Image and Video Computing #2 - Report

1. Cifar10

   (a) **Implement BaseNet**

   Implement the default BaseNet. My 'print(net)' is as it looks.

```
BaseNet(
  (conv1): Sequential(
    (0): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
    (1): BatchNorm2d(6, eps=1e-05, momentum=0.1, affine=True,
    ↪  track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
    ↪  ceil_mode=False)
  )
  (conv2): Sequential(
    (0): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
    ↪  track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
    ↪  ceil_mode=False)
  )
  (fc1): Sequential(
    (0): Linear(in_features=400, out_features=200, bias=True)
    (1): BatchNorm1d(200, eps=1e-05, momentum=0.1, affine=True,
    ↪  track_running_stats=True)
    (2): ReLU()
  )
  (fc2): Linear(in_features=200, out_features=10, bias=True)
)
```

March 6, 2025

I increased the epoch to 30. The Final accuracy on the validation set is as follows:

```
Accuracy of the final network on the val images: 70.1 %
Accuracy of airplane : 72.9 %
Accuracy of automobile : 87.7 %
Accuracy of  bird : 50.8 %
Accuracy of   cat : 49.9 %
Accuracy of  deer : 67.5 %
Accuracy of   dog : 61.9 %
Accuracy of  frog : 80.9 %
Accuracy of horse : 73.5 %
Accuracy of  ship : 84.7 %
Accuracy of truck : 70.9 %
```

(b) **Improve BaseNet**

My Best Model

Table 1: My Model Architecture

Layer No.	Sequential	Layer Type	Kernel Size	Input Dim	Output Dim	Input Channels	Output Channels	Padding
1	conv1	conv2d	3	32	32	3	32	1
2	conv1	relu	-	32	32	32	32	-
3	conv1	conv2d	3	32	32	32	64	1
4	conv1	relu	-	32	32	64	64	-
5	conv1	maxpool2d	2	32	16	64	64	0
6	conv2	conv2d	3	16	16	64	128	1
7	conv2	relu	-	16	16	128	128	-
8	conv2	conv2d	3	16	16	128	128	1
9	conv2	relu	-	16	16	128	128	-
10	conv2	maxpool2d	2	16	8	128	128	0
11	conv3	conv2d	3	8	8	128	256	1
12	conv3	relu	-	8	8	256	256	-
13	conv3	conv2d	3	8	8	256	256	1
14	conv3	relu	-	8	8	256	256	-
15	conv3	maxpool2d	2	4	4	256	256	0
16	fc1	Linear	-	1	1	256	512	-
17	fc1	relu	-	1	1	512	512	-
18	fc2	Linear	-	1	1	512	256	-
19	fc2	relu	-	1	1	256	256	-
20	fc3	Linear	-	1	1	256	128	-
21	fc3	relu	-	1	1	128	128	-
22	fc4	Linear	-	1	1	128	10	-

My improved model has the validation accuracy 87%. Changed the batch nomaralization, dropout, and number of linear layers for the fine tuning. I set Batch Normalization 2d for the conv2d layers and Batch Normalization 1d for the Linear fully connected layers. Set the dropout rate as 0.4.

Table 2: Ablation Cases Results

Experiment	**BatchNorm**	**Dropout**	**FC Layers**	**Accuracy (%)**
My Improved BaseNet (Default Model)	Yes	YES	4	87
No BatchNorm	No	YES	4	86
No Dropout	YES	NO	4	88
FC2 (1 Layer Removed)	YES	YES	2	88
FC1 (2 Layer Removed)	YES	YES	1	87

My ablation study table is as follows:

From the 5 models, the best model is when there are no dropout in the architecture. The accuracy is around 88%(87.9%).

The training loss plot and test accuracy plot for my final modes are as below:

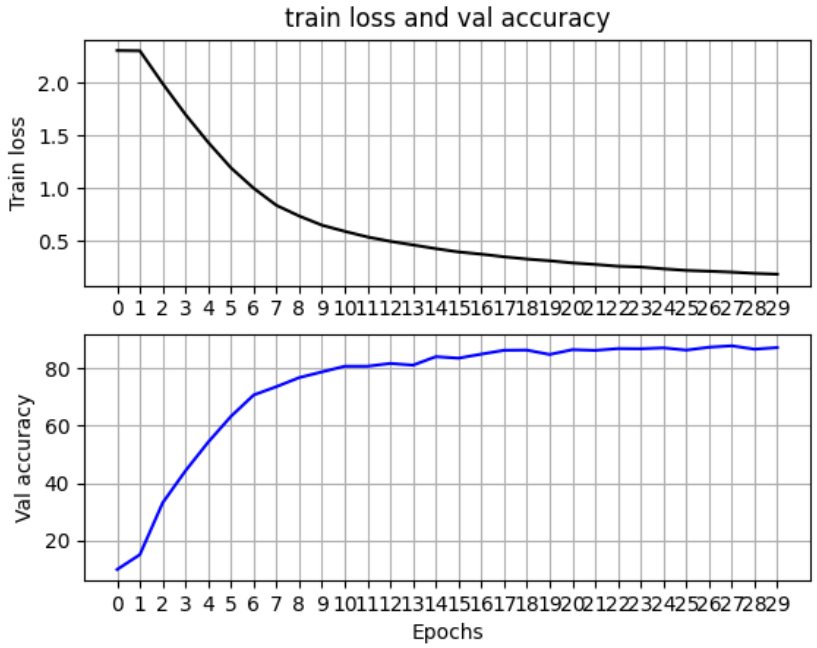

Figure 1: train loss and val accuracy

(c) **Secret test set**

Include Test Accuracy for the best model From the grade scope, my question 1 test accuracy is :

Table 3: Evaluation Q2 Test

Test set Accuracy	Score
Accuracy	85.85000000000001

2. Surface normal

(a) **Implement training cycle**

Implemented code in the Juypter Notebook. To check the validation accuracy, I used the 'simple_predict()' function that provided for the validation check of the model.

(b) **Build on top of ImageNet pre-trained Model**

My build on pre-trained ResNet Model architecture is as follows: I removed the top two layers(fully connected and pooling layers) and add Upsampling and convolution network to reduce the dimensions.

```
  MyModel(
(my_model): Sequential(
  (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3,
  ↪  3), bias=False)
  (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
  ↪  track_running_stats=True)
  (2): ReLU(inplace=True)
  (3): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
  ↪  ceil_mode=False)
  (4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
```

```
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
    )
  )
  (5): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2),
      ↪  padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2),
        ↪  bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
        ↪  track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
      (relu): ReLU(inplace=True)
```

```
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
    )
  )
  (6): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2),
      ↪  padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2),
        ↪  bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
        ↪  track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
    )
  )
  (7): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2),
      ↪  padding=(1, 1), bias=False)
```

```
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2),
        ↪  bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
        ↪  track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
      ↪  padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
      ↪  track_running_stats=True)
    )
  )
)
(upsample): Sequential(
  (0): Upsample(scale_factor=32.0, mode='bilinear')
  (1): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1),
  ↪  padding=(1, 1))
  (2): ReLU(inplace=True)
  (3): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1),
  ↪  padding=(1, 1))
  (4): ReLU(inplace=True)
  (5): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1),
  ↪  padding=(1, 1))
  (6): ReLU(inplace=True)
  (7): Conv2d(64, 3, kernel_size=(1, 1), stride=(1, 1))
)
```

```
)
```

The Final performance of the validation Matrices:

```
Validation loss (L1): 0.253197053745389
Validation metrics: Mean 32.8, Median 26.8, 11.25deg 26.8, 22.5deg
↪  44.2, 30deg 54.1
```

(c) **Increase your model output resolution**

Between the two options of **Atrous convolution** and **Building a learned upsampler**, I decided to use a learned upsampler. Based on the U-net structure with pretrained ResNet18 weights, I implemented the upsampling and decoding processes using the convTranspose2d function and conv2d to establish the skip connections.

I analyzed how modifying loss function parameters and skip connection configurations impact model performance in an ablation study. The loss function has two components: L1 loss for pixel-wise differences and cosine similarity loss for alignment. I experimented with four settings: baseline, increased emphasis on cosine similarity, increased focus on L1 loss, and amplified balanced version. I hope these variations allow the model to compare the benefits of prioritizing angular alignment or absolute differences.

I explore skip connections, which retain spatial details by propagating low-level feature information. The baseline model has all skip connections. We remove early skip connections (Skip1) to assess low-level feature retention. I also remove deeper skip connections (Skip4) to see if high-level features affect reconstruction. Finally, I reduce skip connections to analyze if fewer connections maintain model performance. By comparing validation matrices, I aim to find the optimal balance between loss function design and architectural modifications for high-quality surface normal predictions.

From the alpha and Beta variation, the higher vales have a better result. I can interpret this as that my model is having too much overfitting in the mode. So I will keep the high alpha and beta variation to test the model structure.

From these 4 results, I figured that omitting 2, 3th layer skip connection leads to the best results.

(d) **Visualize your prediction**

Table 4: Ablation Study - Loss Function Variations

Loss Function	Mean AE (°) ↓	Median AE (°) ↓	11.25° (%) ↑	22.5° (%) ↑	30° (%) ↑
Baseline ($\alpha = 1.0, \beta = 1.0$)	42.5	38.6	15.1	31.3	40.5
($\alpha = 0.5, \beta = 1.5$)	43.5	39.8	13.7	30.1	39.3
($\alpha = 1.5, \beta = 0.5$)	41.7	38.1	14.9	31.1	40.6
($\alpha = 2.0, \beta = 2.0$)	40.0	36.1	14.8	32.1	42.3
($\alpha = 5.0, \beta = 5.0$)	38.0	34.4	17.4	34.9	44.8

Table 5: Ablation Study - Skip Connection Variants fixed with alpha =5, beta = 5

Skip Connection	Mean AE ↓	Median AE ↓	11.25° (%) ↑	22.5° (%) ↑	30° (%) ↑
Baseline (Full U-Net)	38.0	34.4	17.4	34.9	44.8
Removing (Skip1)	37.4	32.9	19.4	36.9	46.6
Removing (Skip4)	38.1	34.1	18.6	35.3	44.9
Keep Only Skip2,3	35.9	32.2	15.9	35.3	46.9

I used my own room and current dorm room images to test on my model. Below is my results.

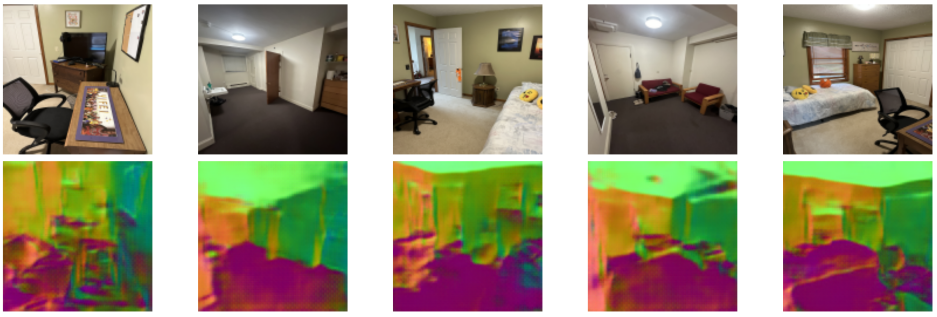

Figure 2: My 5 images

And after there is the 5 images from the validation set test.

In my observations, I believe my model is performing quite well in segmenting the floor. Both my images and the validation models are effectively identifying the floor with a dark purple color. However, it seems challenging to accurately identify the desk as the floor surface. For instance, in the first image, the desk appears quite complex and difficult to discern the surface. This issue persists in

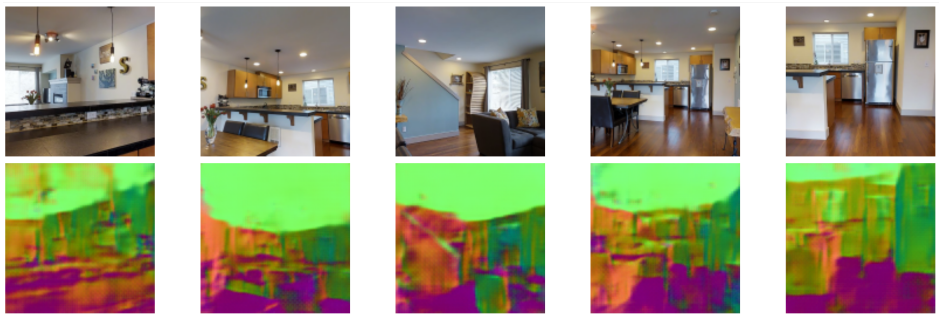

Figure 3: Validation 5 images

the first image of the validation set as well. The resolution of my model needs to be more improved to have a better images.

(e) **Secret test set**

From the gradescope, test scores for the Question 2 test is :

Table 6: Evaluation Q2 Test

Test set Accuracy	Score
Mean error	37.8248
Median error	32.2347
Accurary(11.25 deg)	20.7259
Accurary(22.5 deg)	37.7338
Accurary(30.0 deg)	47.3800