

# DLCV HW1 Report

tags: DLCV

Course	Student ID	Name	Date
2022 Fall NTU DLCV	R10943109	Shiuan-Yun Ding	2022-10-20

HackMD Link: <https://hackmd.io/@mirkat1206/rJBCyY-Gi>

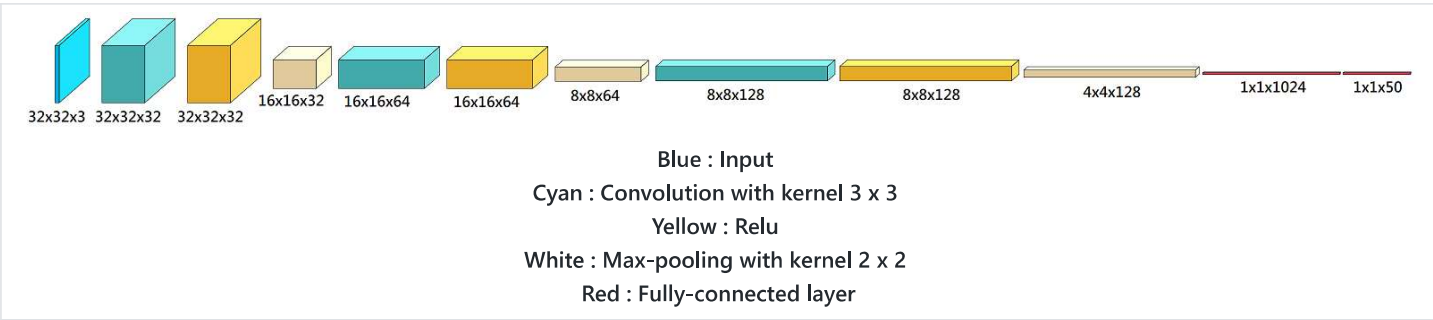
## Problem 1: Image Classification

Input	Output	Training Dataset	Validation Dataset	# of Classes
32x32 RGB Image	Classification Label	22500 Images	2500 Images	50

### (2%) Network Architecture

Network Architecture of Model A

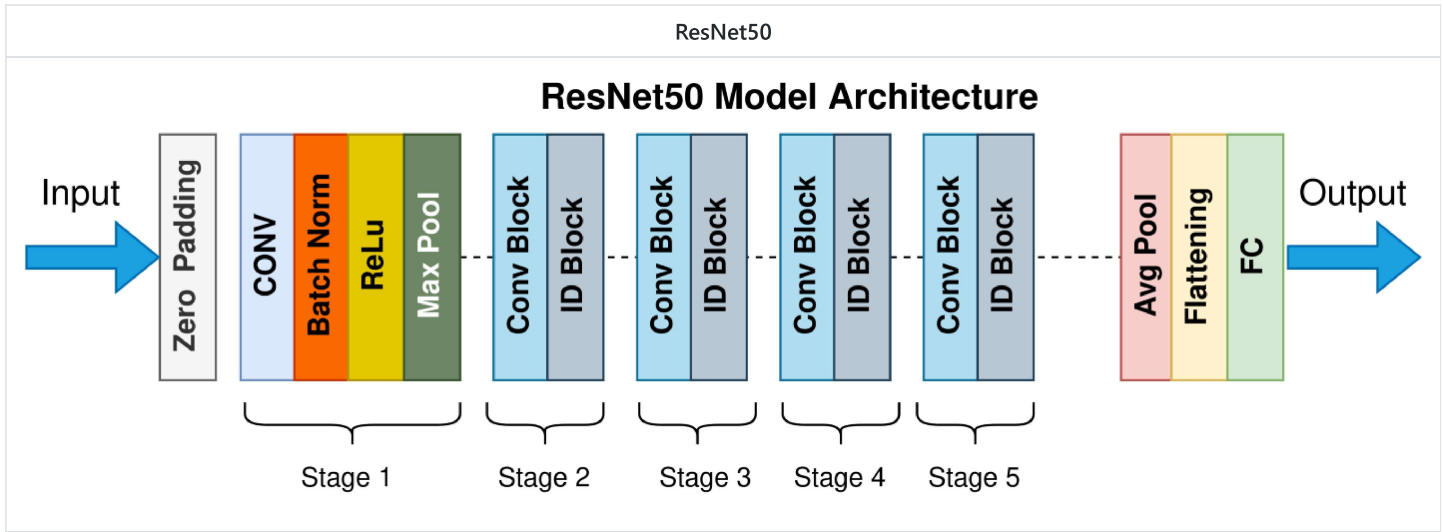
- Input Size : 3 x 32 x 32
- Output Size : 50 x 1 x 1



```
NN(  
  (conv1): Sequential(  
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (1): ReLU()  
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (conv2): Sequential(  
    (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (1): ReLU()  
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (conv3): Sequential(  
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (1): ReLU()  
    (2): Dropout2d(p=0.5, inplace=False)  
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (fc1): Sequential(  
    (0): Flatten(start_dim=1, end_dim=-1)  
    (1): Linear(in_features=2048, out_features=1024, bias=True)  
    (2): Dropout(p=0.5, inplace=False)  
    (3): ReLU()  
  )  
  (fc): Sequential(  
    (0): Linear(in_features=1024, out_features=50, bias=True)  
  )  
)
```

Network Architecture of Model B

- Input Size : 3 x 32 x 32
- Output Size : 50 x 1 x 1



(1%) Accuracy

	Model A	Model B	Simple Baseline	Strong Baseline
Train Set	65.53%	94.40%	x	x
Validation Set	49.04%	80.24%	75.00%	86.00%

(4%) Implementation Details of Model A

- Architecture : see the section above.
- Basic Settings

Optimizer	Loss Function	Batch Size	Epochs
SGD	Cross Entropy Loss	64	30

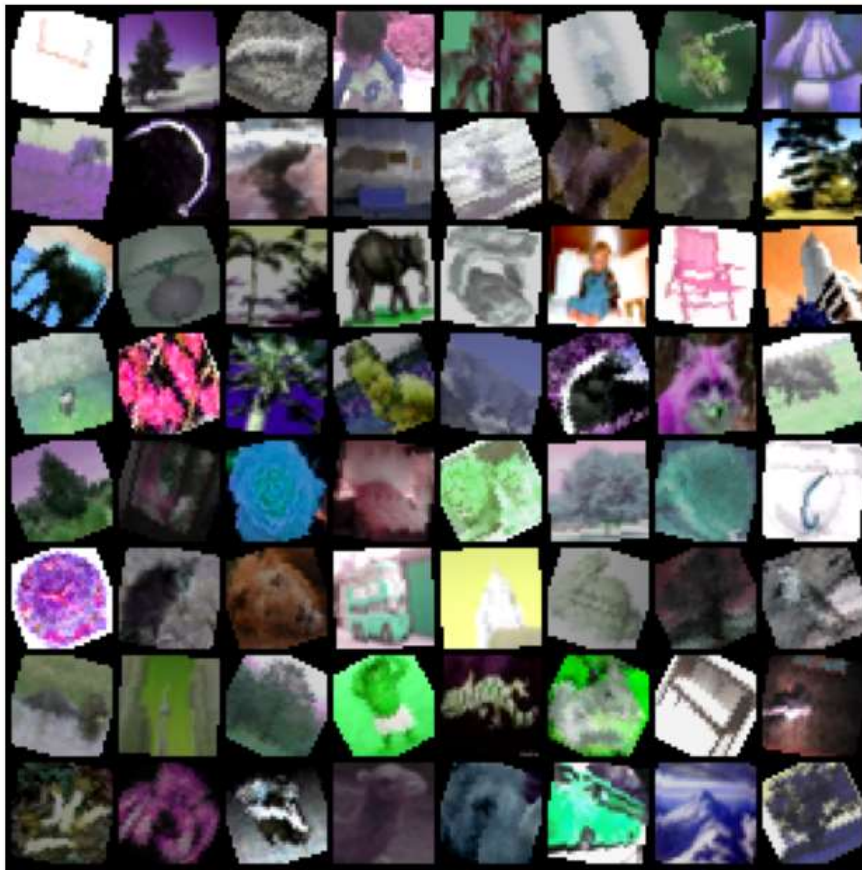
- Optimizer Settings

Optimizer	Learning Rate	Momentum	Weight Decay
SGD	0.01	0.9	0.00001

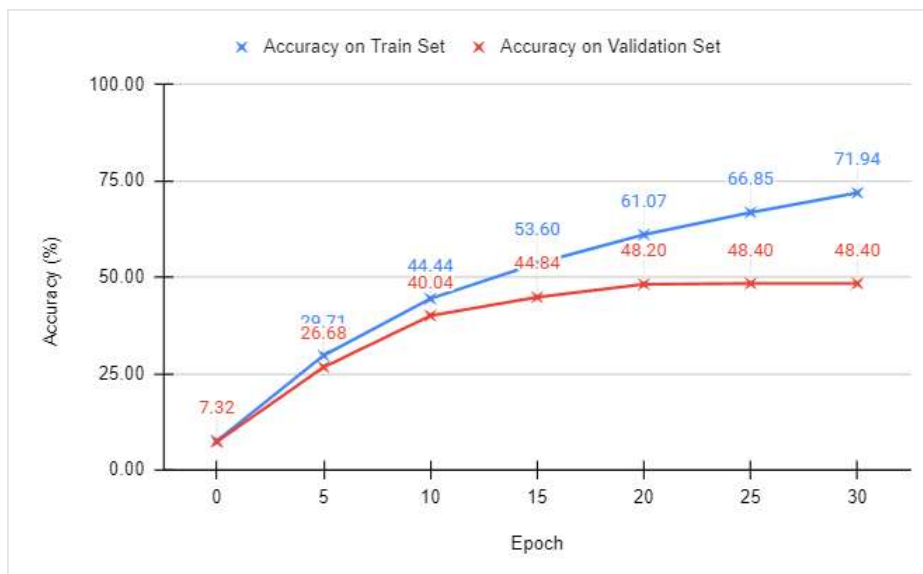
- Data Augmentation

```
transform=transforms.Compose([
    transforms.ColorJitter(brightness=0.5, contrast=0.5, saturation=0.5, hue=0.5),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation((-30, 30), expand=False),
    transforms.ToTensor(),
    transforms.Normalize(mean, std),
])
```

- Visualization of a Train Set Batch



- Accuracy Curve



#### (4%) Implementation Details of Model B

- Architecture : see the section above.
- Basic Settings

Pre-trained Model	Optimizer	Loss Function	Batch Size
ResNet50	Adam	Cross Entropy Loss	64

- Parameters

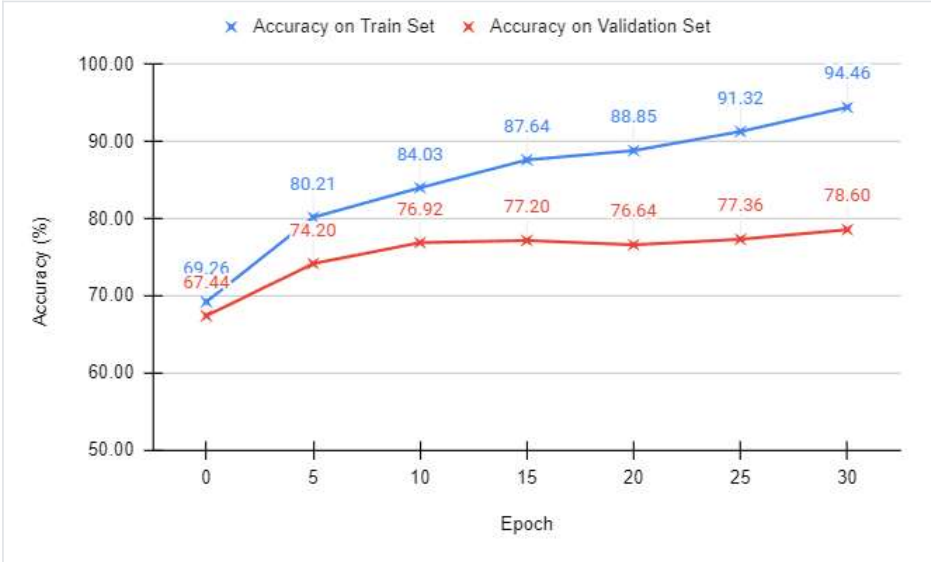
Pre-trained Parameters	Fine-tuned Parameters
ResNet50_Weights.IMAGENET1K_V1	Parameters of the last two layers

• Optimizer Settings

Optimizer	Learning Rate	Betas
Adam	0.0002	(0.9, 0.999)

- Data Augmentation
  - Same as Model A

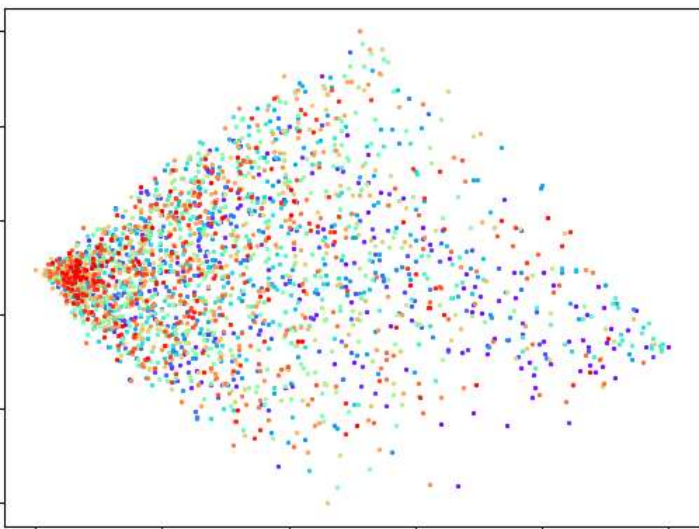
• Accuracy Curve

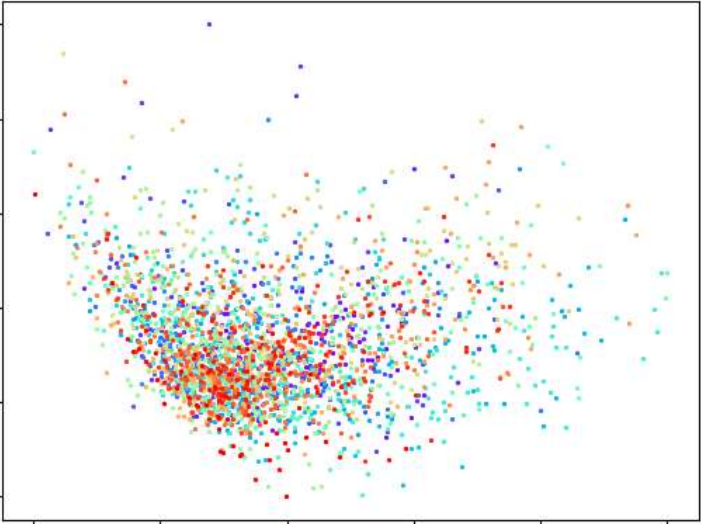
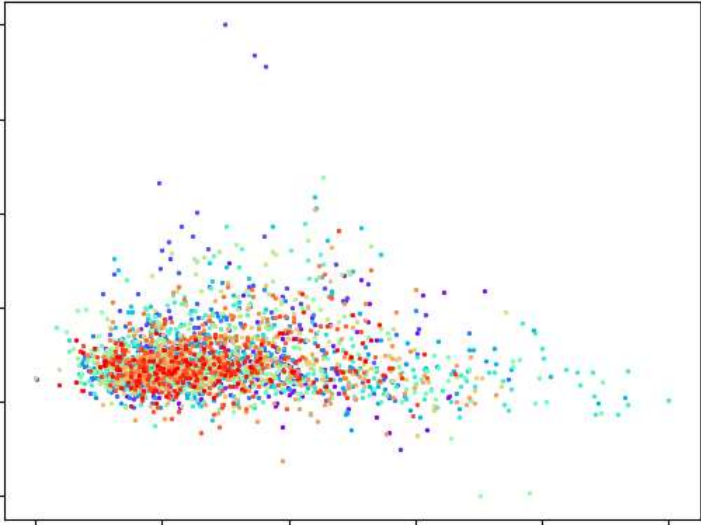


Differences Discusstion

- The architetur of ResNet50 is much deeper and more complicated than the architecture of my model A.
- Initializing ResNet50 with pretrained parameters results in an initial accuracy of 67.44% on validation set. Compared with my model A with random initial parameters, initial accuracy only achieves 7.32% on validation set.
- My model A starts overfitting after 30 epochs with accuracy of around 50% on validation set. On the other hand, the initial accuracy of ResNet50 on validation set starts above 67%, and stops improving at 78%.

(7%) Principal Component Analysis (PCA)

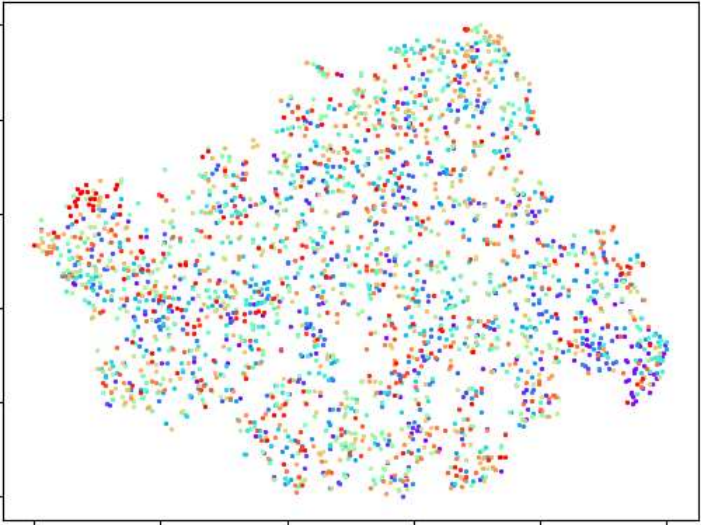
Epoch	Accuracy on Validation Set	t-SNE
0	7.32%	

Epoch	Accuracy on Validation Set	t-SNE
5	26.68%	
30	48.4%	

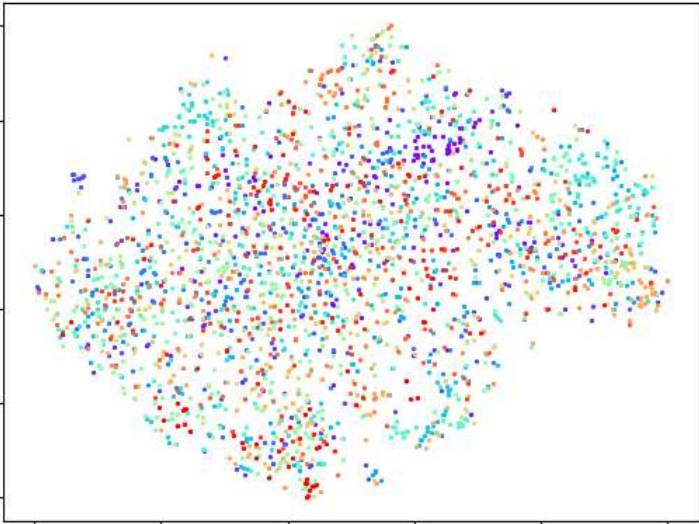
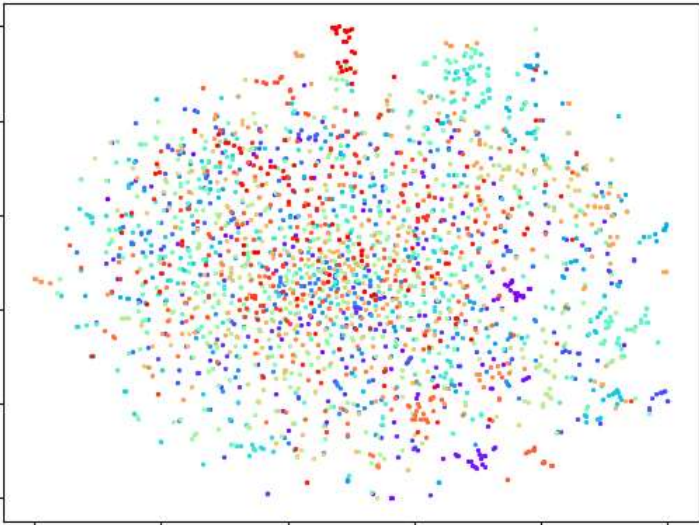
PCA Discusstion

- All the colors(classes) are messed up together even after 30 epochs, which indicates my model A cannot distinguish different classes accurately.

(7%) t-distributed Stochastic Neighbor Embedding (t-SNE)

Epoch	Accuracy on Validation Set	t-SNE
0	7.32%	



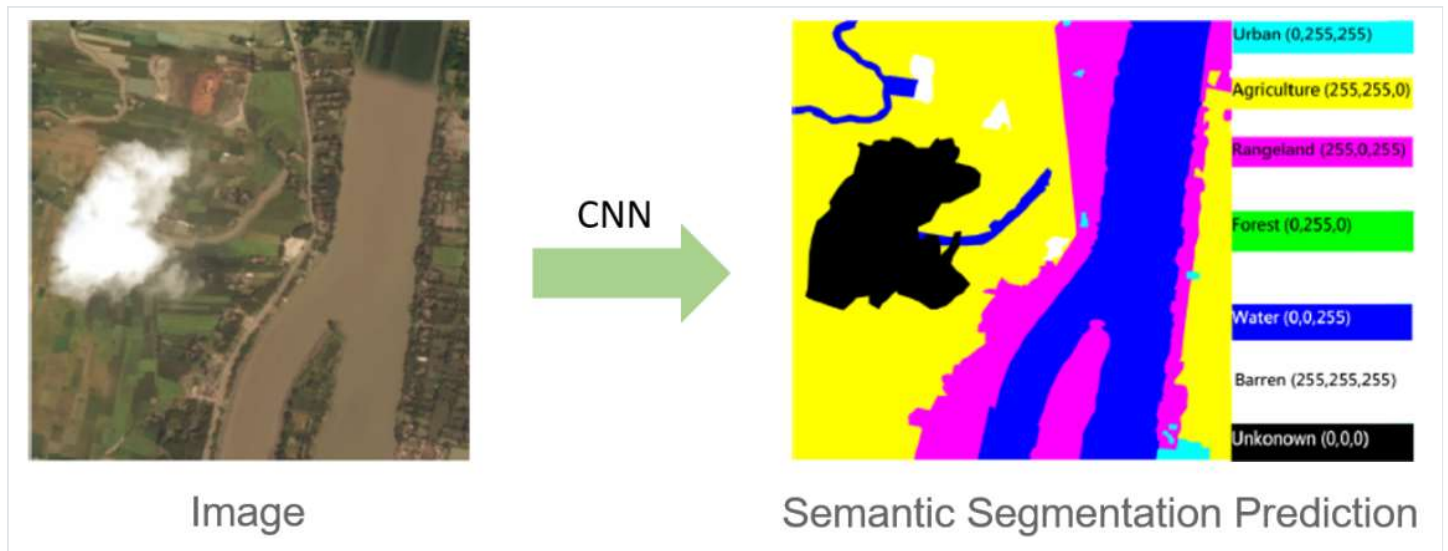
Epoch	Accuracy on Validation Set	t-SNE
5	26.68%	
30	48.4%	

**t-SNE Discusstion**

- At 0-th epoch, all colors(classes) are spread across the canvas equally and randomly.
- At 5-th epoch, some dots are getting together with the same color dots. For example, purplr color dots on the upper-right side, and navy color dots on the upper-left side.
- At 30-th epoch, some group isolately with the same color, such as red, purple, and cyan. But in the middle, every colors still mess up, which means the accuracy of classification task will be terrible.

**Problem 2: Semantic segmentation**

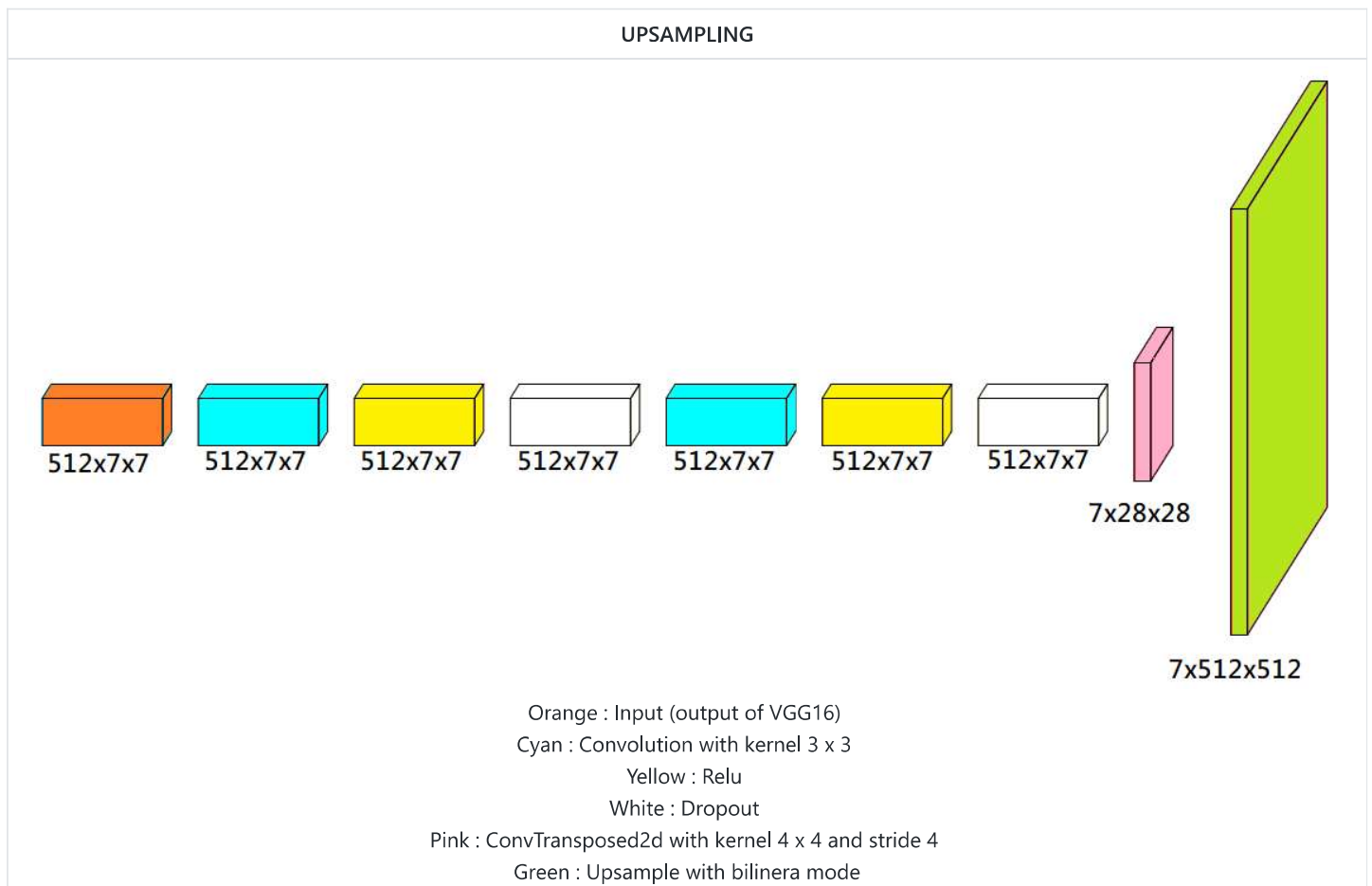
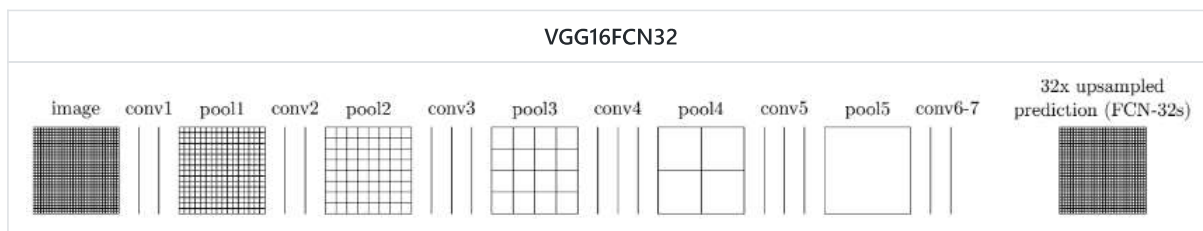
Input	Output	Training Dataset	Validation Dataset	# of Classes
512x512 RGB Image	512x512 Label Image	2000 Images	256 Images	7



## (10%) Network Architecture

### (5%) Network Architecture of Model A

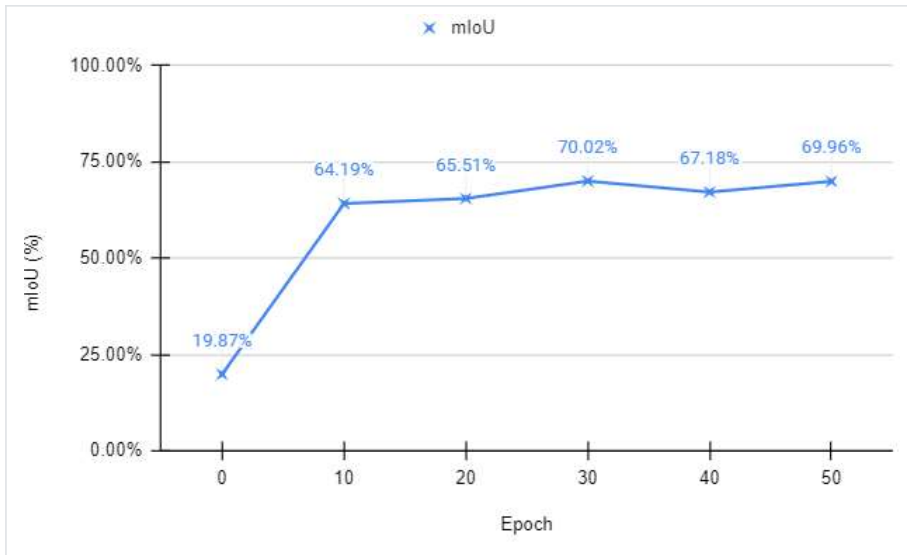
- VGG16FCN32
- Input Size : 3 x 224 x 224
- Output Size : 3 x 512 x 512



## UPSAMPLING

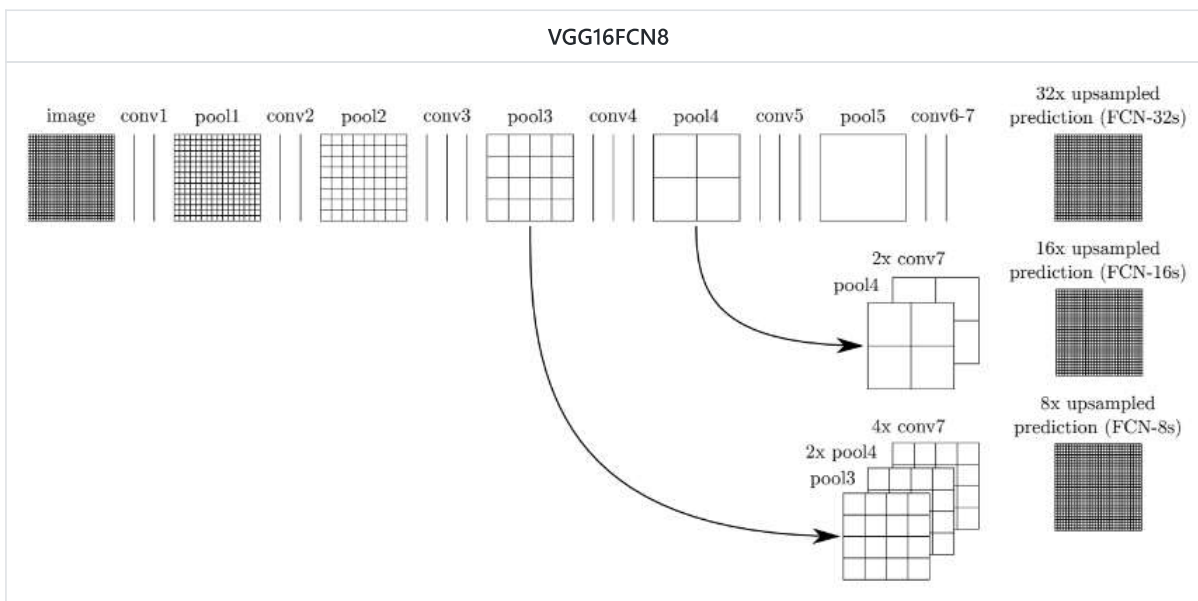
```
(fcn32): Sequential(
  (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU()
  (2): Dropout(p=0.5, inplace=False)
  (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (4): ReLU()
  (5): Dropout(p=0.5, inplace=False)
  (6): ConvTranspose2d(512, 7, kernel_size=(4, 4), stride=(4, 4))
  (7): Upsample(size=(512, 512), mode=bilinear)
)
```

- mIoU Curve



### (5%) Network Architecture of Model B

- VGG16FCN8
- Input Size : 3 x 224 x 224
- Output Size : 3 x 512 x 512



### Upsampling Layers

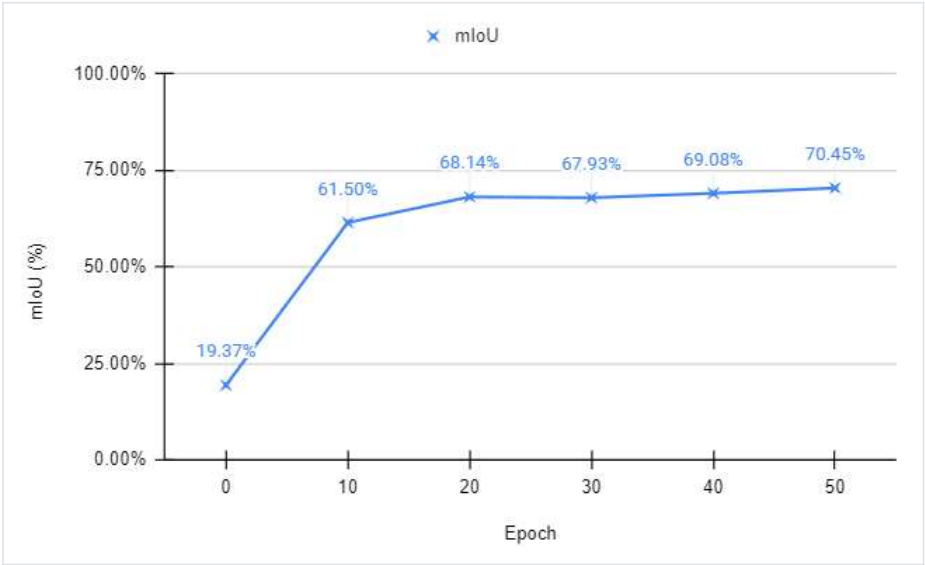
 Cyan : pool3  
 Yellow : pool4 + upsample  
 Pink : pool + convolution  
 Green : Upsample with bilinear mode



### Upsampling Layers

```
(allupsample): Sequential(
  (0): ConvTranspose2d(256, 7, kernel_size=(8, 8), stride=(8, 8))
  (1): Upsample(size=(512, 512), mode=bilinear)
)
```

- mIoU Curve



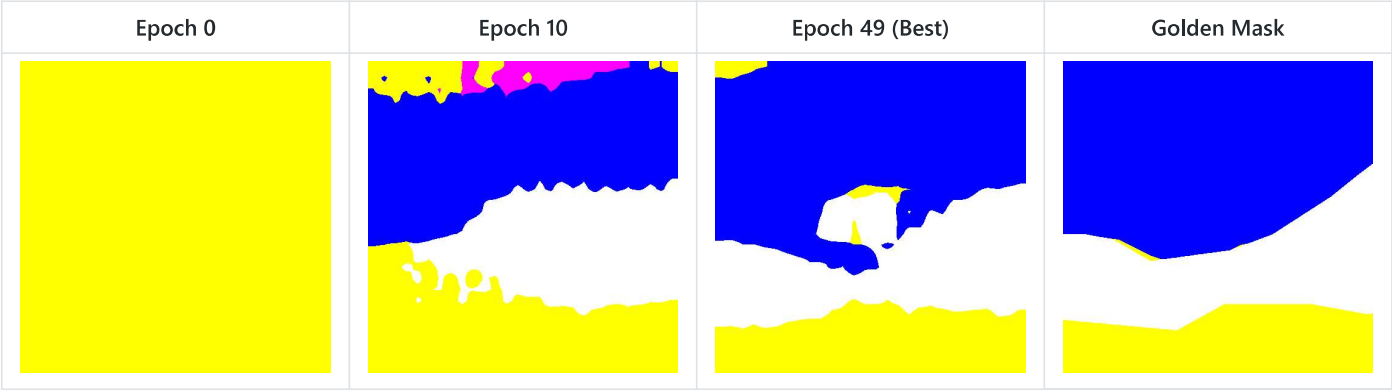
### (3%) mIoU

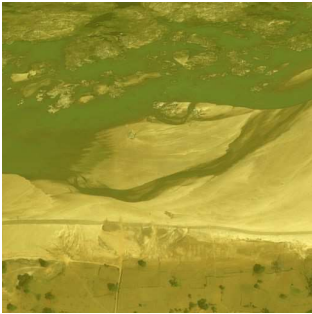
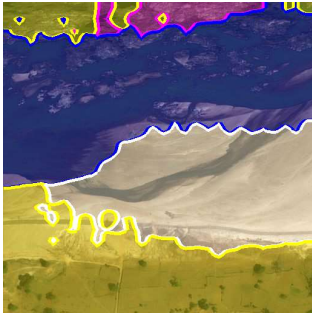
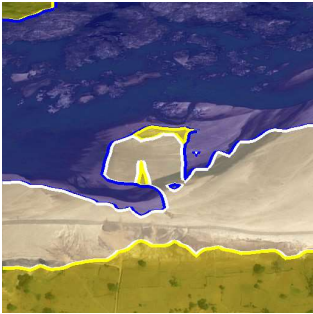

class	VGG16FCN32	VGG16FCN8	Simple Baseline	Strong Baseline
#0	0.73199	0.76444	x	x
#1	0.88839	0.89563	x	x
#2	0.37572	0.45892	x	x
#3	0.79303	0.80901	x	x
#4	0.73959	0.76431	x	x
#5	0.70920	0.73841	x	x
mean_iou	0.706319	0.738451	0.69	0.73

### (7%) Predicted Segmentation Mask

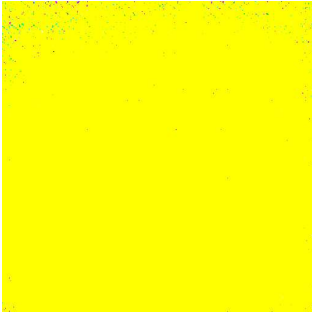

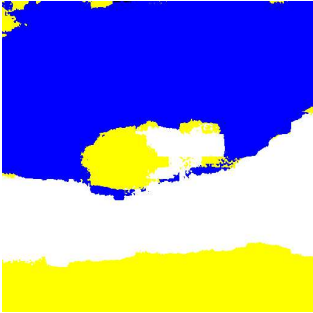
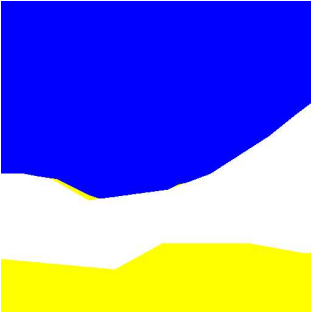
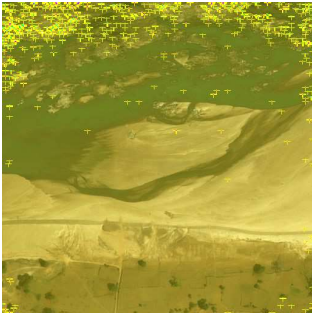
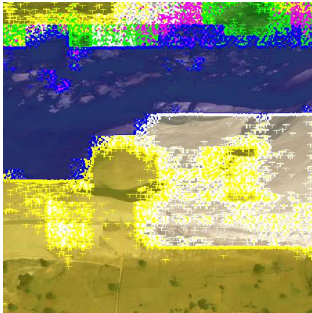


0013

- Model A : VGG16FCN32



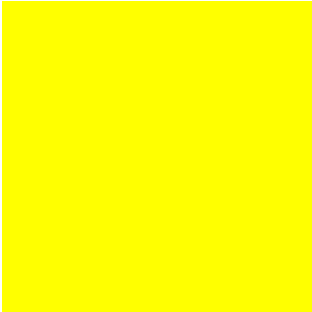
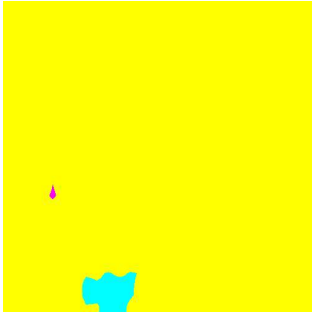





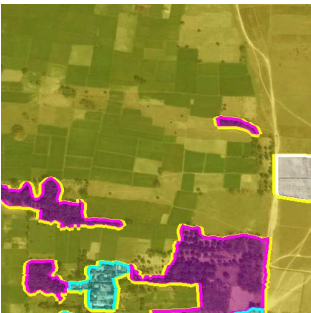
Epoch 0	Epoch 10	Epoch 49 (Best)	Golden Mask
			

• Model B : VGG16FCN8


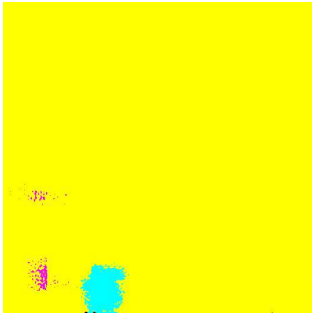





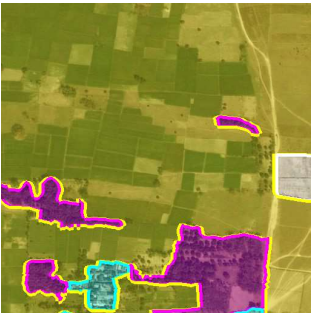
Epoch 0	Epoch 10	Epoch 128 (Best)	Golden Mask
			
			

0062

• Model A : VGG16FCN32


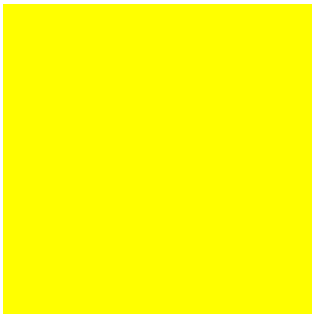
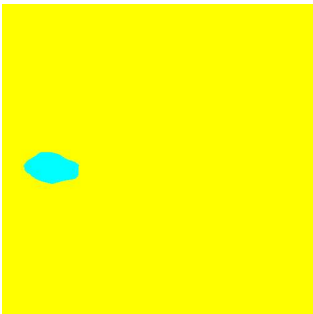

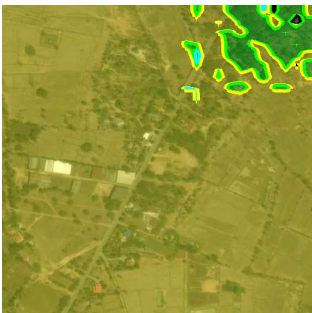


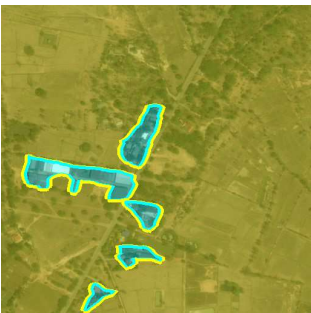
Epoch 0	Epoch 10	Epoch 49 (Best)	Golden Mask
			
			

- Model B : VGG16FCN8

Epoch 0	Epoch 10	Epoch 128 (Best)	Golden Mask
			
			

0104

- Model A : VGG16FCN32

Epoch 0	Epoch 10	Epoch 49 (Best)	Golden Mask
			
			

- Model B : VGG16FCN8

Epoch 0	Epoch 10	Epoch 128 (Best)	Golden Mask
---------	----------	------------------	-------------

Epoch 0	Epoch 10	Epoch 128 (Best)	Golden Mask
