Depth Estimation from Single Image Using CNN-Residual Network

Tăng Hoài Duy và Phạm Hà Văn Đông

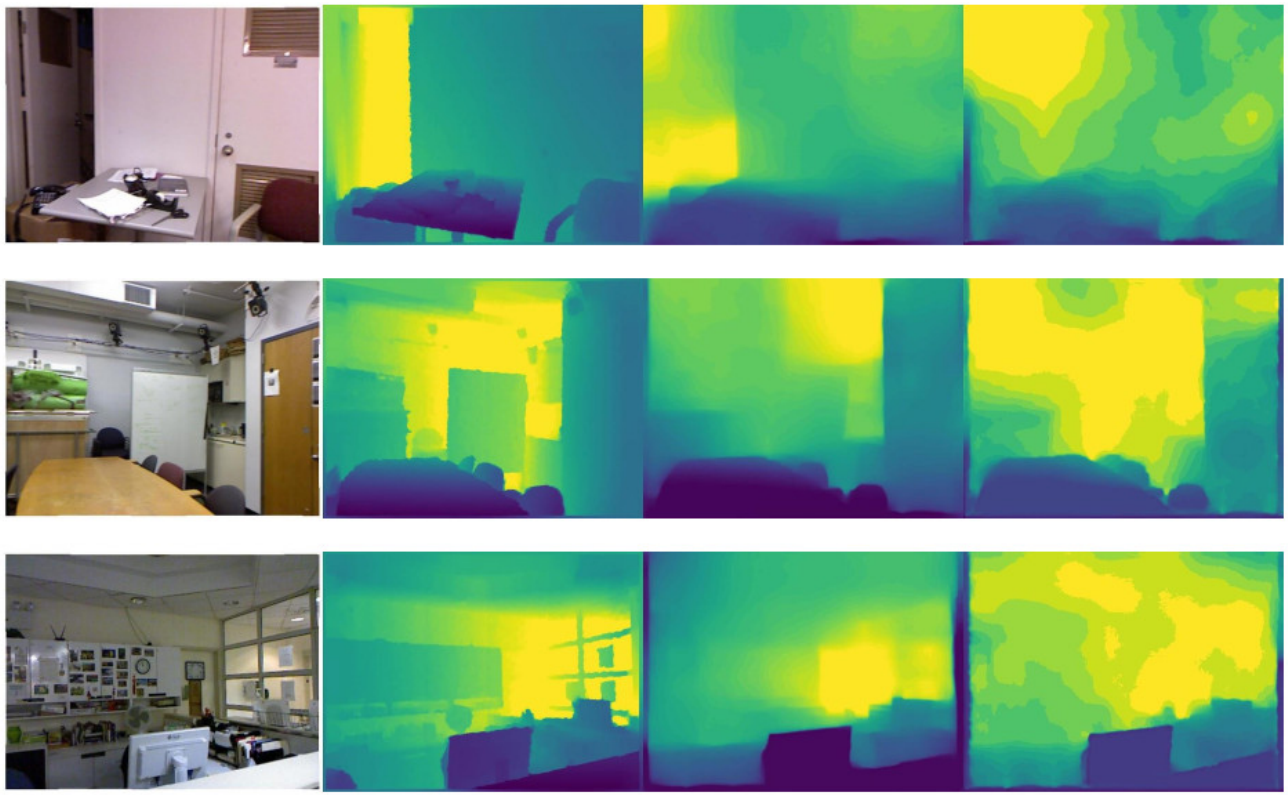
Đại Học Công Nghiệp Thành Phố Hồ Chí Minh

Thị Giác Máy Tính
Ngày 15 tháng 10 năm 2022

Depth Estimation

Giới Thiệu

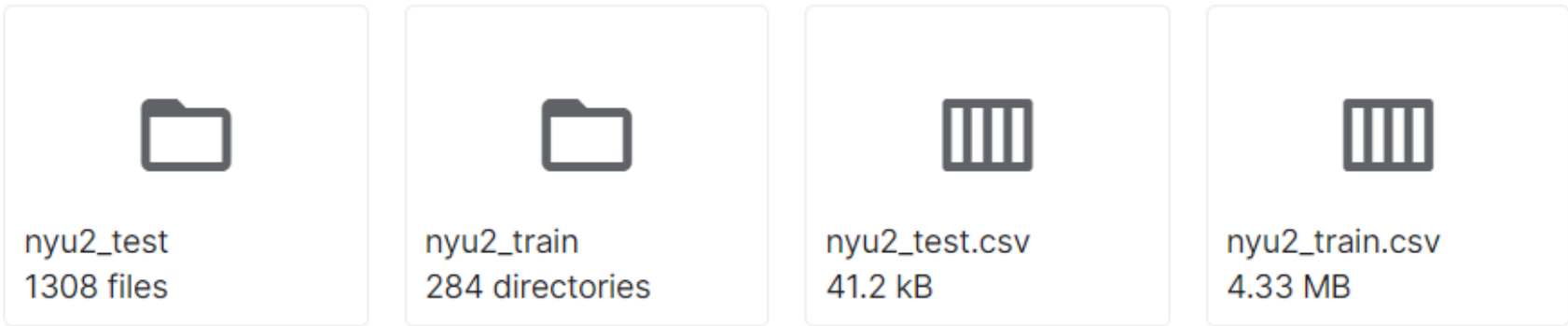
Depth estimation là bài toán ước lượng độ sâu của từng pixel so với camera. Tức là pixel nào càng gần camera thì tại ảnh đầu ra pixel đó có màu đậm và ngược lại. Mạng nơ-ron (CNN) đã được sử dụng để tìm hiểu các mối quan hệ giữa các pixel màu và độ sâu.



Hình: Mô tả bài toán

Depth Estimation

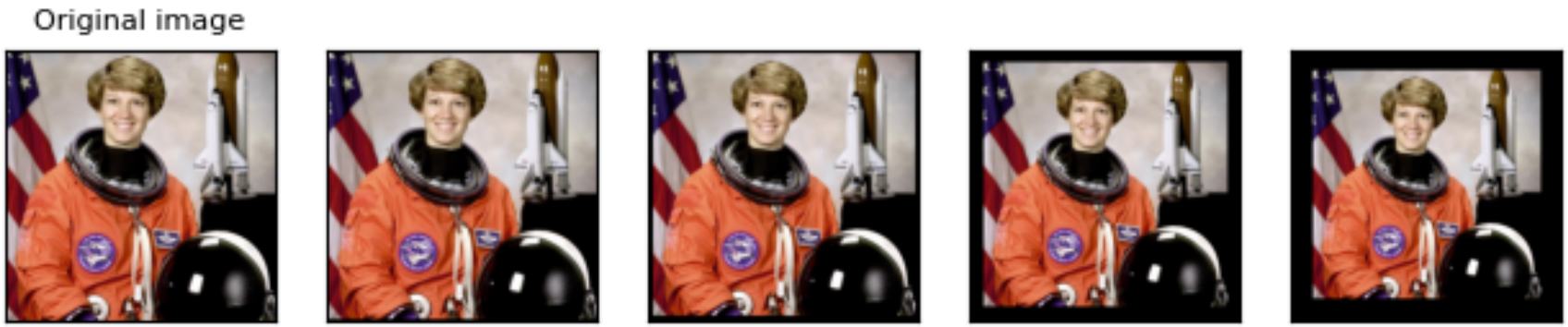
Tập Dữ Liệu



Hình: Mô tả tập dữ liệu NYU Depth V2

Tiền Xử Lí

- Vì kích thước ảnh lớn hơn so với kích thước input của mô hình
=> Thay đổi kích thước đầu vào.
- Pytorch có hỗ trợ module torchvision.transforms: để biến đổi và tăng cường hình ảnh.

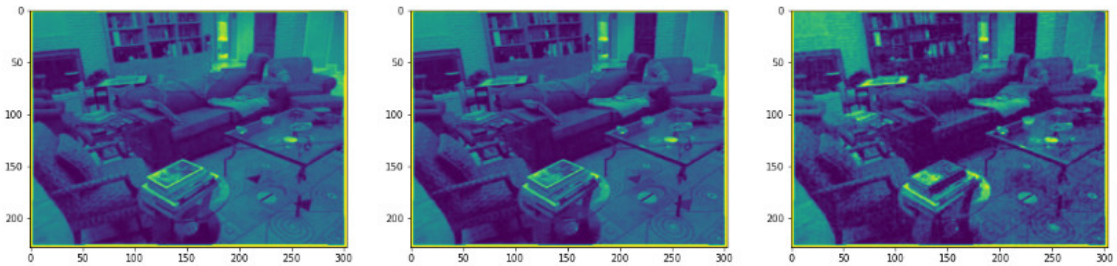


Hình: Transforms Image

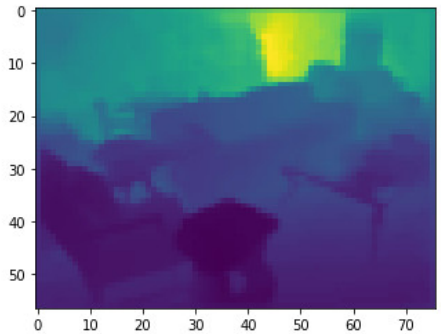
DataLoader

Sử dụng module torch.utils.data.Dataset để load dữ liệu vào mô hình.

```
Feature batch shape: torch.Size([32, 3, 228, 304])
Labels batch shape: torch.Size([32, 1, 57, 76])
```



```
4]:
<matplotlib.image.AxesImage at 0x7f28f5cd7490>
```



Hình: DataLoader

Depth Estimation

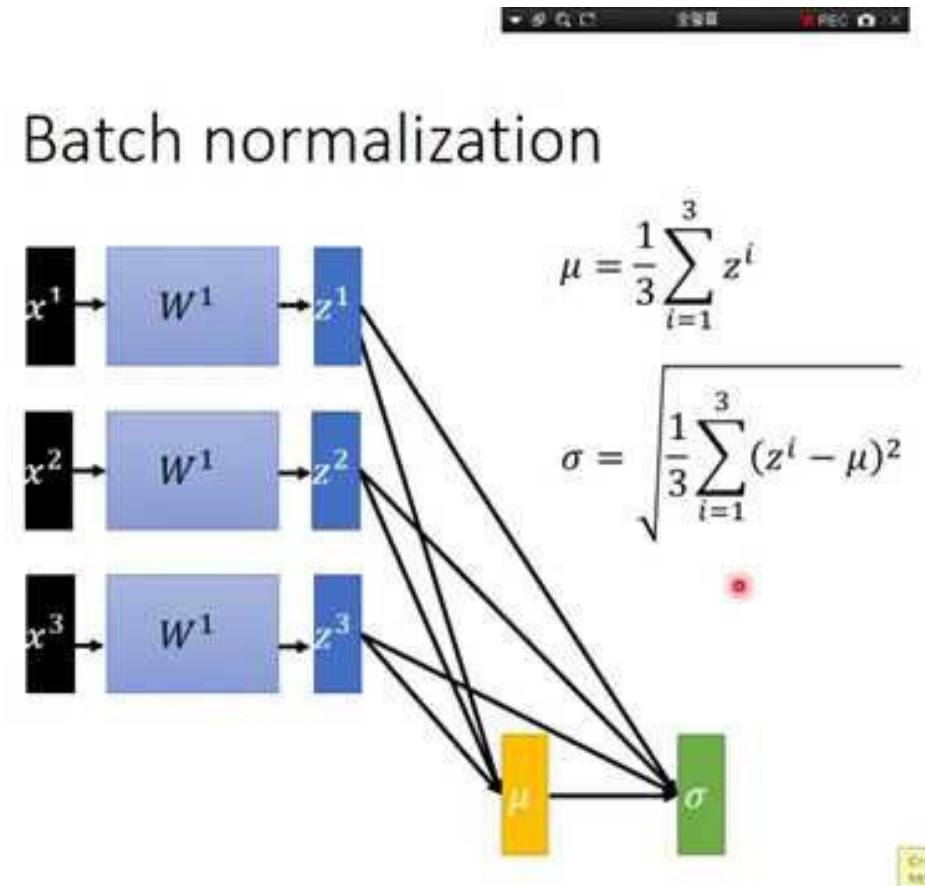
Mô Hình

```
        ReLU-155          [-1, 512, 8, 10]               0
      Conv2d-156          [-1, 512, 8, 10]       2,359,296
 BatchNorm2d-157          [-1, 512, 8, 10]           1,024
        ReLU-158          [-1, 512, 8, 10]               0
      Conv2d-159         [-1, 2048, 8, 10]       1,048,576
 BatchNorm2d-160         [-1, 2048, 8, 10]           4,096
        ReLU-161         [-1, 2048, 8, 10]               0
  Bottleneck-162         [-1, 2048, 8, 10]               0
      Conv2d-163          [-1, 512, 8, 10]       1,048,576
 BatchNorm2d-164          [-1, 512, 8, 10]           1,024
        ReLU-165          [-1, 512, 8, 10]               0
      Conv2d-166          [-1, 512, 8, 10]       2,359,296
 BatchNorm2d-167          [-1, 512, 8, 10]           1,024
        ReLU-168          [-1, 512, 8, 10]               0
      Conv2d-169         [-1, 2048, 8, 10]       1,048,576
 BatchNorm2d-170         [-1, 2048, 8, 10]           4,096
        ReLU-171         [-1, 2048, 8, 10]               0
  Bottleneck-172         [-1, 2048, 8, 10]               0
      Conv2d-173         [-1, 1024, 8, 10]       2,098,176
 BatchNorm2d-174         [-1, 1024, 8, 10]           2,048
      Conv2d-175         [-1, 512, 16, 20]      13,107,712
        ReLU-176         [-1, 512, 16, 20]               0
      Conv2d-177         [-1, 256, 16, 20]       1,179,904
      Conv2d-178         [-1, 256, 16, 20]       6,553,856
     upmodel-179         [-1, 512, 16, 20]               0
      Conv2d-180         [-1, 256, 32, 40]       3,277,056
        ReLU-181         [-1, 256, 32, 40]               0
      Conv2d-182         [-1, 128, 32, 40]         295,040
      Conv2d-183         [-1, 128, 32, 40]       1,638,528
     upmodel-184         [-1, 256, 32, 40]               0
      Conv2d-185         [-1, 128, 64, 80]         819,328
        ReLU-186         [-1, 128, 64, 80]               0
      Conv2d-187          [-1, 64, 64, 80]          73,792
      Conv2d-188          [-1, 64, 64, 80]         409,664
     upmodel-189         [-1, 128, 64, 80]               0
      Conv2d-190           [-1, 1, 62, 78]           1,153
        ReLU-191           [-1, 1, 62, 78]               0
================================================================
Total params: 52,964,289
Trainable params: 29,456,257
Non-trainable params: 23,508,032
----------------------------------------------------------------
Input size (MB): 0.79
Forward/backward pass size (MB): 441.37
Params size (MB): 202.04
Estimated Total Size (MB): 644.20
----------------------------------------------------------------
```

Hình: Mô Hình Dữ Liệu

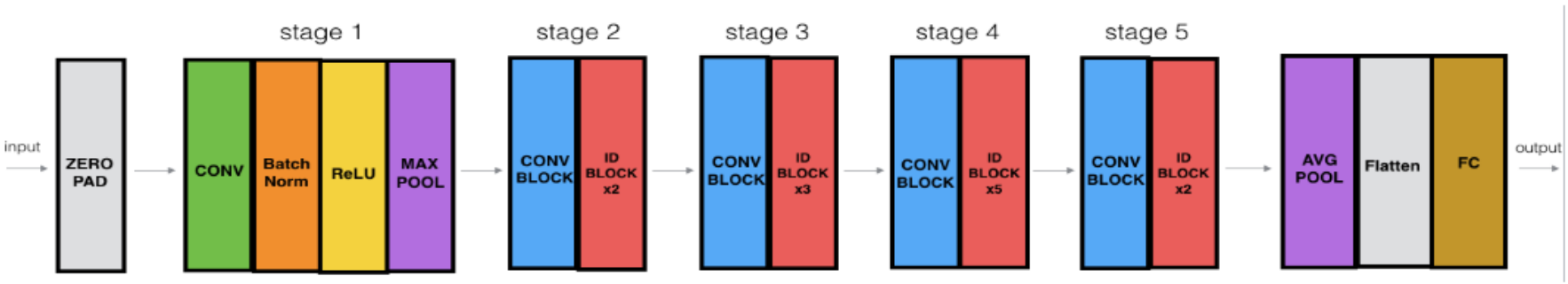
Batch Normalization

Batch Normalization là chuẩn hóa lại dữ liệu sau khi qua cấc lớp tích chập 1 hoặc nhiều layer.
Khi sử dụng batch norm, các giá trị trung bình và độ lệch chuẩn được tính toán đối với batch tại thời điểm áp dụng chuẩn hóa (normalization).



ResNet50

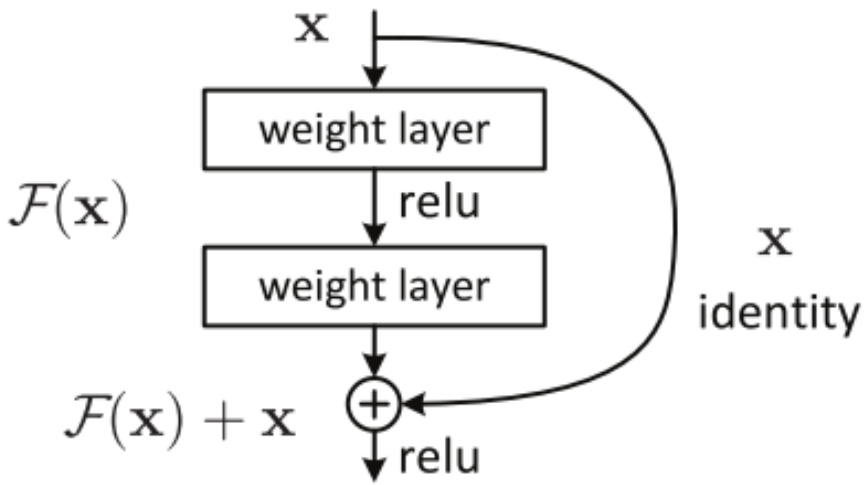
Sử dụng các skip connection để ánh xạ đầu vào từ những layer trước đó tới những layer sau



Hình: ResNet50

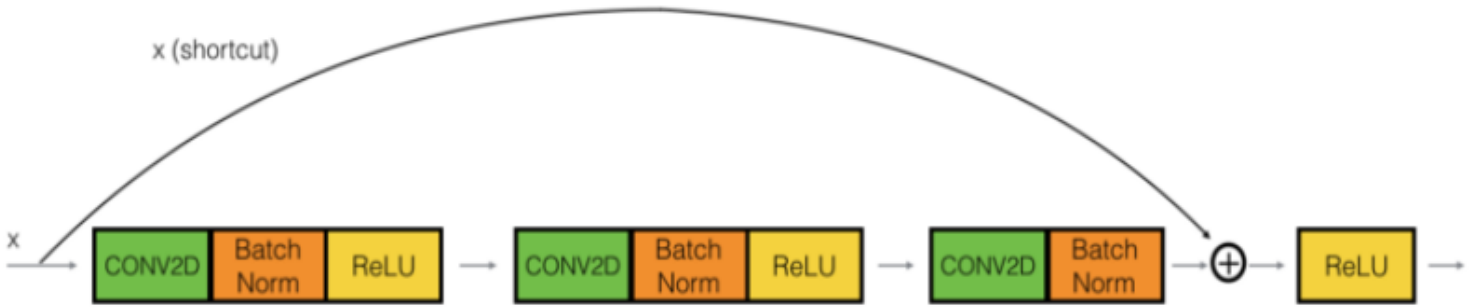
Ưu điểm của ResNet50

Mạng ResNet là một mạng CNN được thiết kế để làm việc với hàng trăm lớp. Một vấn đề xảy ra khi xây dựng mạng CNN với nhiều lớp chập sẽ xảy ra hiện tượng Vanishing Gradient dẫn tới quá trình học tập không tốt.
=> Cho nên giải pháp mà ResNet đưa ra là sử dụng skip connection đồng nhất để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một Residual Block, như trong hình sau :



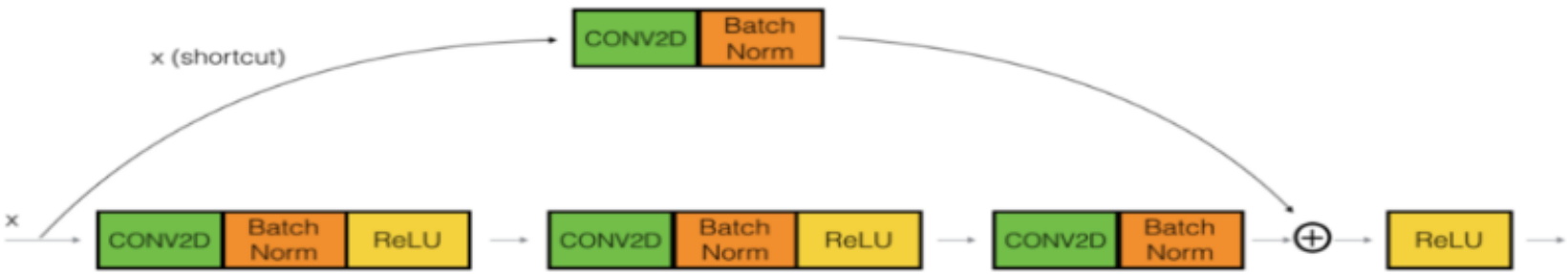
Hình: ResNet50

Identity Block



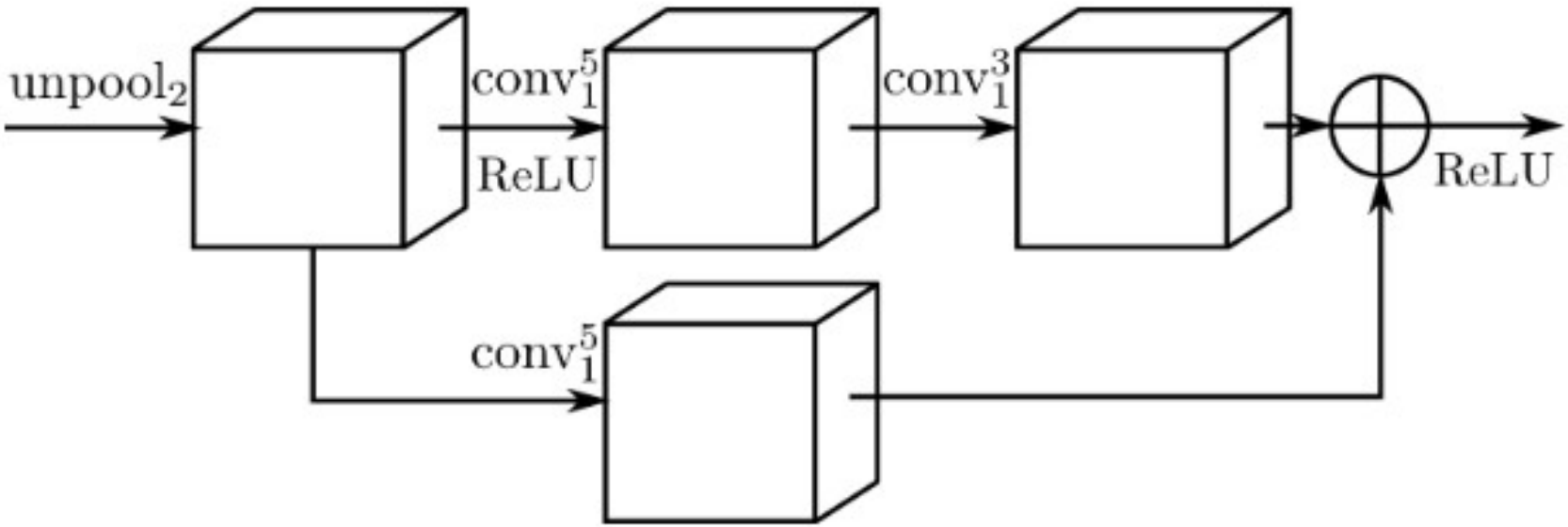
Hình: Identity Block

Convolutional Block



Hình: Convolutional Block

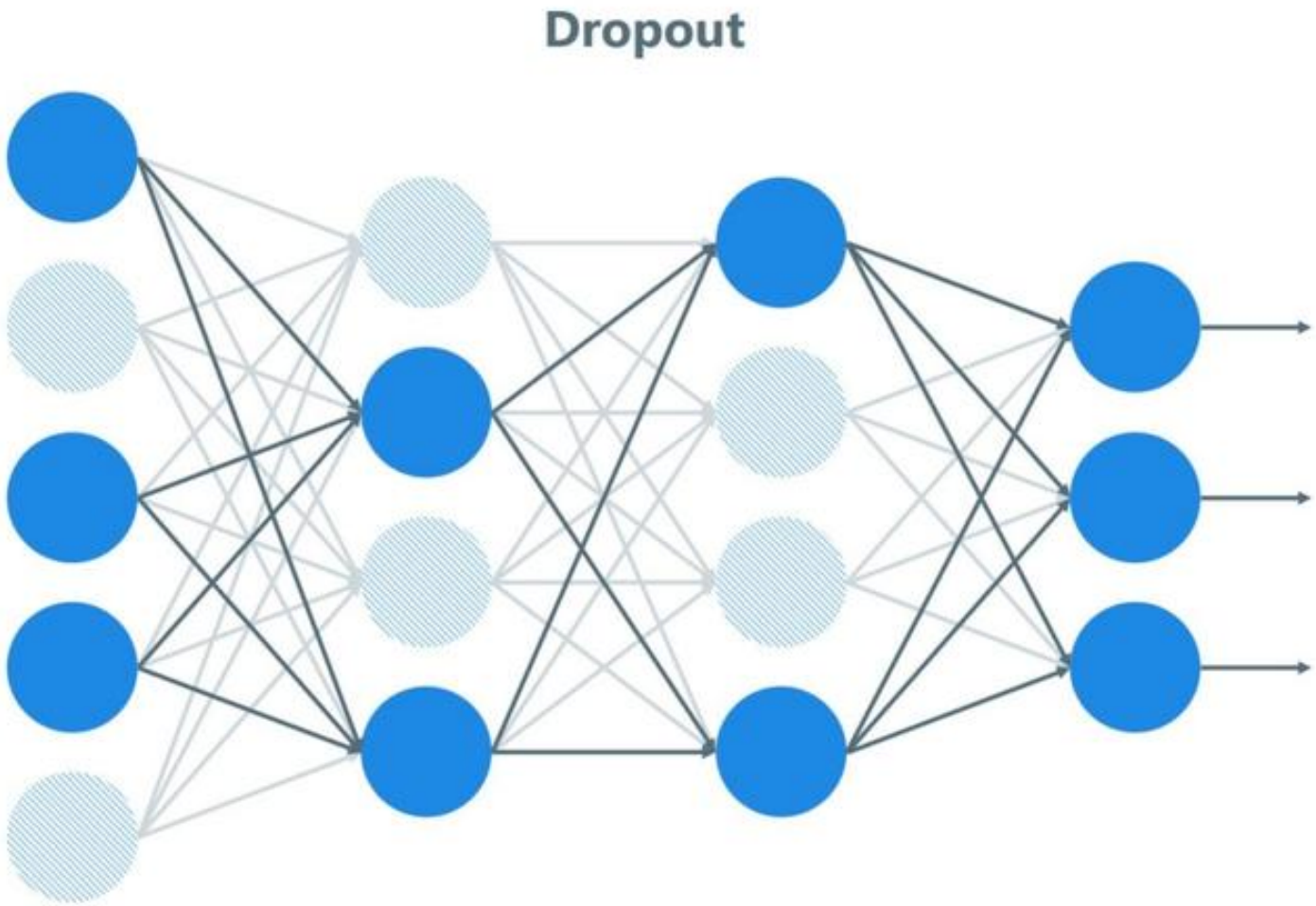
Up-projection



Hình: Up-projection

=> Tăng gấp đôi kích thước ảnh, giảm 1 nữa số layer

Dropout(Dùng trong CNN+FC)

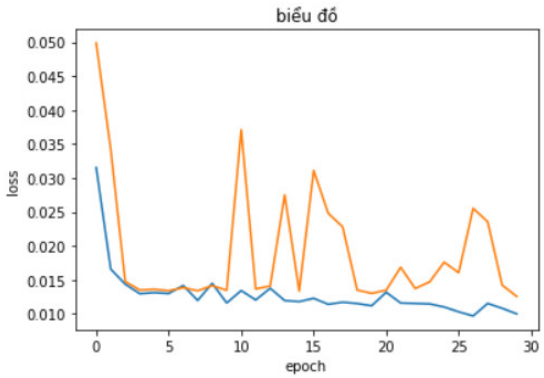


=> Loại bỏ ngẫu nhiên 1 số lượng node trên mỗi layer → tránh overfit

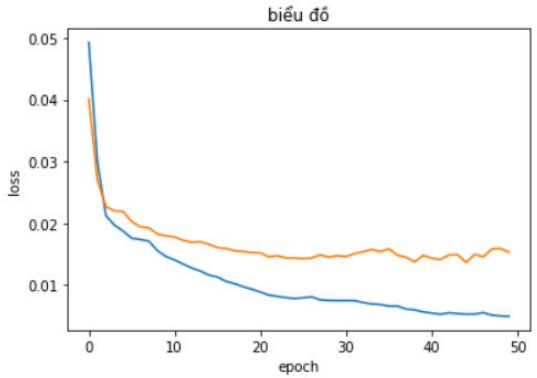
Depth Estimation

Đánh Giá Kết Quả (Train Loss)

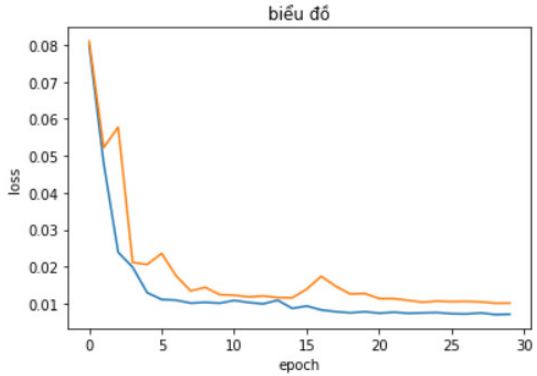
[CNN+FC]



[CNN]



[CNN+Residual]



Hình: Train Loss

Depth Estimation

Kết Luận

	$t < 1.25$	Abs rel diff	RMSE
CNN+FC	0.307	3.4e5	1.12
CNN	0.195	1.2e6	1.22
CNN+trans	0.143	3.2e5	1.23
CNN+Res	0.634	2.35e4	1.74

Table 1. Metric evaluation results

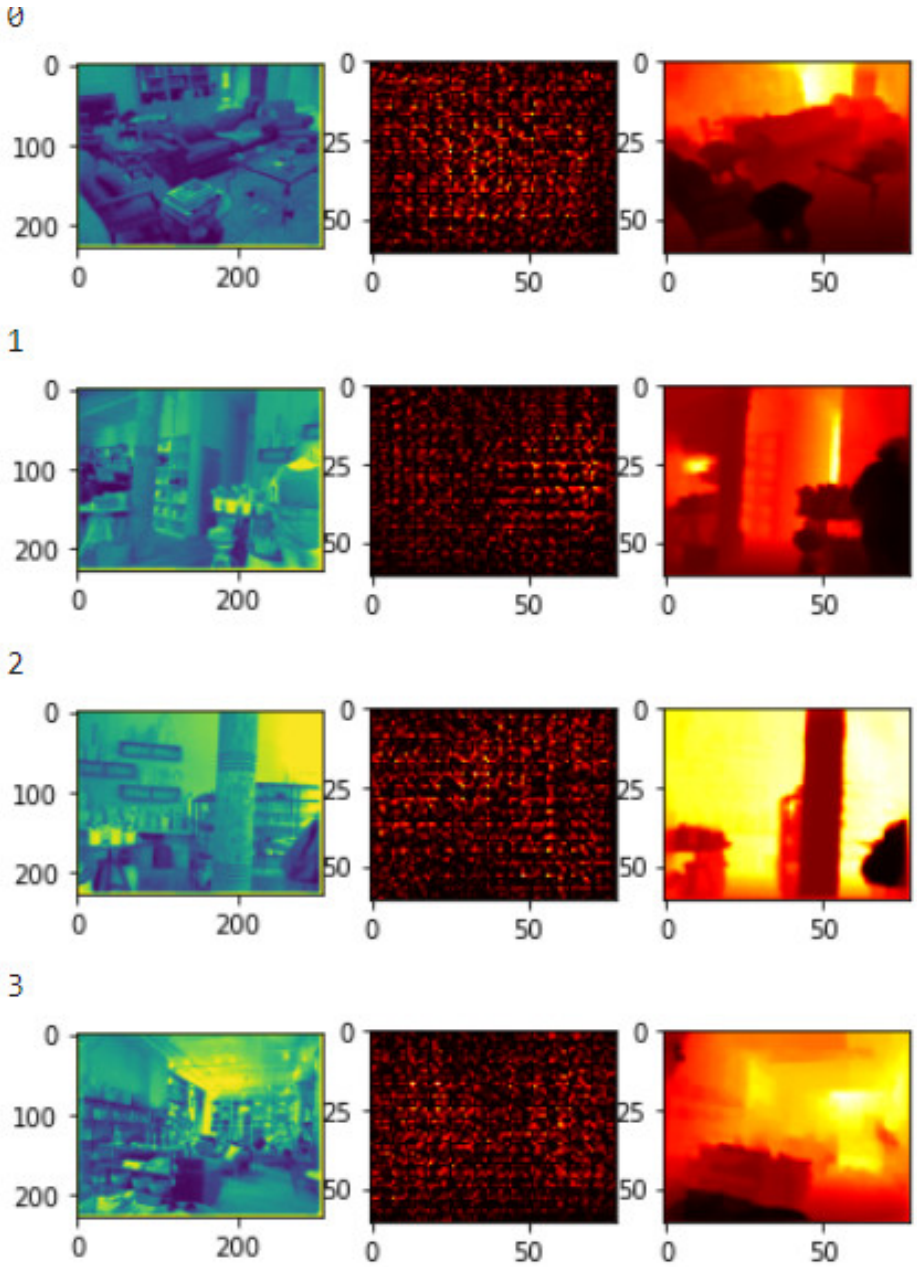
```
print(output.min(),output.max())
```

```
tensor(0., device='cuda:0', grad_fn=<MinBackward1>) tensor(0.5762, device='cuda:0', grad_fn=<MaxBackward1>)
```

```
print(output.min(),output.max())
```

```
tensor(0., device='cuda:0', grad_fn=<MinBackward1>) tensor(0.6944, device='cuda:0', grad
_fn=<MaxBackward1>)

tensor(0., device='cuda:0', grad_fn=<MinBackward1>) tensor(2.8364, device='cuda:0', grad
_fn=<MaxBackward1>)
```


0
0
100
200
0
200
0
25
50
0
50
0
25
50
0
50
1
0
100
200
0
200
0
25
50
0
50
0
25
50
0
50
2
0
100
200
0
200
0
25
50
0
50
0
25
50
0
50
3
0
100
200
0
200
0
25
50
0
50
0
25
50
0
50