num_channels //= 2
```

```
        net.add(transition_block(num_channels))
```

Tương tự như ResNet, một tầng gộp toàn cục và một tầng kết nối đầy đủ sẽ được thêm vào cuối mạng để tính toán đầu ra.

```
net.add(nn.BatchNorm(),
        nn.Activation('relu'),
        nn.GlobalAvgPool2D(),
        nn.Dense(10))
```