Master's Thesis Presentation

Analysis of Unsupervised Segmentation Methods using SLIC Refinement

Presented by: Md. Saiful Islam Sajol (30042836)

Examiners: Prof. Dr.-Ing. Andreas Schwung
Prof. Dr. Dominik Aufderheide

Supervisor: Jan Niclas Reimann, M.Sc.

Fachhochschule Südwestfalen
University of Applied Sciences

Outline

1. **Image Segmentation**
    - What is image?
    - Different Types of Segmentation and Applications
    - Deep learning based Image Segmentation Approaches
2. **Motivation**
    - Deep learning based Image Segmentation Approaches
    - Architecture for Unsupervised Approach
3. **Method Description and Problem Statement**
    - Base Architecture
    - Results of Base Architecture
    - Challenge and Limitations
4. **Proposed Architecture 1 : FCN with Classification Approach**
    - Results of FCN with Classification Approach
5. **Proposed Architecture 2 : FCN with Autoencoder Approach**
    - Results of FCN with Classification Approach

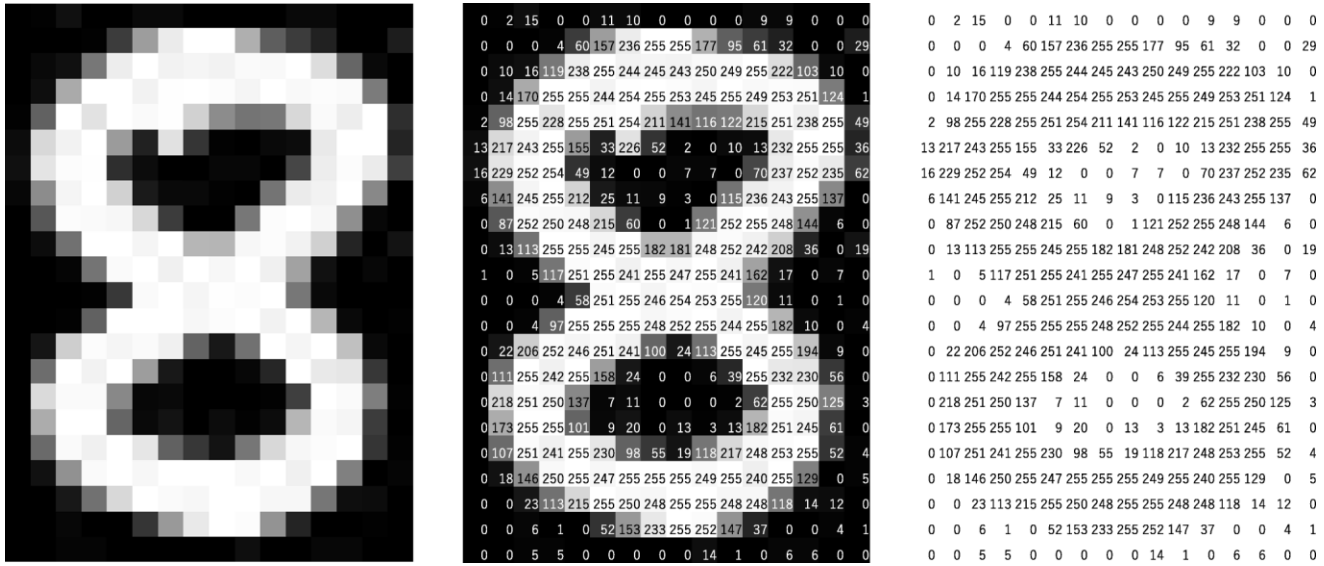
Outline

5. Comparisons of Different Architecture
6. Challenges and Future Work
7. Conclusion
8. Appendix

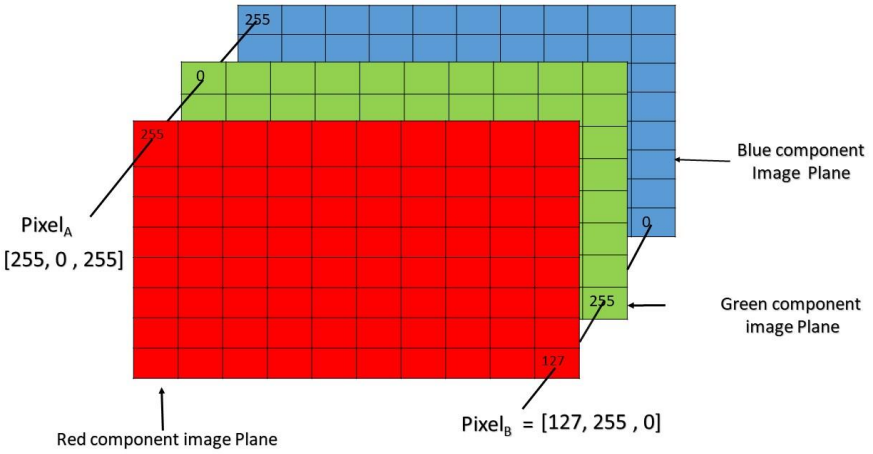
Image Segmentation

What is an Image?

- An image is an array of numbers.



Grey Image



Pixel of an RGB image are formed from the corresponding pixel of the three component images

https://www.geeksforgeeks.org/matlab-rgb-image-representation/

RGB Image

Yamashita, R., Nishio, M., Do, R.K.G. *et al.* Convolutional neural networks: an overview and application in radiology. *Insights Imaging* **9,** 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9

Fachhochschule Südwestfalen
University of Applied Sciences

Image Segmentation

Four Main Categories of Computer Vision Tasks

Image Classification

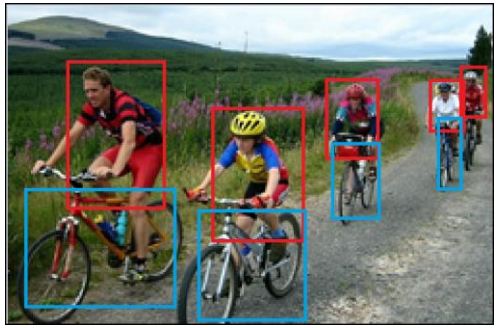
- Classify the object within an image

Semantic Segmentation

- Classify the object class for each pixel (pixelwise classification)

Object Detection

- Classify and detect the objects with bounding boxes



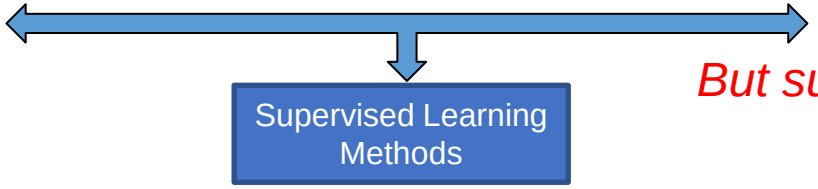
Instance Segmentation

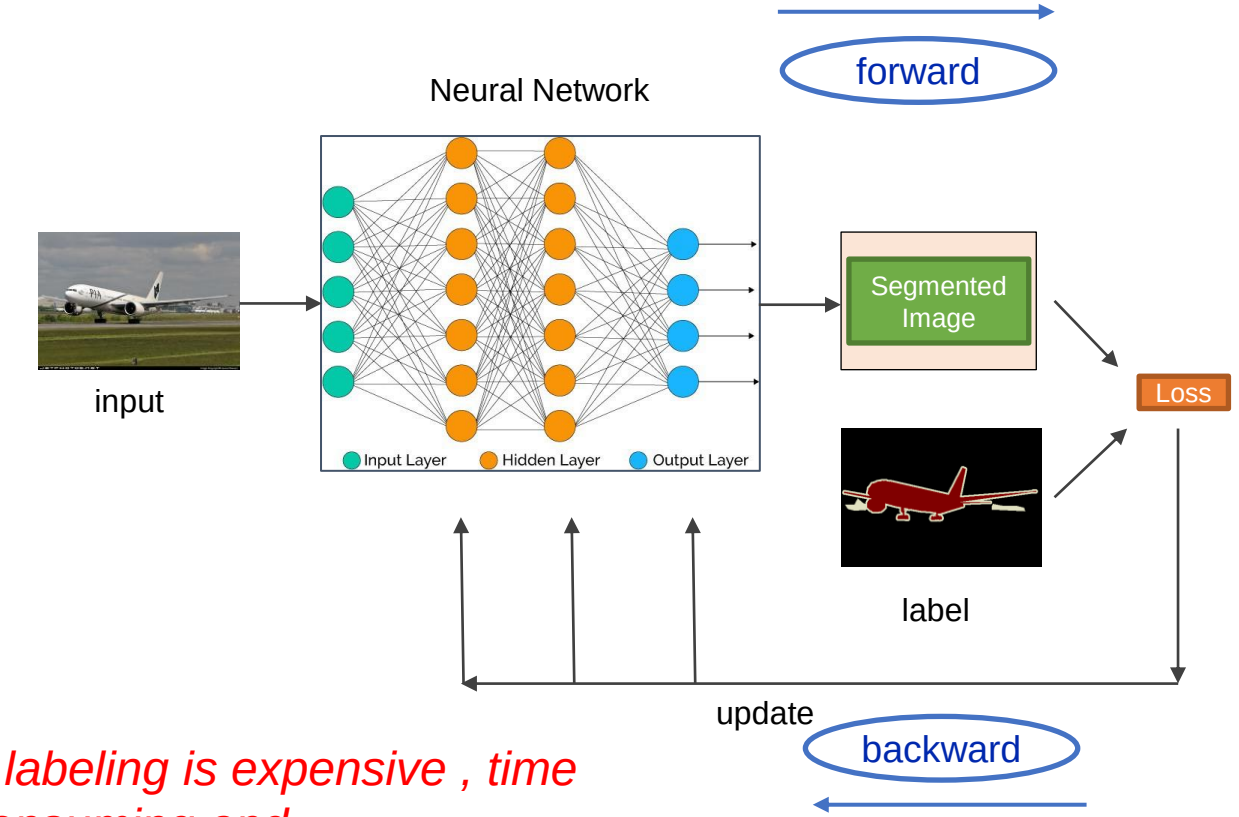
- Label each foreground pixel with object and instance

Motivation

Popular Deep Learning Based Image Segmentation Models

- Fully convolutional networks
- Convolutional models with graphical models
- Convolutional models with active contour models
- Dilated convolutional models and DeepLab family
- Encoder-decoder based models
- Multi-scale and pyramid network based models
- Generative models and adversarial training
- R-CNN based models (for instance segmentation)
- Attention-based models
- Recurrent neural network based models





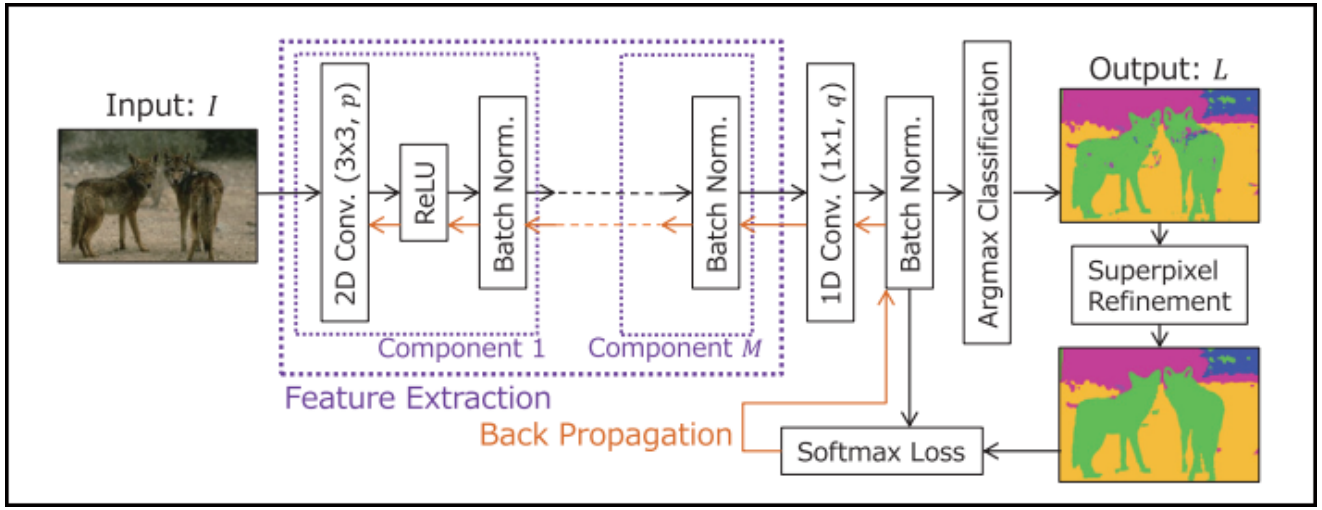
But supervised labeling is expensive , time consuming and difficult to obtain!!

Figure: A Basic Neural Network for Supervised Segmentation

So, we tried to find an unsupervised way

Method Description
Architecture for Unsupervised Approach

We Used CNN for Unsupervised Image Segmentation with SLIC Refinement to calculate Loss for Backpropagation



A. Kanezaki, "Unsupervised Image Segmentation by Backpropagation," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB, Canada, 2018, pp. 1543-1547, doi: 10.1109/ICASSP.2018.8462533.

Main Components

- ❑ Convolutional Operation
- ❑ Argmax Classification
- ❑ Superpixel Refinement (SLIC)
- ❑ Softmax Loss/ Cross Entropy Loss

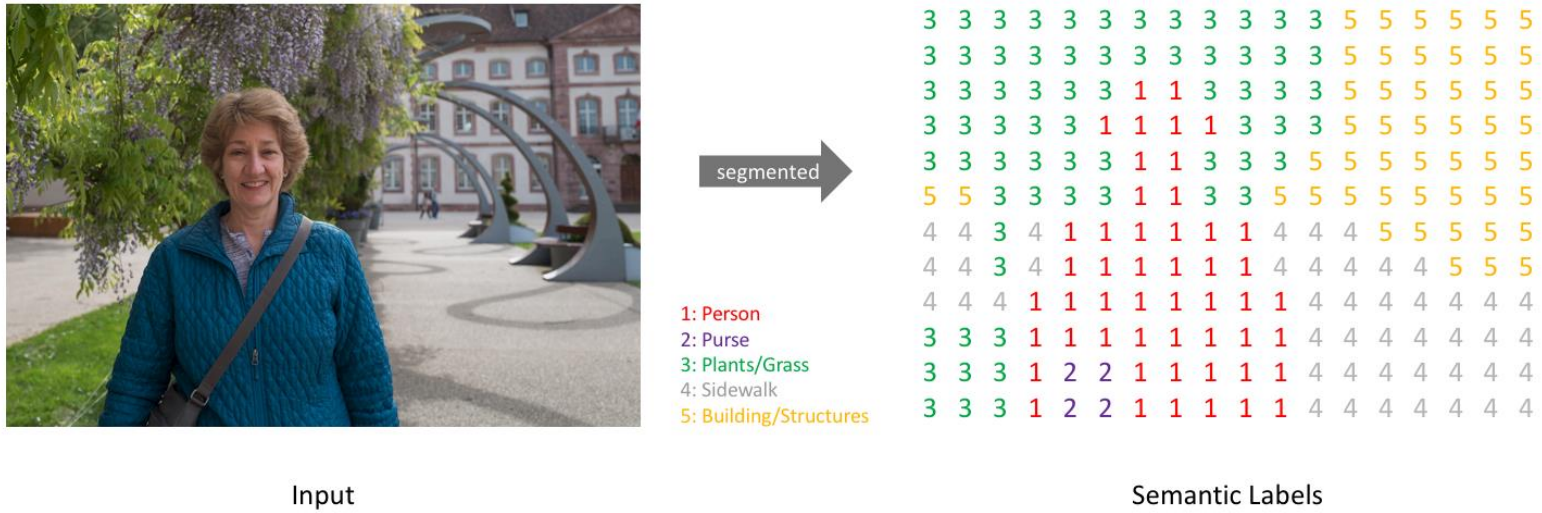
Method Description
Architecture for Unsupervised Approach

Argmax Classification

$$g(x) = x^2$$

$$g(1) = 1$$
$$g(2) = 4$$
$$g(3) = 9$$
$$g(4) = 16$$

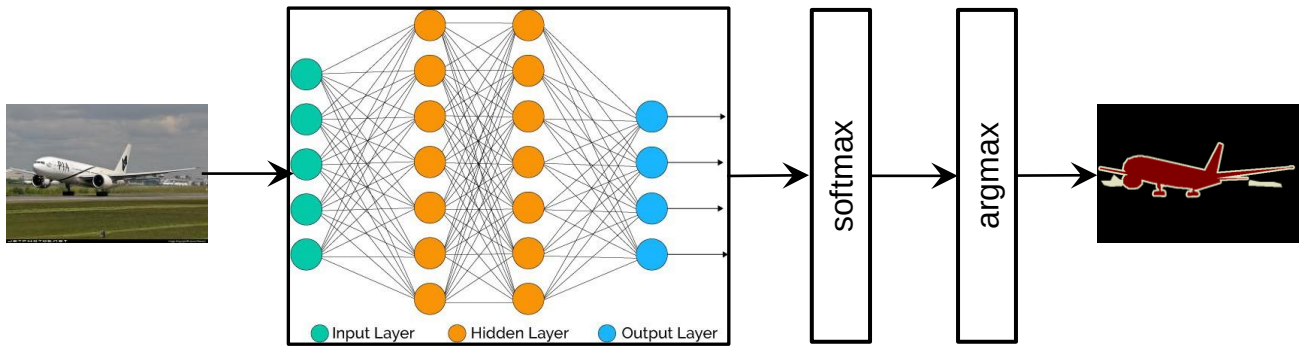
$$Argmax(g(x)) = 4$$



Input

Semantic Labels

Image source: https://www.jeremyjordan.me/semantic-segmentation/



Fachhochschule Südwestfalen
University of Applied Sciences

Method Description
Architecture for Unsupervised Approach

Superpixel

A superpixel is a group of pixels that shares common characteristics (like color, pixel intensity)

- Carry more information
- Similar visual color
- Convenient and compact representation of image

SLIC Superpixel

Simple Linear Iterative Clustering (SLIC) adapts **kmeans** clustering to generate superpixels based on their **color similarity and proximity**

Why SLIC Superpixel?

- Assigning same label to all pixels in a superpixel
- Very good edge detector
- Outperforms other state of the art algorithms

SLIC Superpixels Compared to State-of-the-art Superpixel Methods . Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine S¨usstrunk

Method Description
Architecture for Unsupervised Approach

How SLIC algorithm works?

- Convert RGB image to CIELAB Color space (5d vector)

  RGB → $[l\ a\ b\ x\ y]$

  where $[l\ a\ b\ ]$ is the pixel color vector

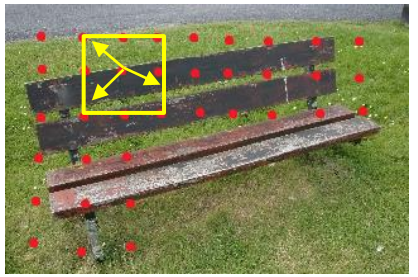
  $x\ y$ is the pixel position

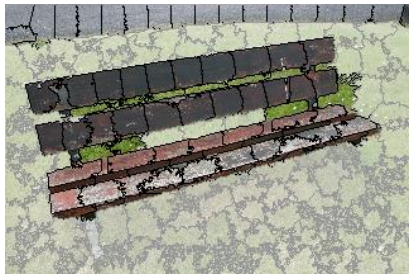
- Take no. of superpixel, K and Calculate $S = \sqrt{N/K}$
- Create a grid of initial centeres with S spacing
- Calculate gradients and move the grid centres which are on edges
- Perform K means locally

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}$$

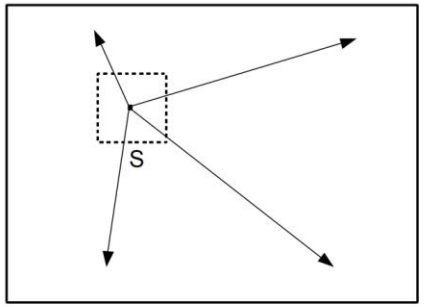
$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

$$D = \sqrt{{d_c}^2 + (\frac{d_s}{S})^2}\ m^2$$

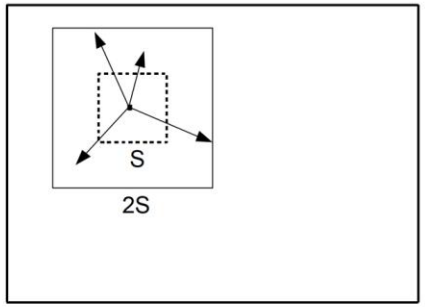




N	Number of pixels in the input image
K	Number of Superpixels used to segment the input image
N\K	Approximate number of pixels in each superpixel
S=√(N/K)	Approximate length of a superpixel



(a) standard k-means searches the entire image



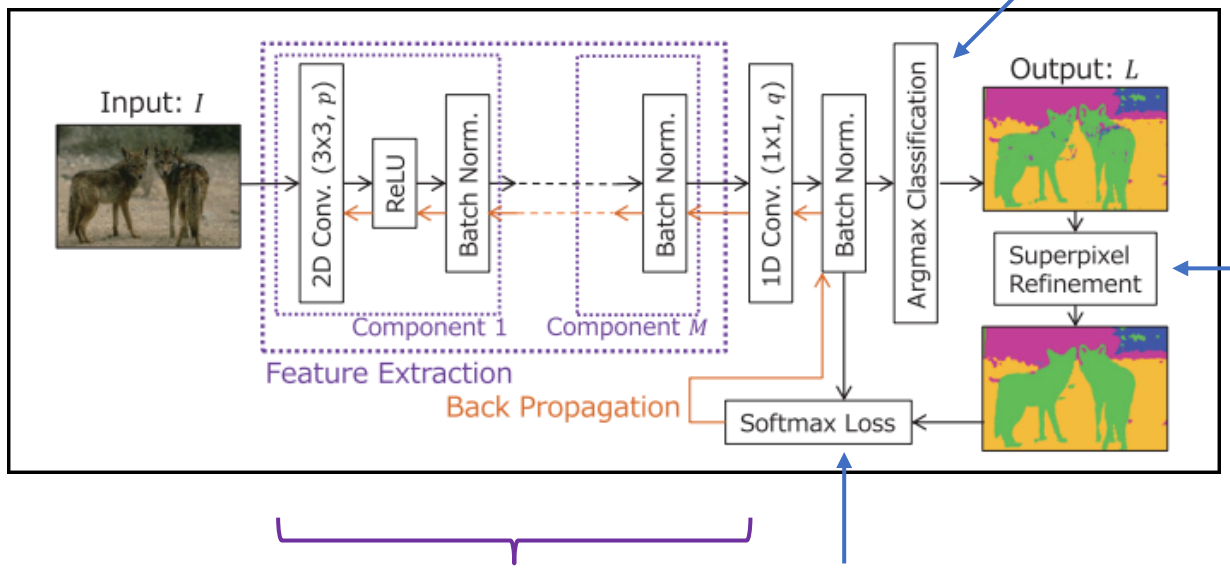
(b) SLIC searches a limited region

Source:SLIC Superpixels Compared to State-of-the-art Superpixel Methods . Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine S¨usstrunk

Method Description
Architecture for Unsupervised Approach

FCN for Unsupervised Image Segmentation with SLIC Refinement

(a) Pixels of similar features assigns the same label.
(b) Spatially continuous pixels assigns the same label



(a) Refines the segmented image
(b) Forces all of the pixels in each superpixel to have the same cluster label
(c) Puts spatial continuity constraint

between the network responses and the refined cluster labels

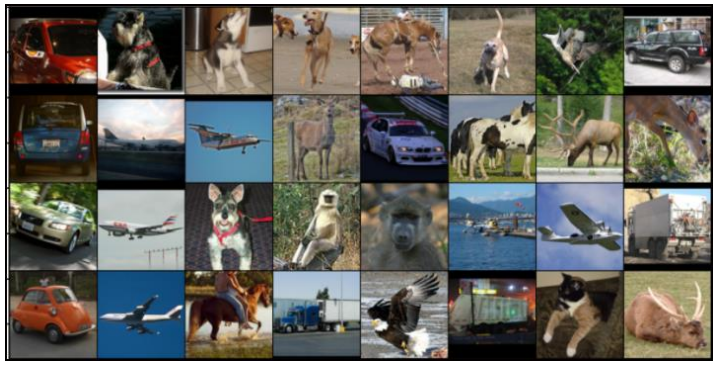
The above architecture is used to reproduce the results

Datasets

CIFAR10

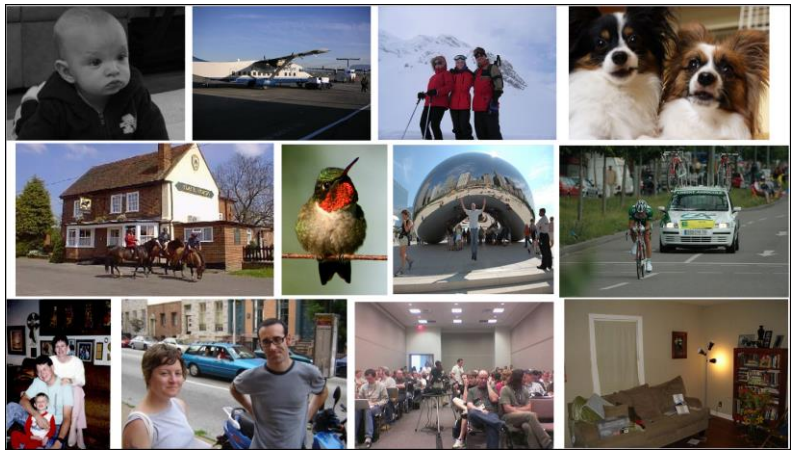
- 60,000 images
- 32x32 pixels
- 50,000 train images
- 10,000 test images
- 10 classes

STL10



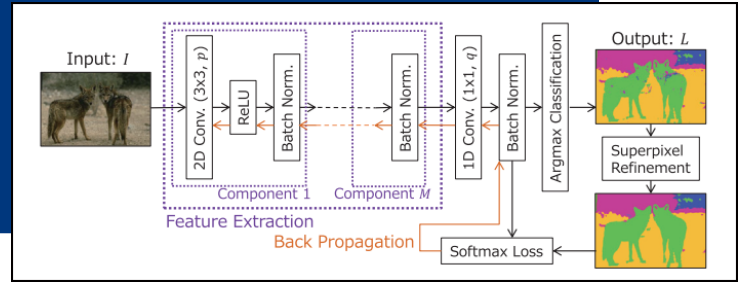
- 13,000 images
- 96x96 pixels
- 5,000 train images
- 8,000 test images
- 10 classes

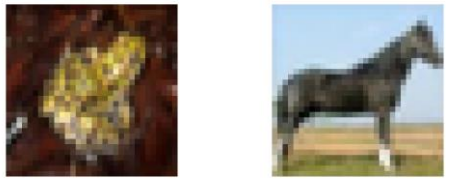
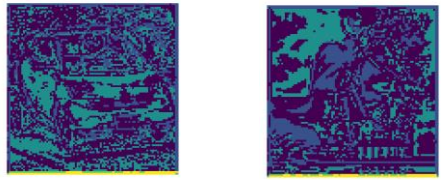
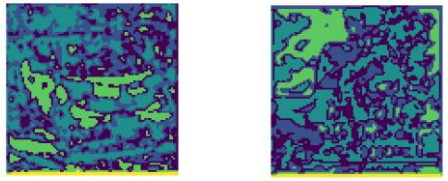
Pascal VOC 2012



- 2,913 images
- 500x350 pixels (minimum)
- 1,464 train images
- 1,449 test images
- 20 classes
- We down sampled to 224x224pixels

Base Architecture
Results

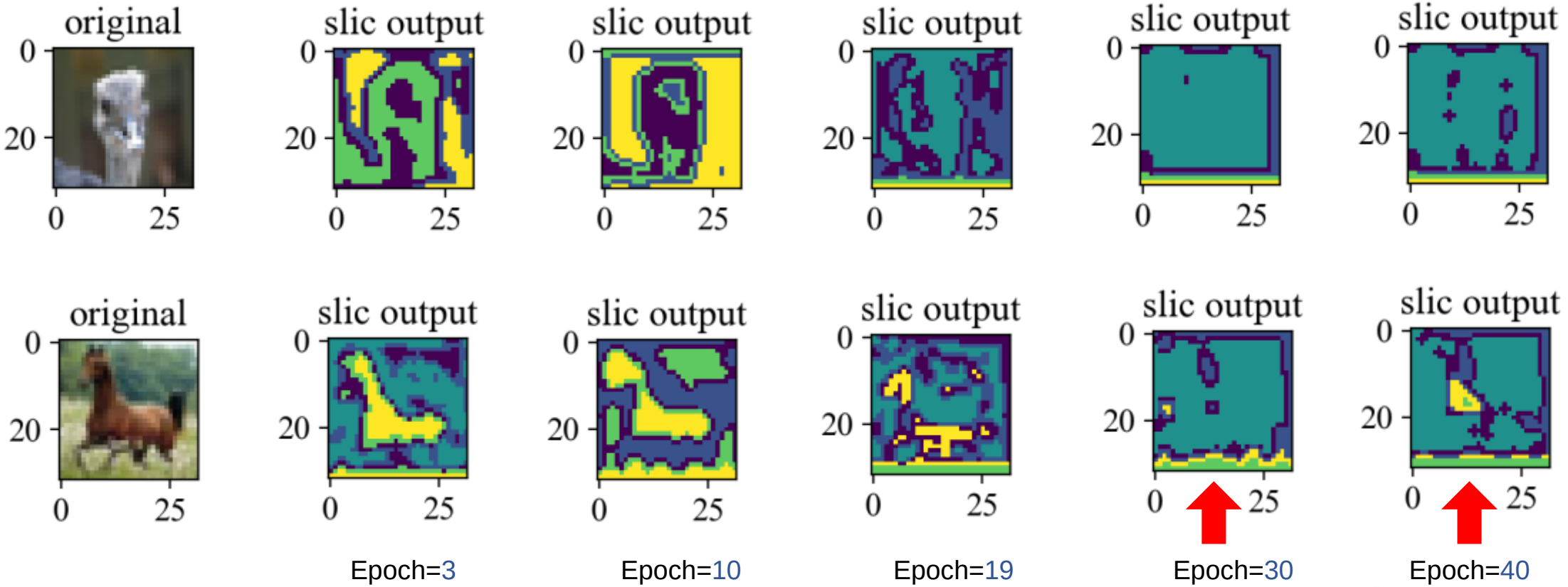


	Dataset: CIFAR10	Dataset: STL10	Dataset: Pascal VOC
Main Image			
Argmax Output			
SLIC Output			
	Epochs=30, LR=0.001, BS=16, Opt=SGD	Epochs=30, LR=0.0001, BS=5, Opt=SGD	Epochs=30, LR=0.001, BS=16, Opt=Adam

Base Architecture
Challenges and Limitations

Segmentation During the Training

Dataset: CIFAR10

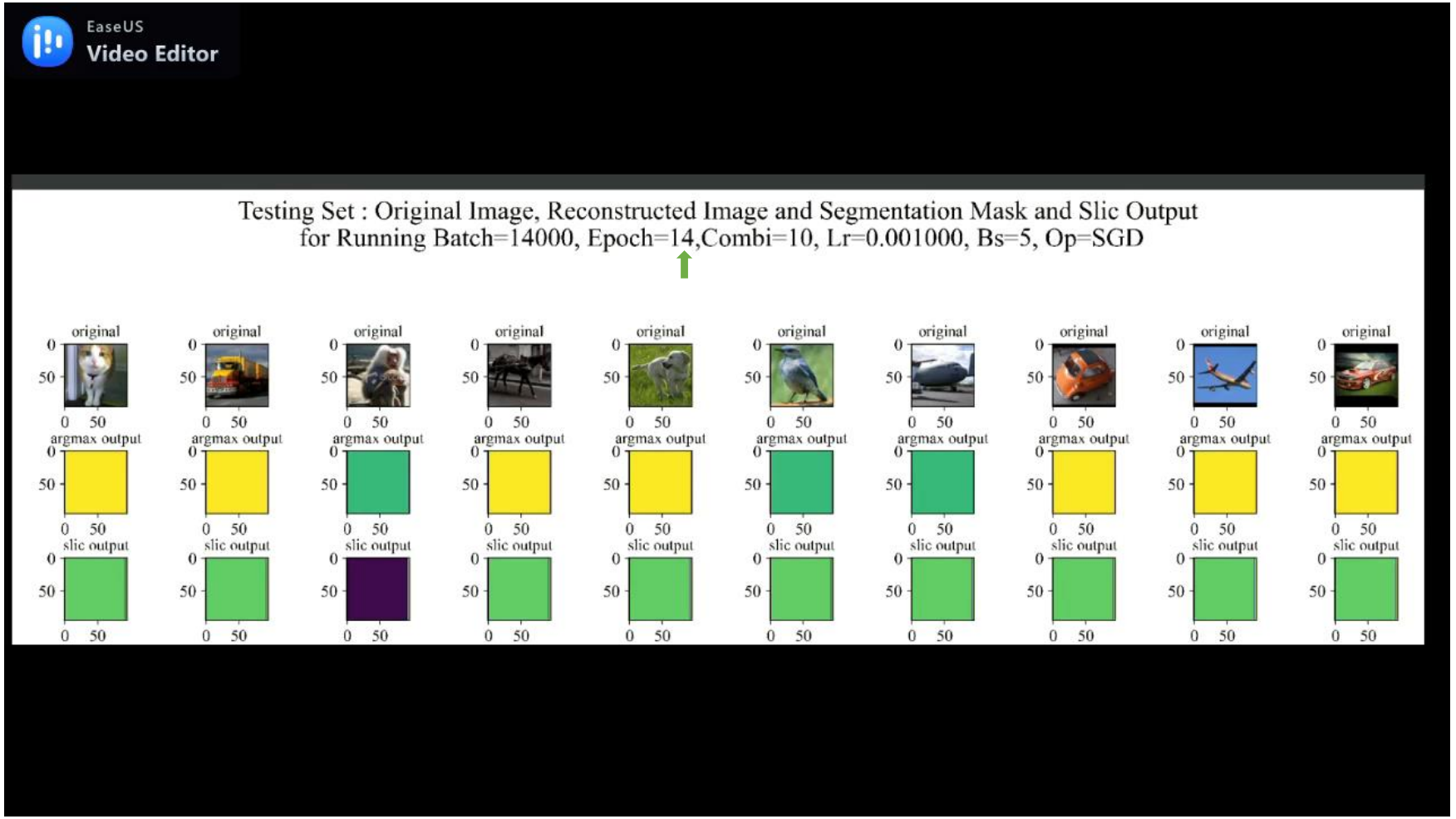


LR=0.001, BS=32, Opt=Adam

Base Architecture
Problem Description of the Base Architecture

Segmentation During the Training

Dataset: STL10



Challenges and Limitations

- The model is not robust enough
- Segmentations were merging over training
- Greatly affected by parameters
- Results are different at multiple random restarts
- No mechanism to restricts learned experience

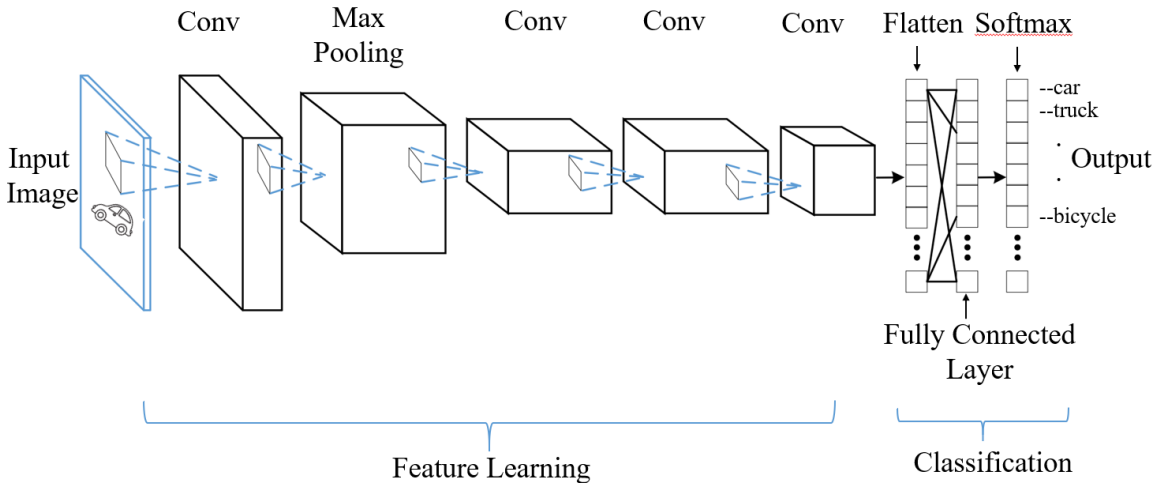
Proposed Architectures for Problem Solution

We propose two possible approaches to solve these problems

1. FCN with Classification Approach
2. FCN with Autoencoder Approach

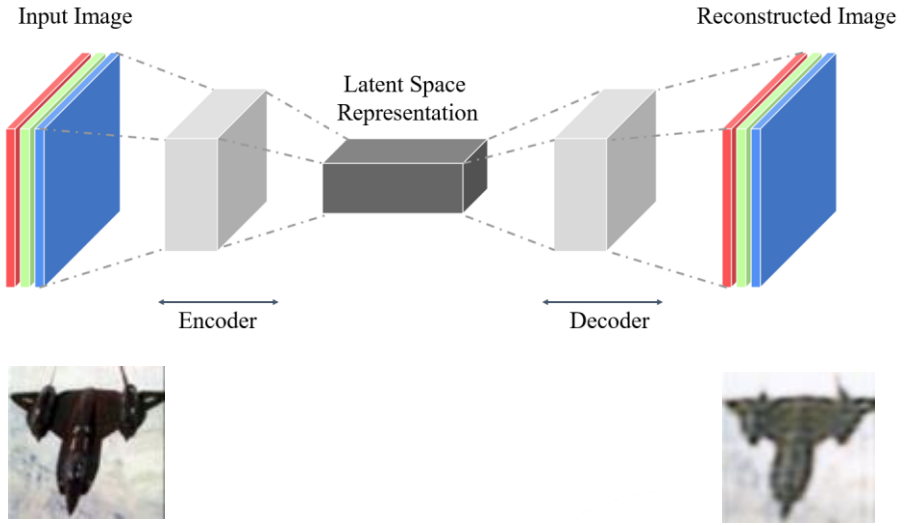
Image Classification

Image classification is where a network classify images into given classes or labels
A class is essentially a label, for instance,
'car' → 0
'truck' → 1
'van' → 2



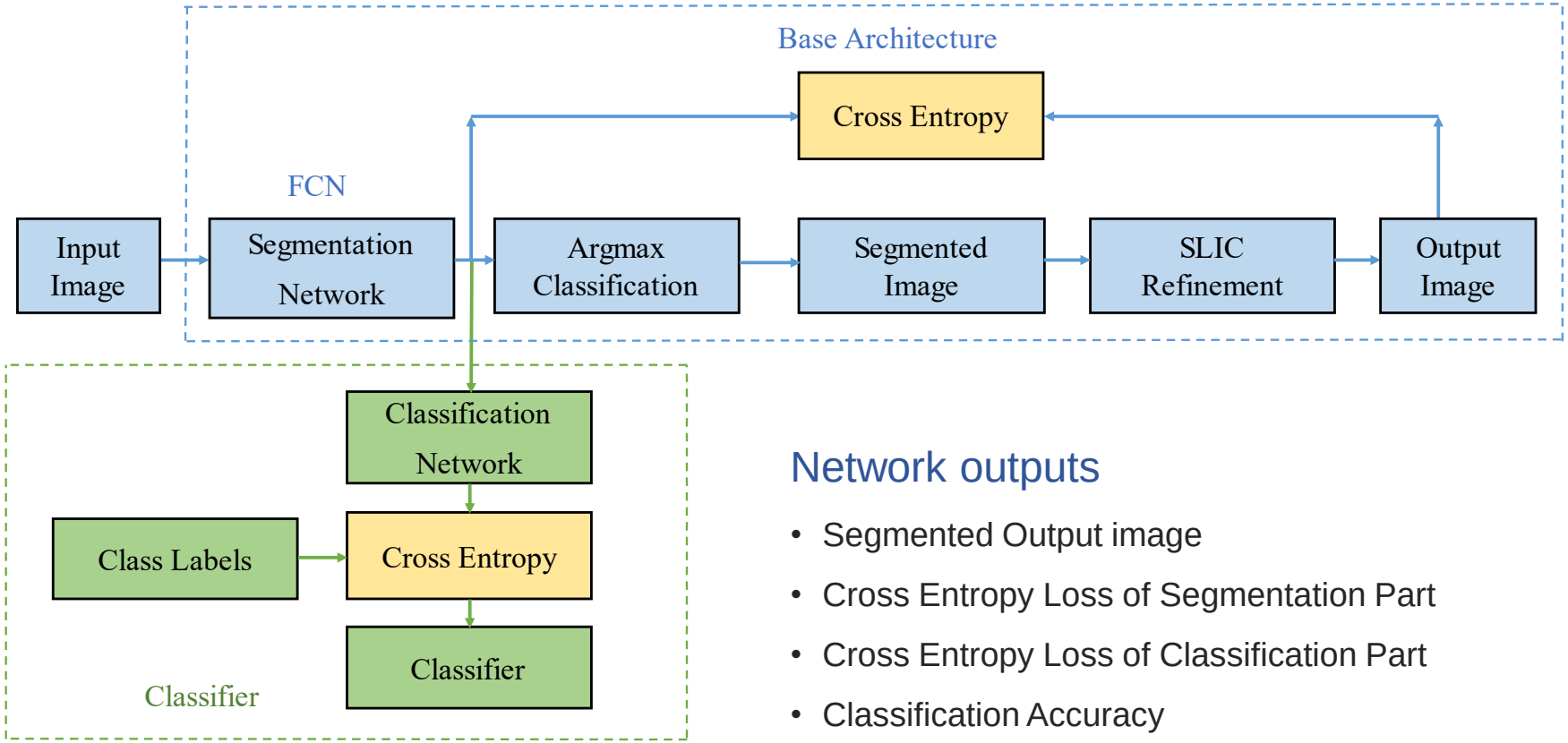
Autoencoder

The autoencoder extracts features, and compresses all the important attributes and again reconstruct the original input.



Proposed Architecture 1 for Problem Solution

Proposed Architecture 1: FCN with Classification Approach



Network outputs

- Segmented Output image
- Cross Entropy Loss of Segmentation Part
- Cross Entropy Loss of Classification Part
- Classification Accuracy

Hyperparameters for Network Tunning

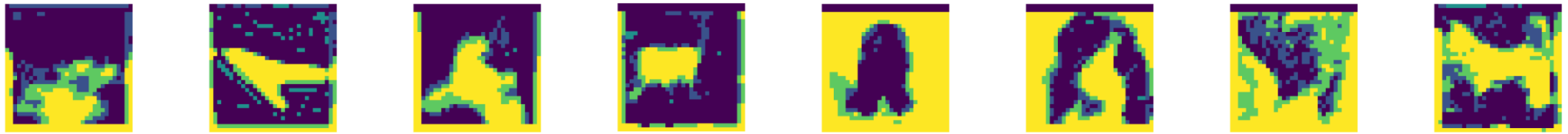
Hyperparameters	Range / Value
Epochs	30, 40, 50, 60, 100
Normalization	minmax (0 to 1)
Activation Function	ReLU
Batch Size	5, 16, 32, 50, 100
Learning Rate	0.1, 0.01, 0.001, 0.0001
Optimizer	Adam, SGD
Initializer	Xavier Initializer
Scheduler	StepLR

Results
Proposed Architecture 1: FCN with Classification Approach

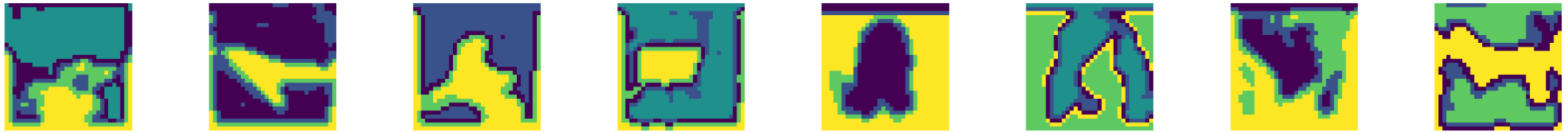
Dataset: CIFAR10

Main Image

Argmax Output



SLIC Output

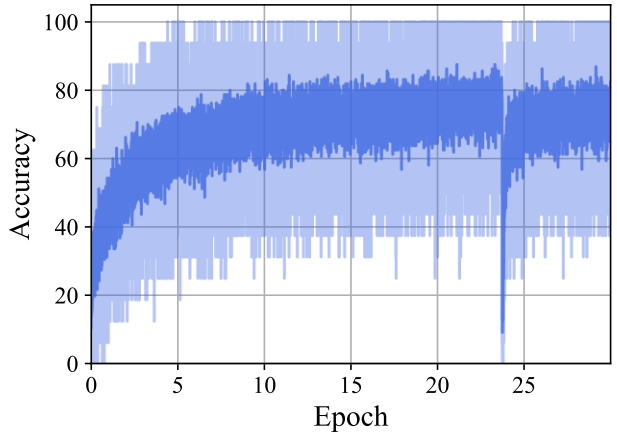


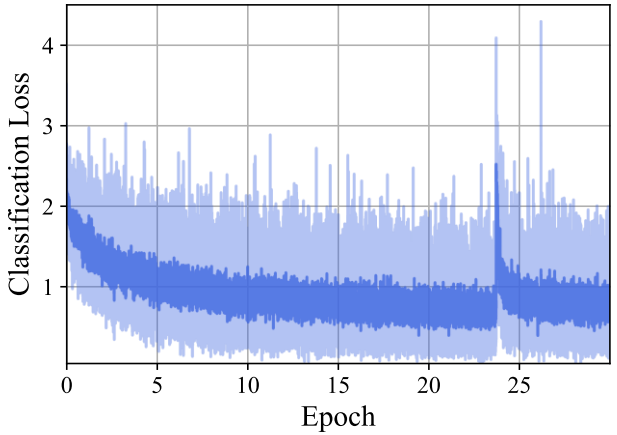
Epochs=30, LR=0.01, BS=16, Opt=SGD

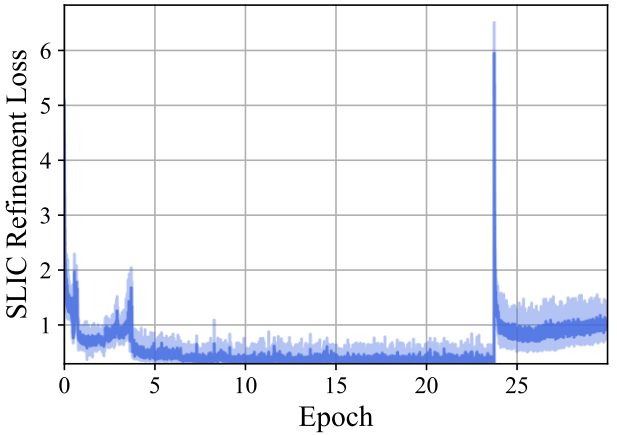
Results
Proposed Architecture 1: FCN with Classification Approach

Dataset: CIFAR10

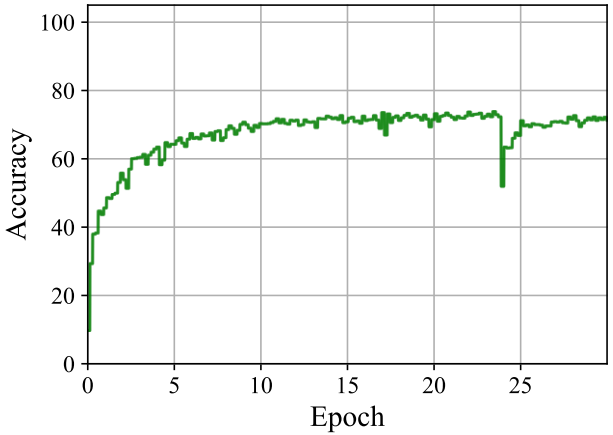
Training

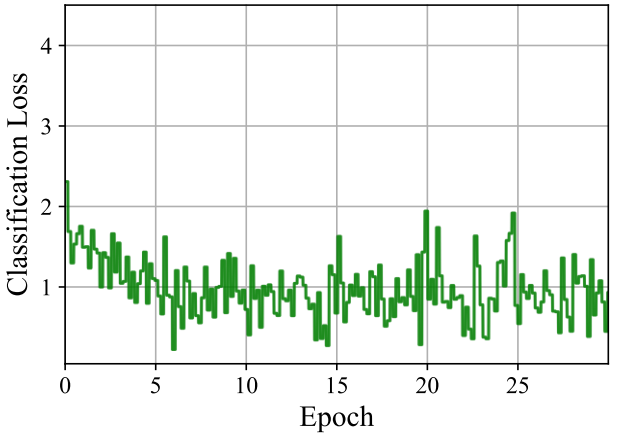


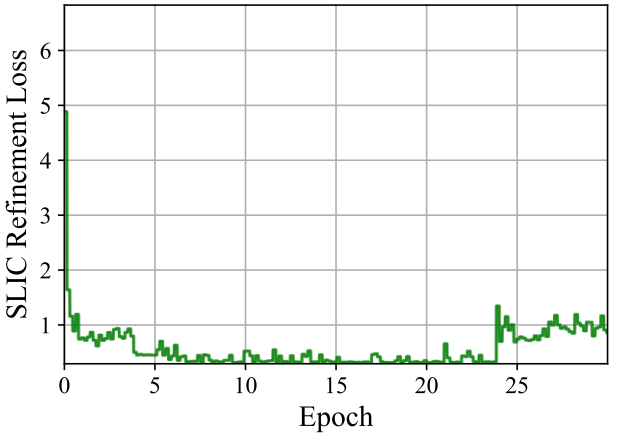




Testing







Epochs=30, LR=0.01, BS=16, Opt=SGD

Results
Proposed Architecture 1: FCN with Classification Approach

Dataset: STL10

Main Image

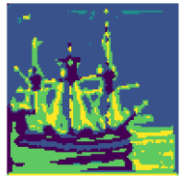
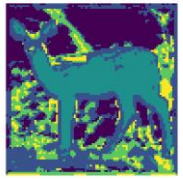
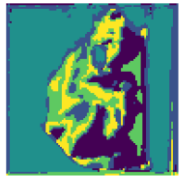
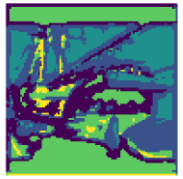
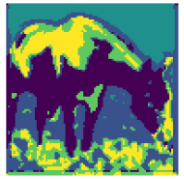






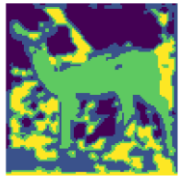
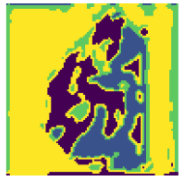
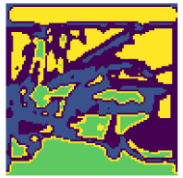
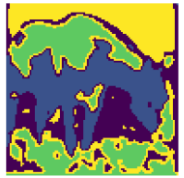
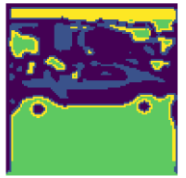

Argmax Output




SLIC Output



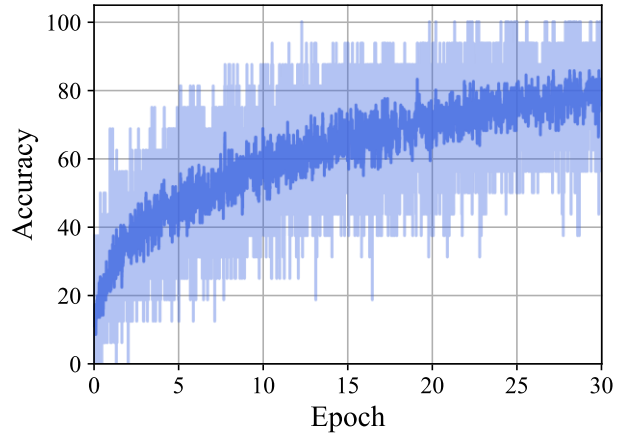







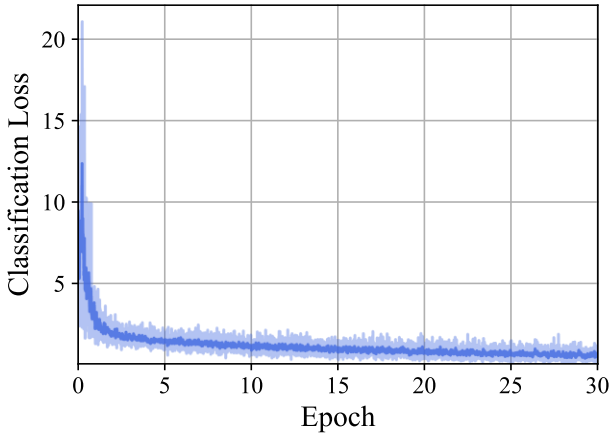
Epochs=30, LR=0.001, BS=16, Opt=SGD

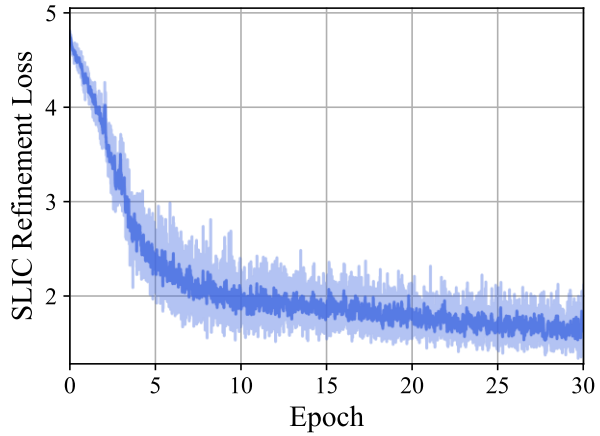
Results
Proposed Architecture 1: FCN with Classification Approach

Dataset: STL10

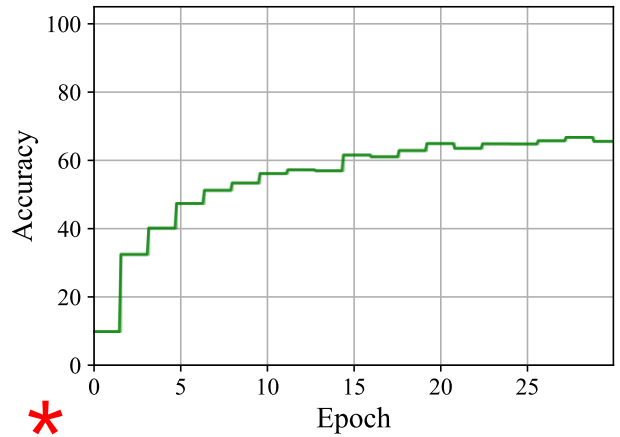
Training



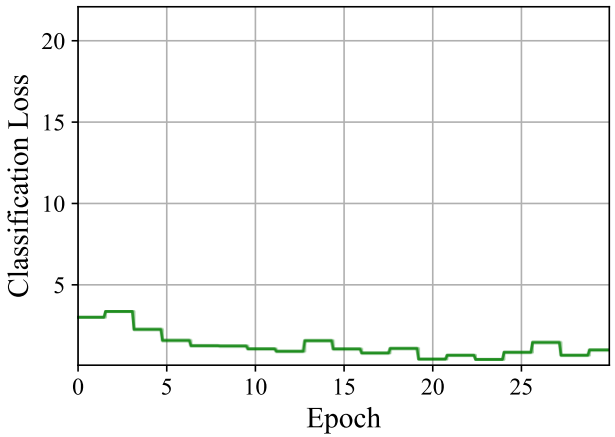


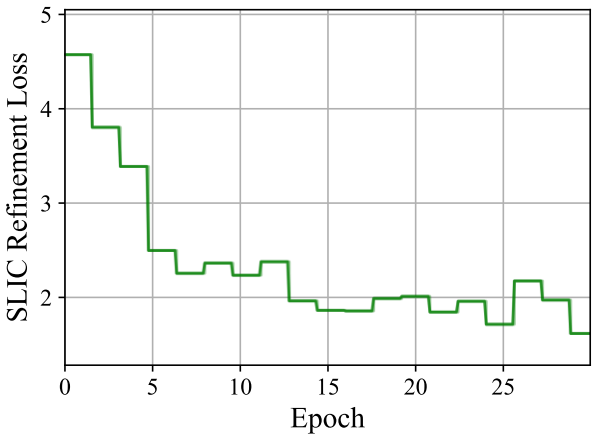


Testing



*

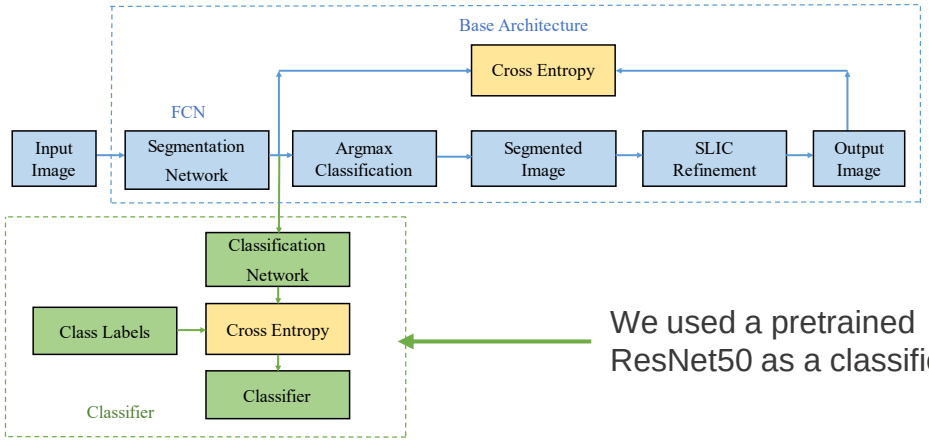




Epochs=30, LR=0.001, BS=16, Opt=SGD

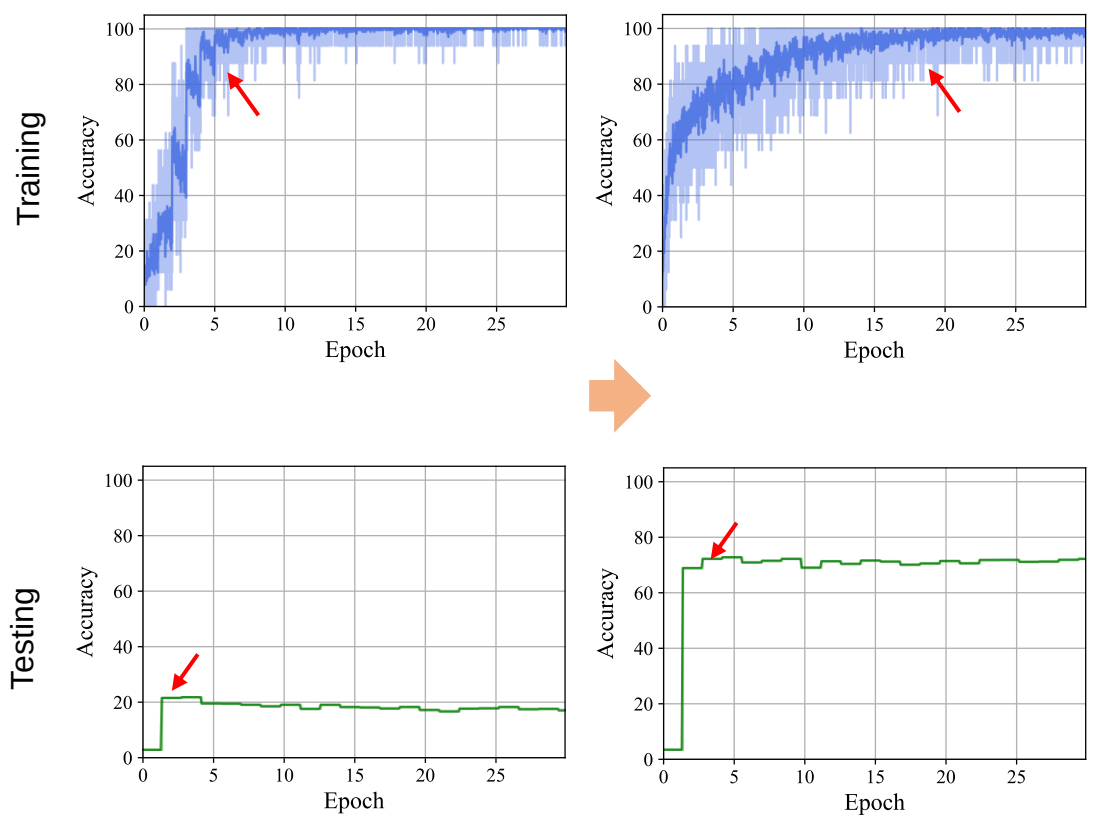
Results
Proposed Architecture 1: FCN with Classification Approach

Dataset: Pascal VOC



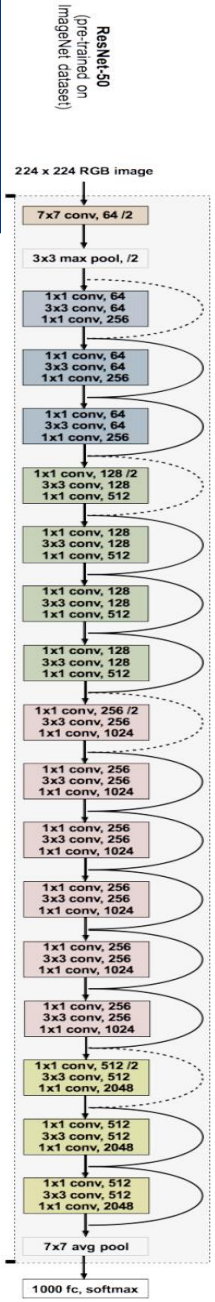
We used a pretrained ResNet50 as a classifier

- Initially four convolutional units were used for classification part of the Pascal VOC dataset
- The model was highly overfitting
- So, we used **a pretrained ResNet50** model (pretrained on ImageNet Dataset) to avoid over fitting , and used **Transfer Learning** method to boost up the training process



Before using ResNet50

After using ResNet50

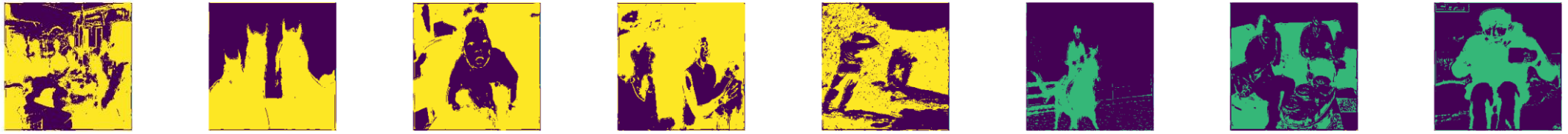


Results
Proposed Architecture 1: FCN with Classification Approach

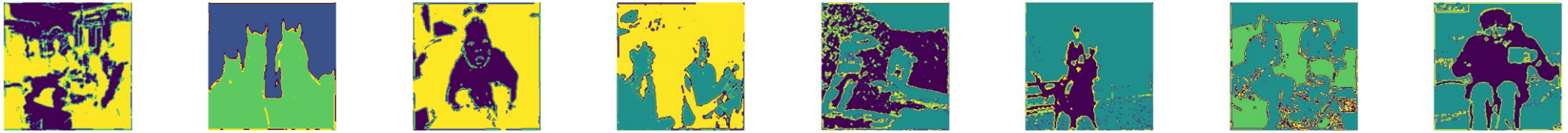
Dataset: Pascal VOC

Main Image

Argmax Output



SLIC Output

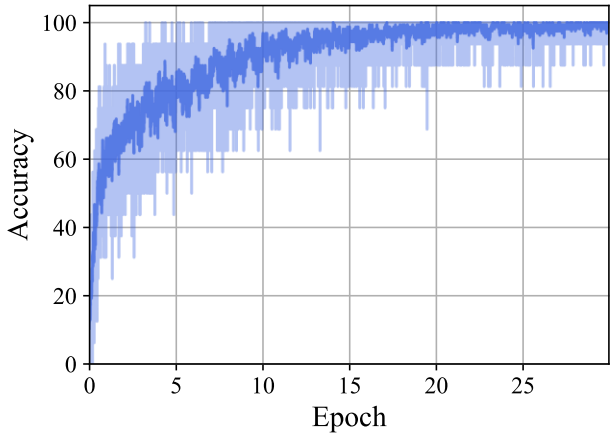


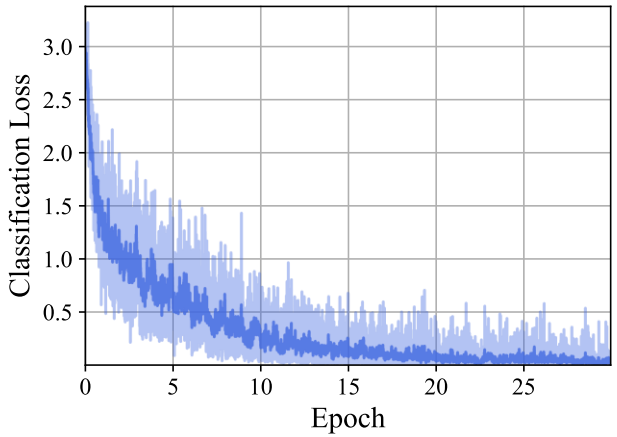
Epochs=30, LR=0.001, BS=16, Opt=SGD

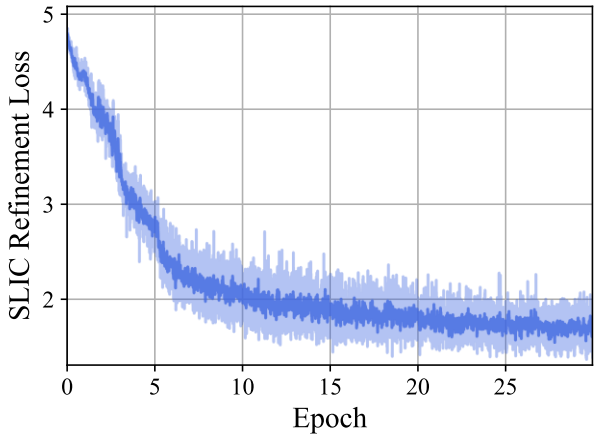
Results
Proposed Architecture 1: FCN with Classification Approach

Dataset: Pascal VOC

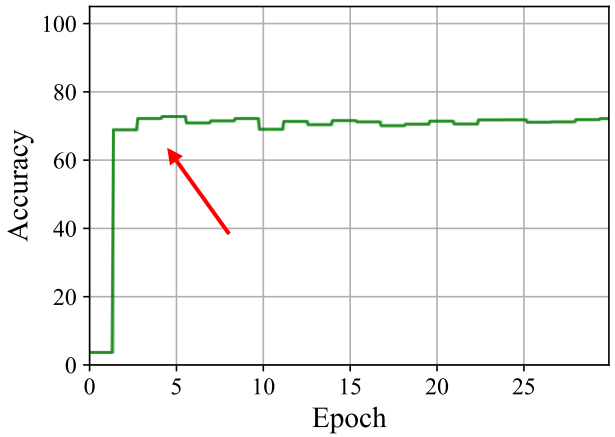
Training

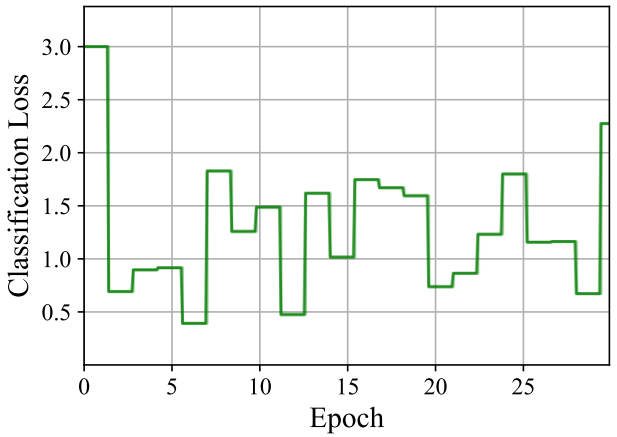


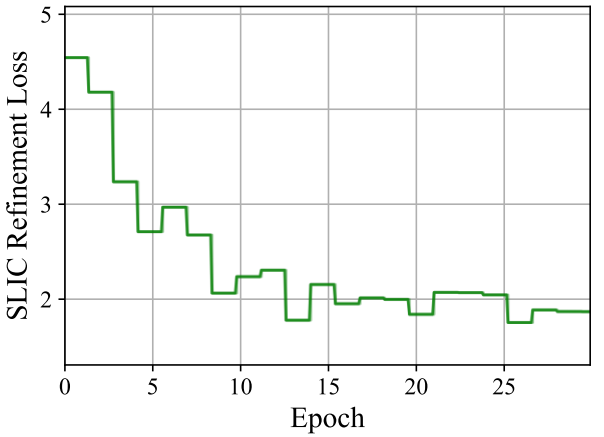




Testing



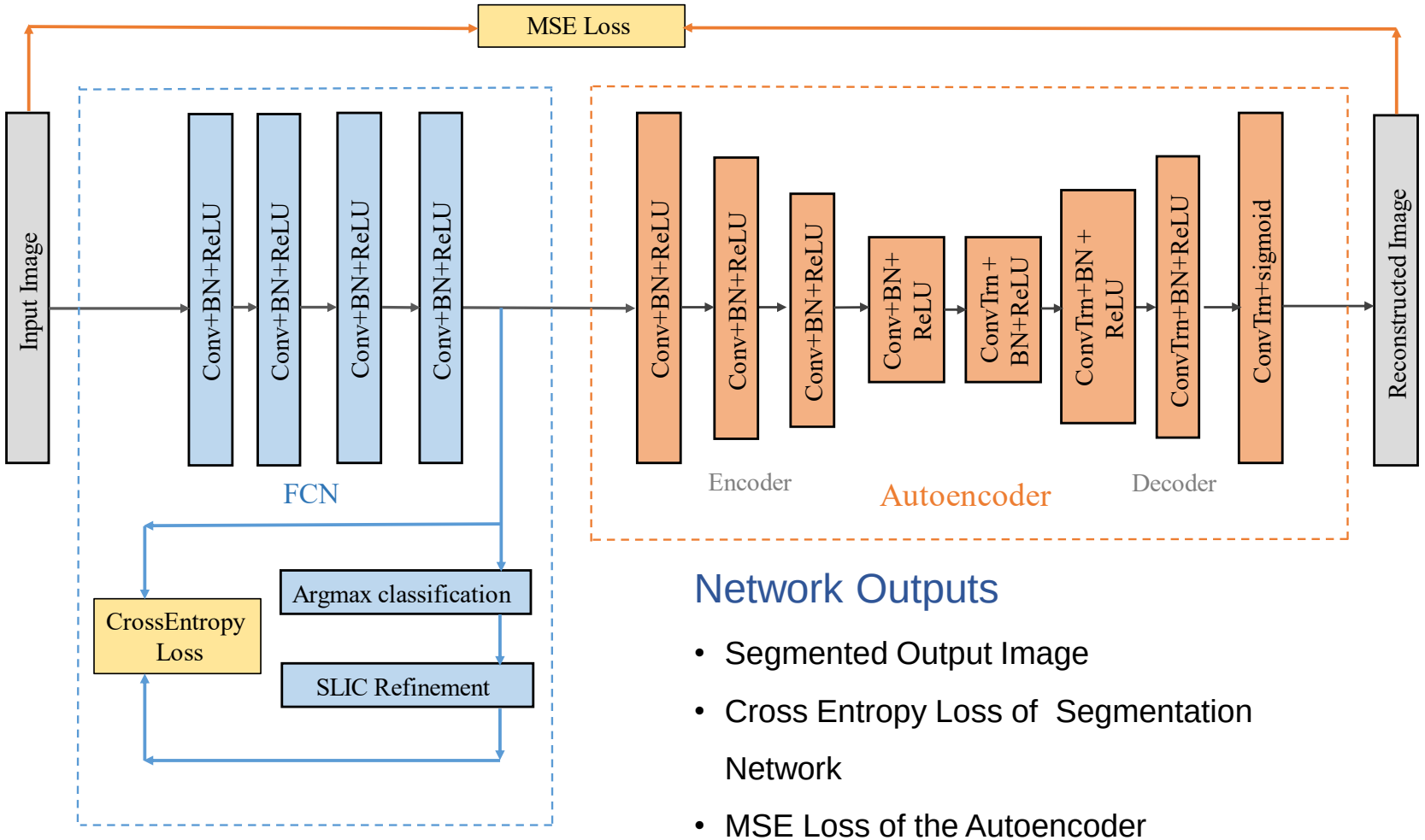




Epochs=30, LR=0.001, BS=16, Opt=SGD

Proposed Architecture 2 for Problem Solution

Proposed Architecture 2: FCN with Autoencoder Approach



Network Outputs

- Segmented Output Image
- Cross Entropy Loss of Segmentation Network
- MSE Loss of the Autoencoder
- Reconstructed Image

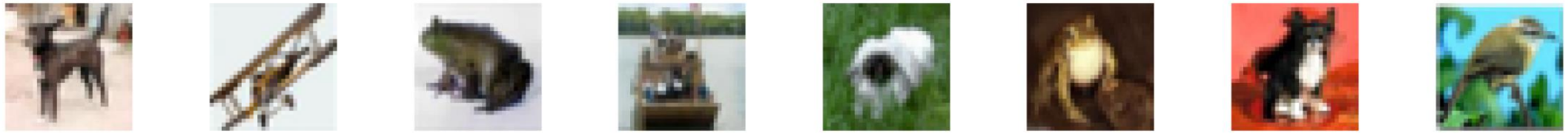
Hyperparameters for Network Tunning

Hyperparameters	Range / Value
Epochs	30, 40, 50, 60, 100
Normalization	minmax (0 to 1)
Activation Function	ReLU
Batch Size	5, 16, 32, 50, 100
Learning Rate	0.1, 0.01, 0.001, 0.0001
Optimizer	Adam, SGD
Initializer	Xavier Initializer
Scheduler	StepLR

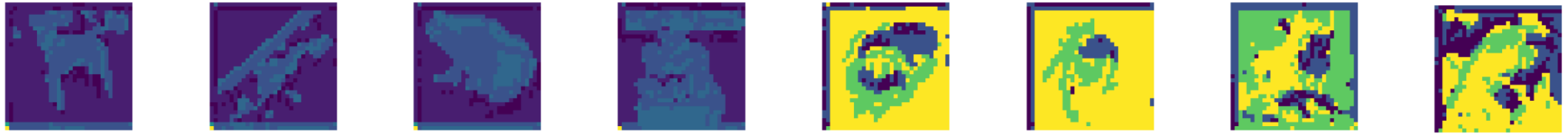
Results
Proposed Architecture 2: FCN with Autoencoder Approach

Dataset: CIFAR10

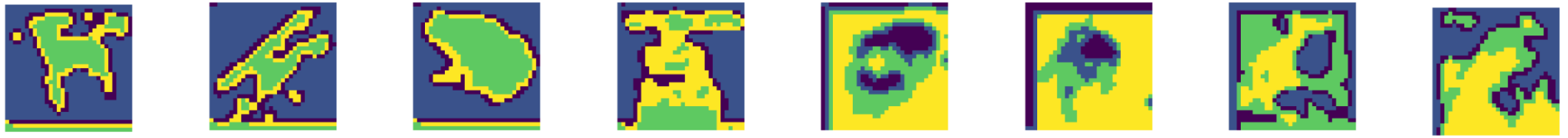
Main Image



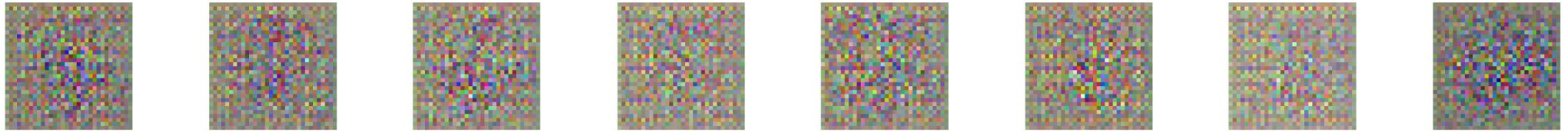
Argmax Output



SLIC Output



Reconstructed Image

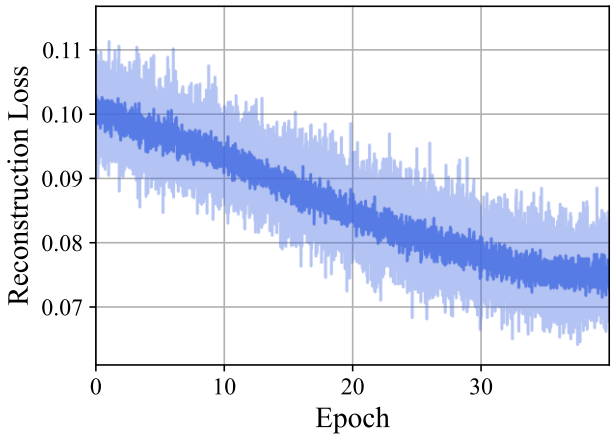


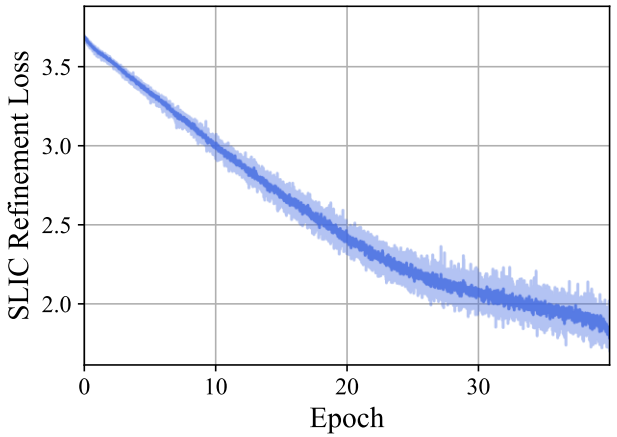
Epochs=40, LR=0.0001, BS=100, Opt=SGD

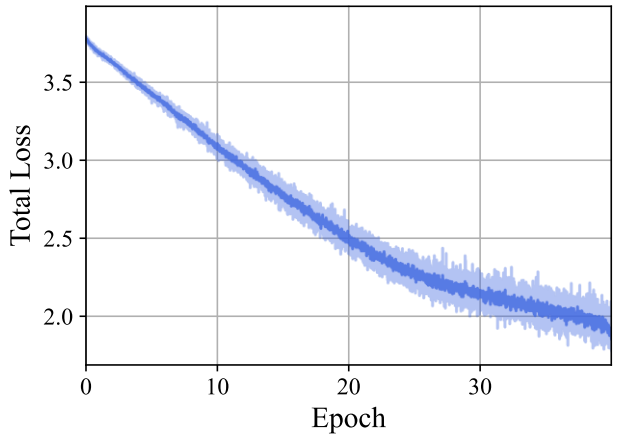
Results
Proposed Architecture 2: FCN with Autoencoder Approach

Dataset: CIFAR10

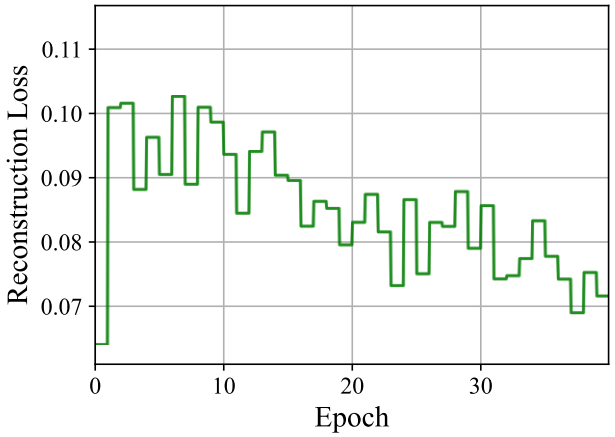
Training

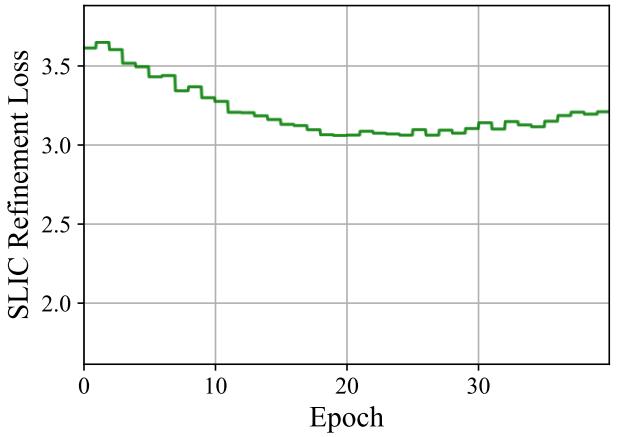


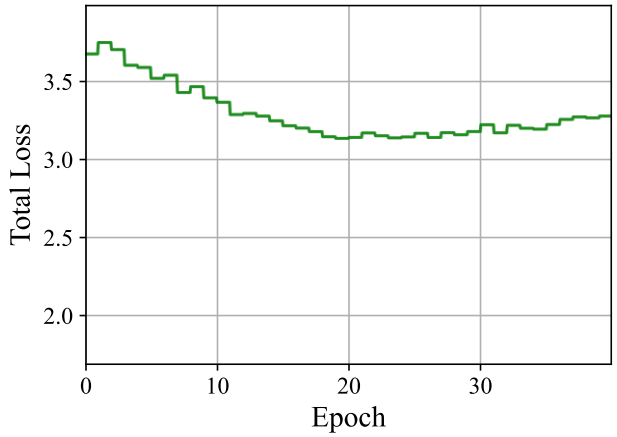




Testing







Epochs=40, LR=0.0001, BS=100, Opt=SGD

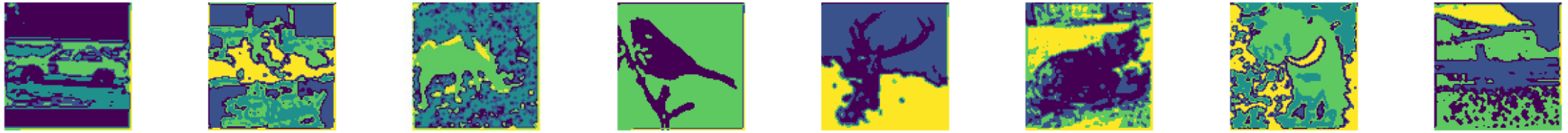
Results
Proposed Architecture 2: FCN with Autoencoder Approach

Dataset: STL10

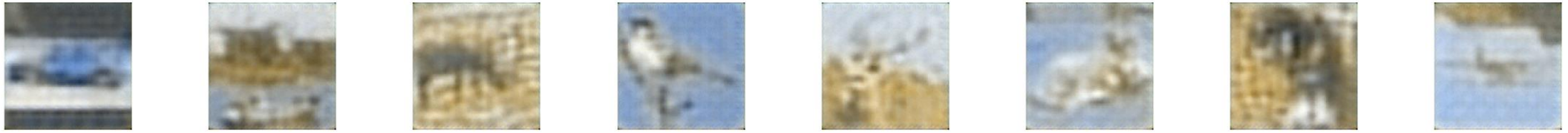
Main Image

Argmax Output

SLIC Output



Reconstructed Image

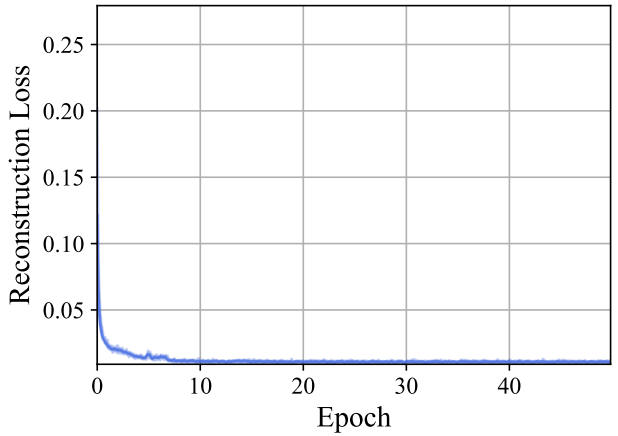


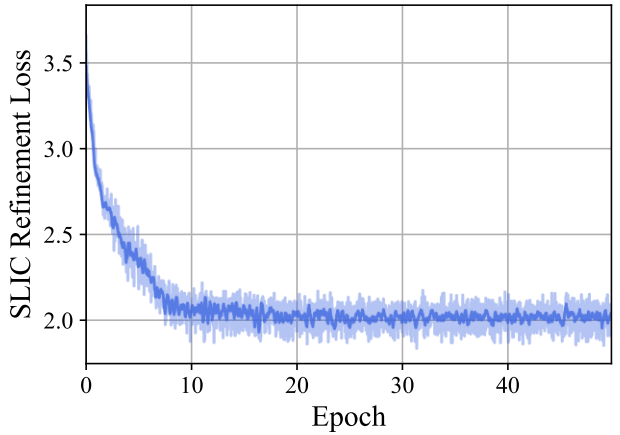
Epochs=50, LR=0.005, BS=100, Opt=Adam

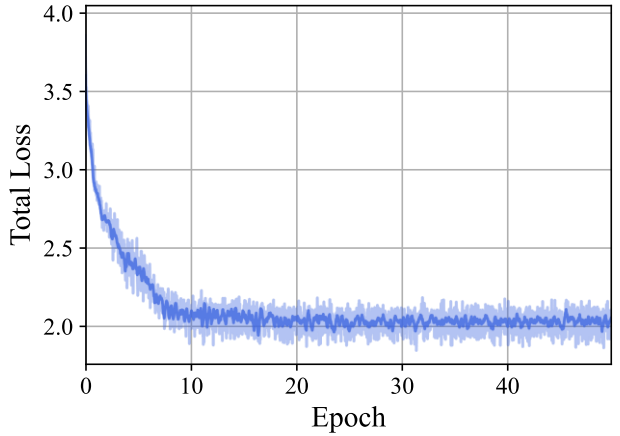
Results
Proposed Architecture 2: FCN with Autoencoder Approach

Dataset: STL10

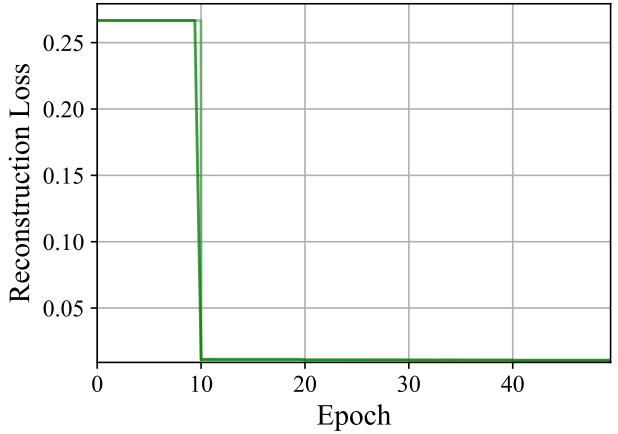
Training

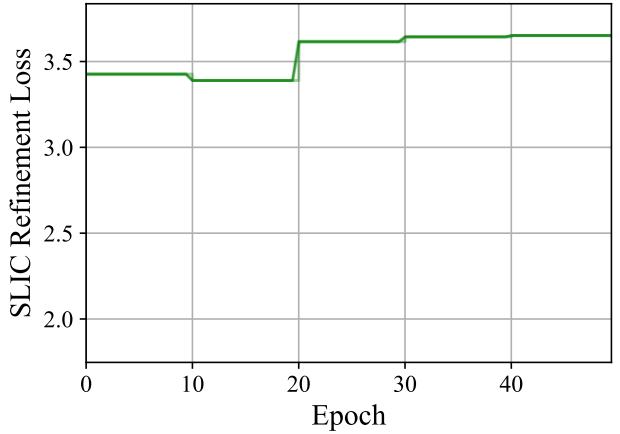


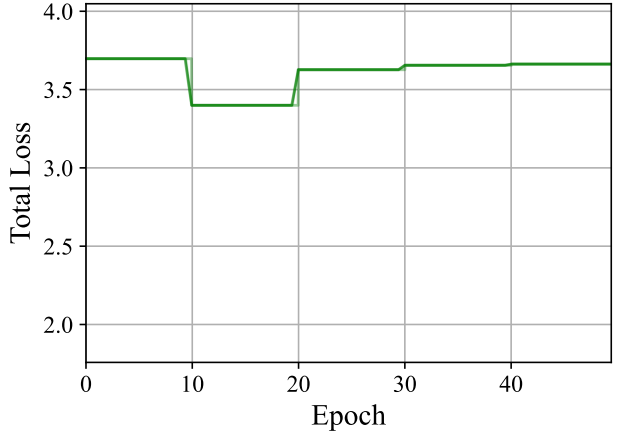




Testing







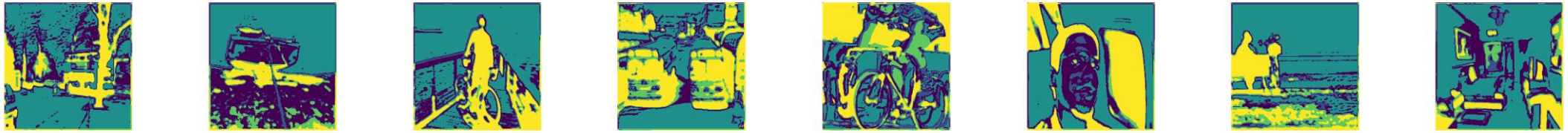
Epochs=50, LR=0.005, BS=100, Opt=Adam

Results
Proposed Architecture 2: FCN with Autoencoder Approach

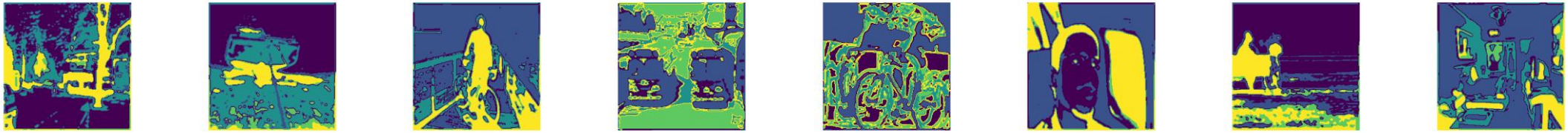
Dataset: Pascal VOC

Main Image

Argmax Output



SLIC Output



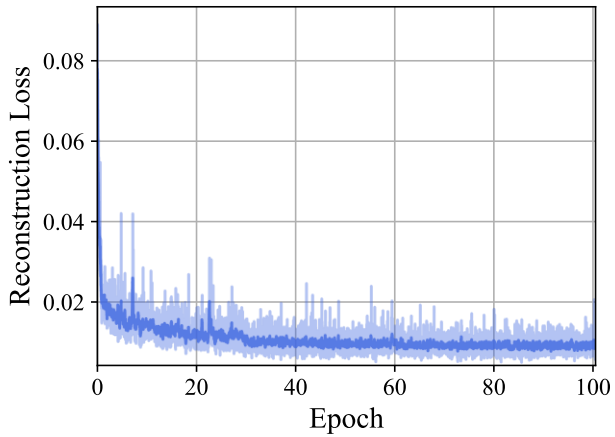
Reconstructed Image

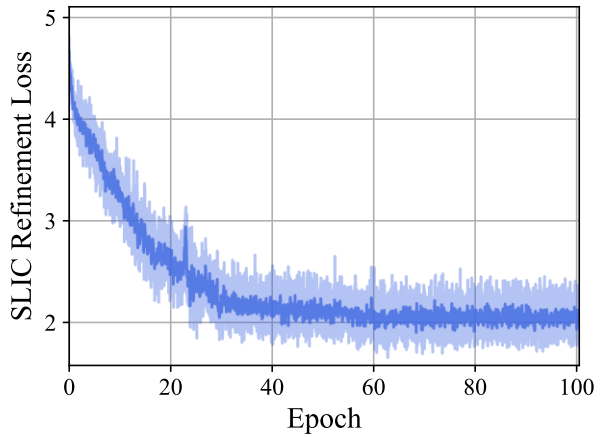
Epochs=100, LR=0.001, BS=16, Opt=Adam

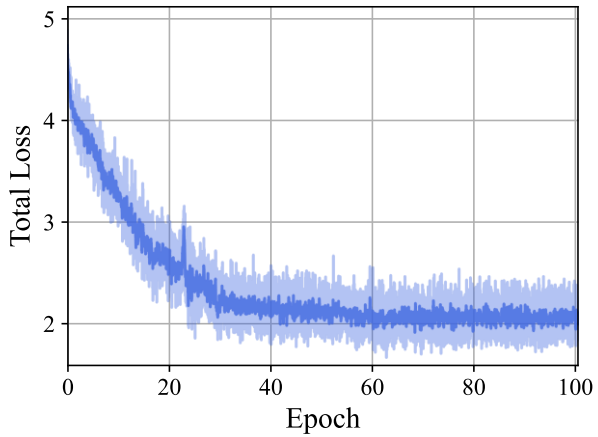
Results
Proposed Architecture 2: FCN with Autoencoder Approach

Dataset: Pascal VOC

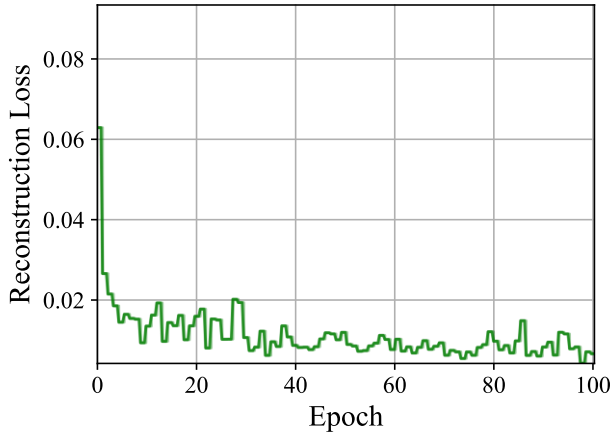
Training

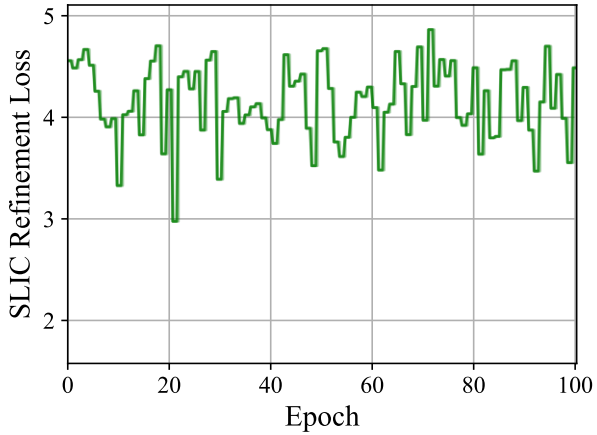


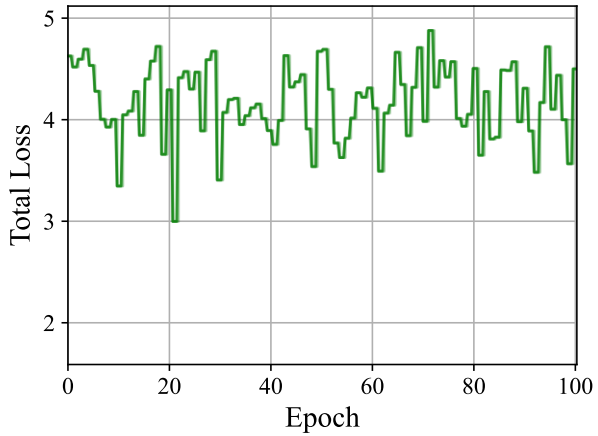




Testing

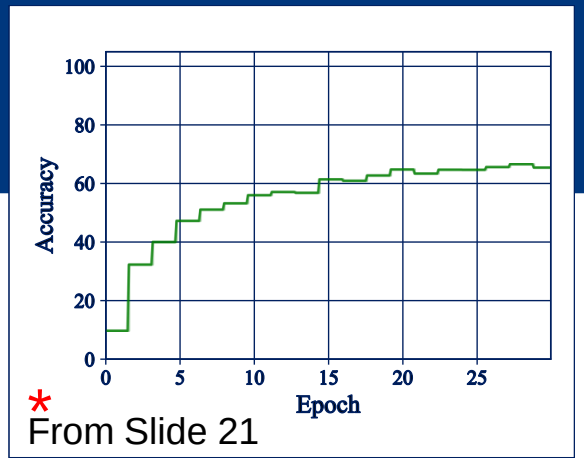






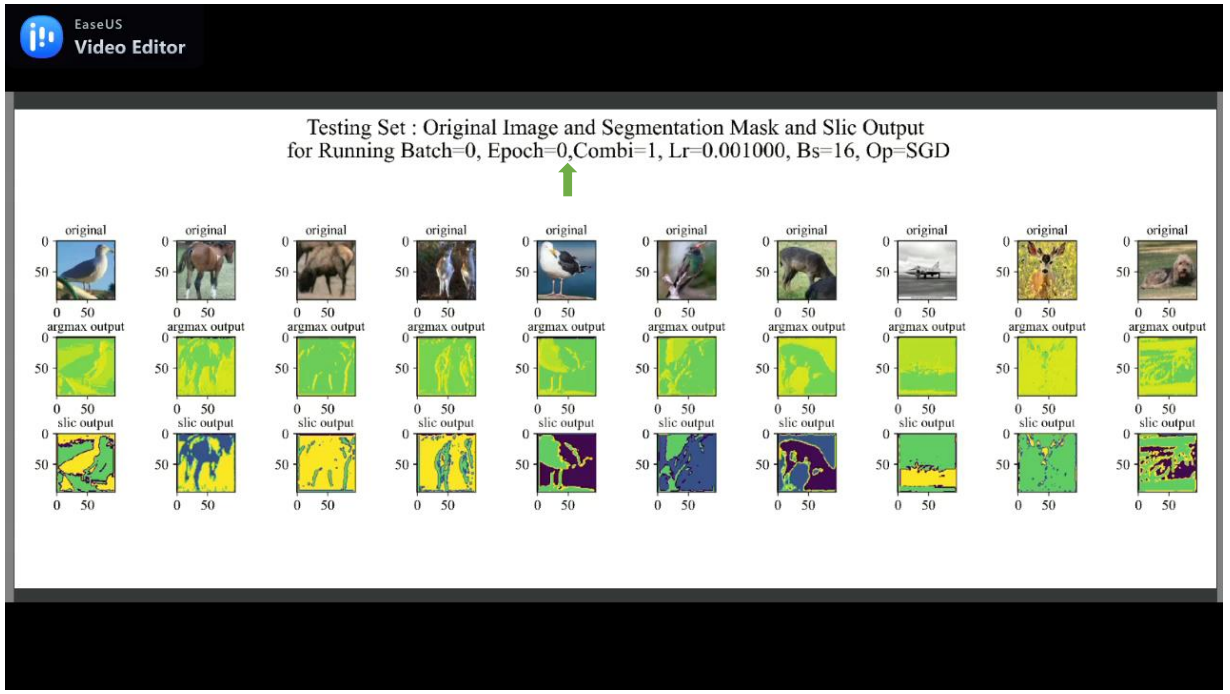
Epochs=100, LR=0.001, BS=16, Opt=Adam

Dataset: STL10



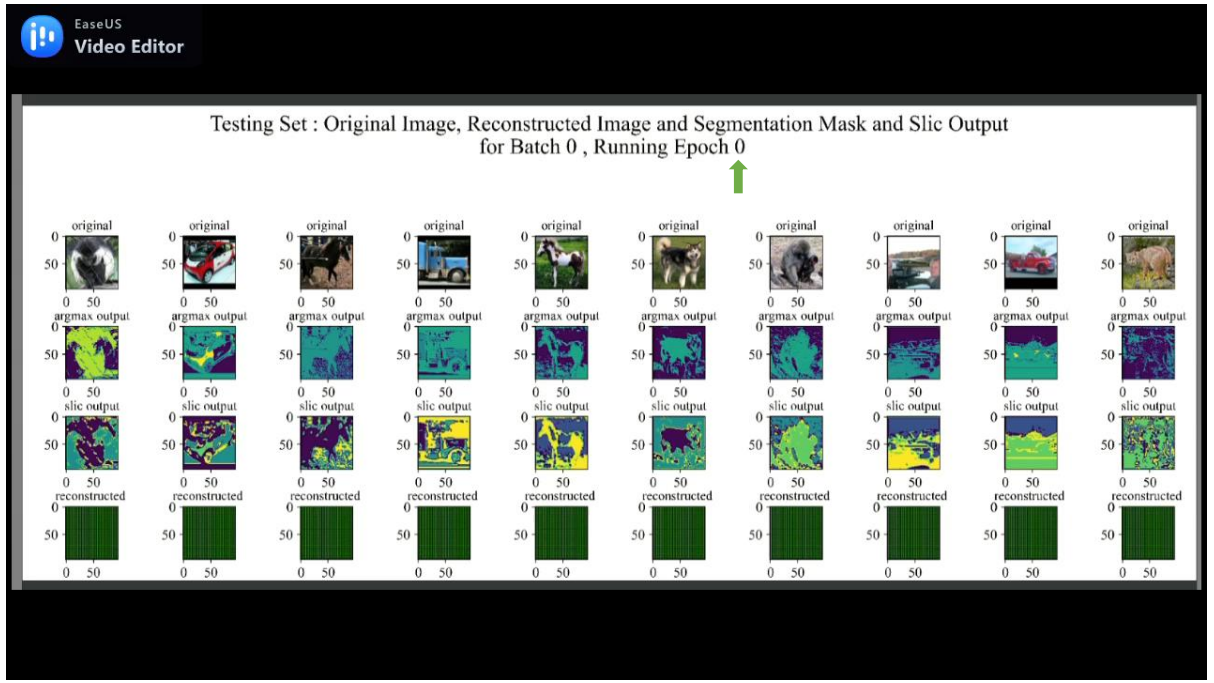
*From Slide 21

Effect of Using Classifier



Epochs=30, LR=0.001, BS=16, Opt=SGD

Effect of Using Autoencoder

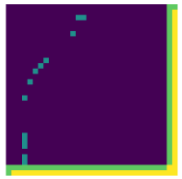
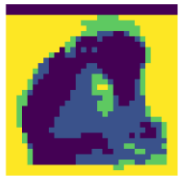
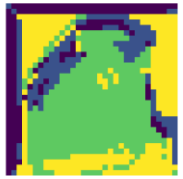
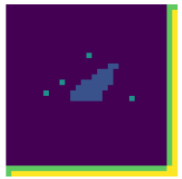
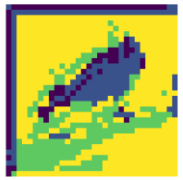
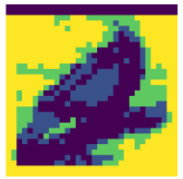
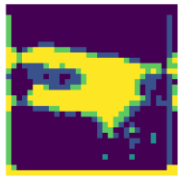
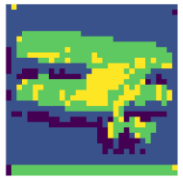
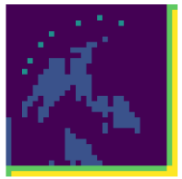
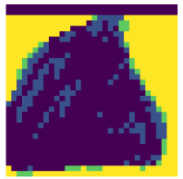
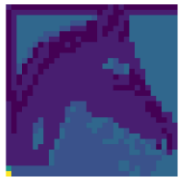


Epochs=50, LR=0.005, BS=100, Opt=Adam

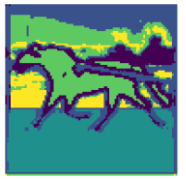
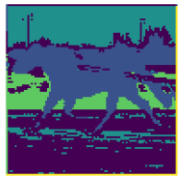
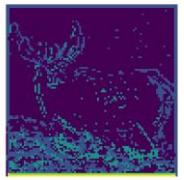
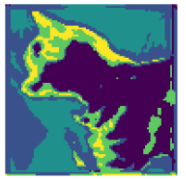
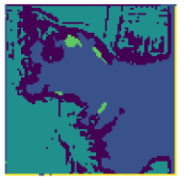
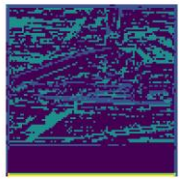
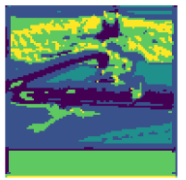
Both of the proposed approaches successfully restrict the network from merging away

Comparison of Results

Dataset: CIFAR10

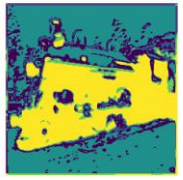
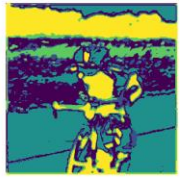
Main Image	Base Architecture	Proposed Approach 1	Proposed Approach 2
			
			
			
			

Dataset: STL10

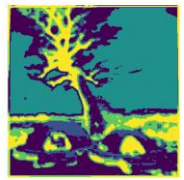
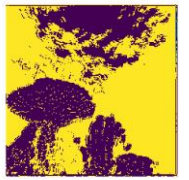
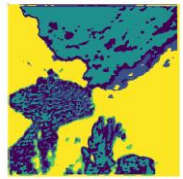
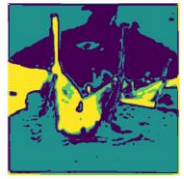
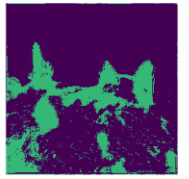
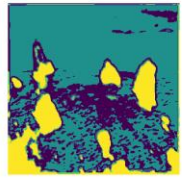
Main Image	Base Architecture	Proposed Approach 1	Proposed Approach 2
			
			
			
			

Comparison of Results

Dataset: Pascal VOC

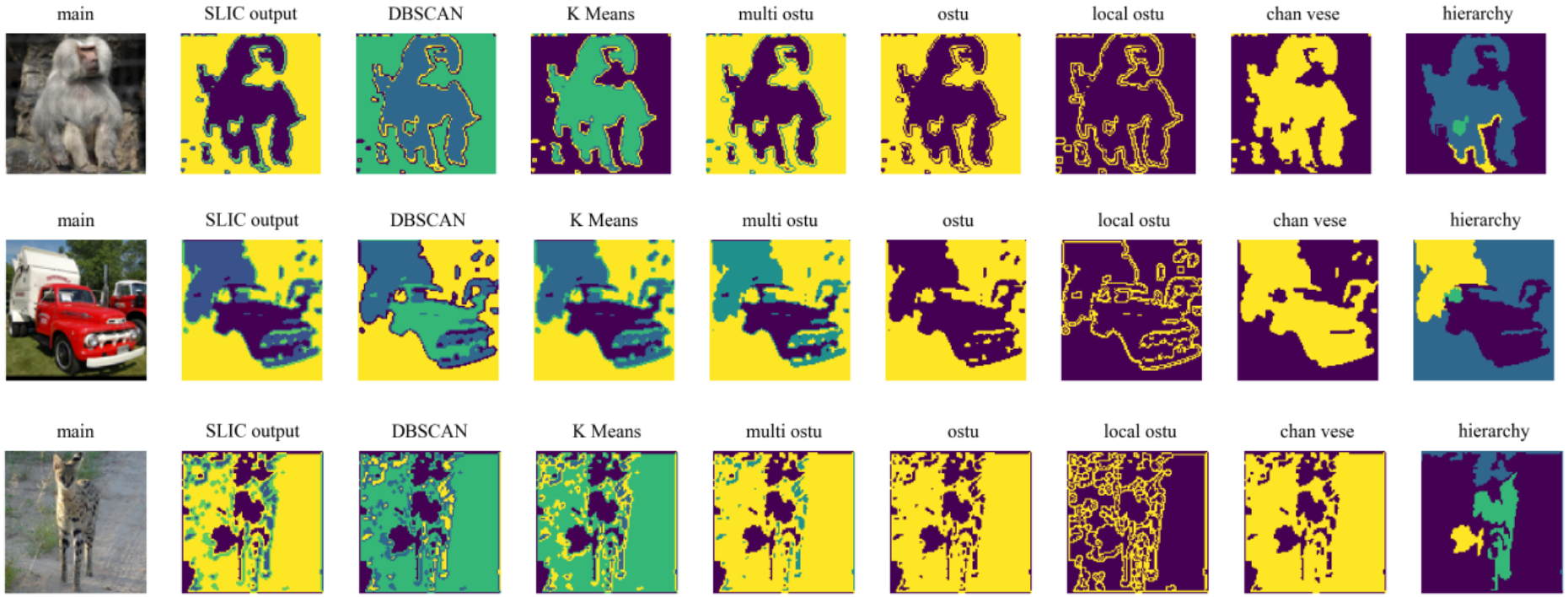
Main Image	Base Architecture	Proposed Approach 1	Proposed Approach 2
			
			
			
			

Dataset: BSDS500

Main Image	Base Architecture	Proposed Approach 1	Proposed Approach 2
			
			
			
			

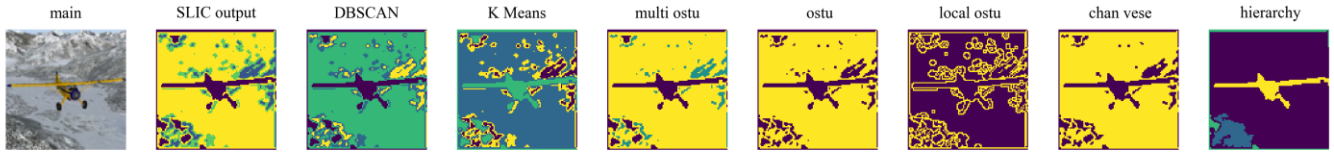
Future Work
Post Processing of Segmented Images

Different clustering techniques are tried to reduce noise of segmented images and to evaluate segmentation performance.

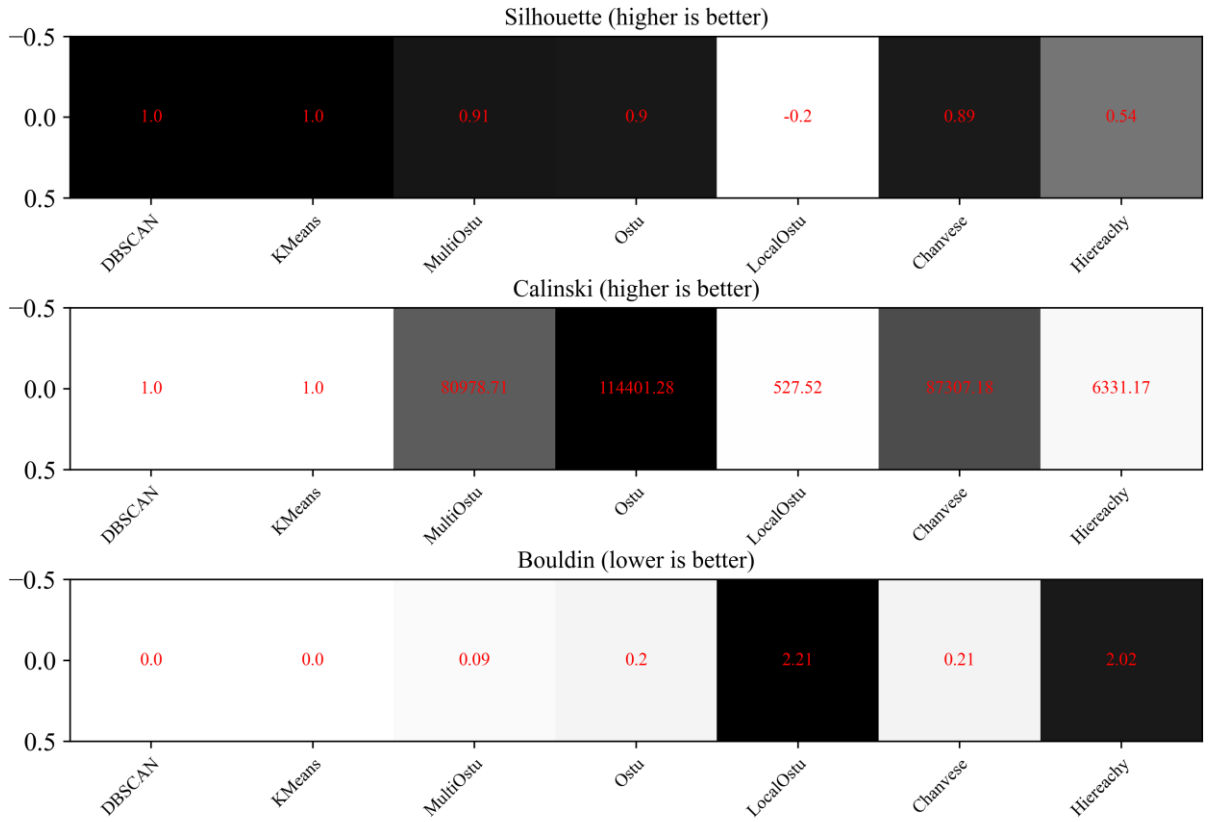


Future Work
Post Processing of Segmented Images

Comparison of Different Clustering Technique



We tried different clustering techniques to reduce noise of segmented images and evaluate performance



Silhouette Score

Calinski-Harabasz Index

Davies-Bouldin Index

Fachhochschule Südwestfalen
University of Applied Sciences

Conclusion

- Very less work has been done on unsupervised segmentation deep learning techniques.
- We took the idea of combining CNN with SLIC Refinements from Asako Kanezaki .
- And develop two more stable architectures.
- We also used the new architectures on three popular image datasets and found promising results.
- The learnt models are checked on BSDS500 dataset.
- In addition, to find the best post processing methods and to design a performance metric for segmented images, 7 different cluster techniques. Further research is needed to design performance metric of segmentation in fully unsupervised way.

Thank you!

Questions ?

Appendix

Total Number of Trainable Parameters

	CIFAR10	STL10	Pascal VOC
Base Architecture	24832	24832	24832
Proposed FCN with classifier approach	177,506	2,288,468	23,954,173
Proposed FCN with autoencoder approach	2,25,883	20,5115	6,09,915

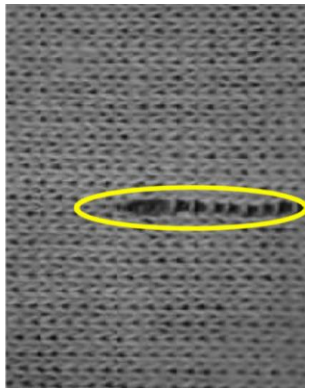
Approximate Training Time

	CIFAR10	STL10	Pascal VOC
Base Architecture	5:39:53 h(epoch 30)	7:15:10 h(epoch 30)	8:37:34h (epoch 30)
Proposed FCN with classifier approach	3:54:18 h(epoch 30)	6 hApprx. (epoch 30)	1 day, 8:37:21h(epoch 30)
Proposed FCN with autoencoder approach	7:26:49h (epoch 100)	9h Apprx. (epoch 50)	21:35:56h (epoch 100)

Appendix

Applications

- Road condition monitoring
- Surface defect inspection in industrial monitoring
- Fabric defect detection systems and methods
- Medical image analysis
- Anomaly detection on industrial products



Fabric defect detection systems and methods—A systematic literature review
Author links open overlay panelKazımHanbayaMuhammed FatihTalu

https://devisionx.com/industrial-applications/food-and-beverage/

Fachhochschule Südwestfalen
University of Applied Sciences

Layer Description of Base Architecture (Appendix)

Dataset: CIFAR10

Table 5.1: Description of convolutional neural network (CNN) architecture used in the CIFAR10 dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	32x32x3	64	3x3x3	1x1	1x1	32x32x64
Conv2+BN+Relu	32x32x64	32	3x3x64	1x1	0x0	32x32x32
Conv3+BN+Relu	32x32x32	64	3x3x32	1x1	1x1	32x32x64
Conv4+BN+Relu	32x32x64	32	3x3x64	1x1	0x0	32x32x32

```
+----------------+------------+
|    Modules     | Parameters |
+----------------+------------+
| seq1.0.weight  |    1728    |
|  seq1.0.bias   |     64     |
| seq1.1.weight  |     64     |
|  seq1.1.bias   |     64     |
| seq1.3.weight  |    2048    |
|  seq1.3.bias   |     32     |
| seq1.4.weight  |     32     |
|  seq1.4.bias   |     32     |
| seq1.6.weight  |   18432    |
|  seq1.6.bias   |     64     |
| seq1.7.weight  |     64     |
|  seq1.7.bias   |     64     |
| seq1.9.weight  |    2048    |
|  seq1.9.bias   |     32     |
| seq1.10.weight |     32     |
|  seq1.10.bias  |     32     |
+----------------+------------+
Total Trainable Params: 24832
```

Dataset: STL10

Table 5.3: Description of convolutional neural network (CNN) architecture used in STL10 dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	96x96x3	64	3x3x3	1x1	1x1	96x96x64
Conv2+BN+Relu	96x96x64	32	3x3x64	1x1	0x0	96x96x32
Conv3+BN+Relu	96x96x32	64	3x3x32	1x1	1x1	96x96x64
Conv4+BN+Relu	96x96x64	32	3x3x64	1x1	0x0	96x96x32

```
+----------------+------------+
|    Modules     | Parameters |
+----------------+------------+
| seq1.0.weight  |    1728    |
|  seq1.0.bias   |     64     |
| seq1.1.weight  |     64     |
|  seq1.1.bias   |     64     |
| seq1.3.weight  |    2048    |
|  seq1.3.bias   |     32     |
| seq1.4.weight  |     32     |
|  seq1.4.bias   |     32     |
| seq1.6.weight  |   18432    |
|  seq1.6.bias   |     64     |
| seq1.7.weight  |     64     |
|  seq1.7.bias   |     64     |
| seq1.9.weight  |    2048    |
|  seq1.9.bias   |     32     |
| seq1.10.weight |     32     |
|  seq1.10.bias  |     32     |
+----------------+------------+
Total Trainable Params: 24832
```

Fachhochschule Südwestfalen
University of Applied Sciences

Layer Description of Base Architecture (Appendix)

Dataset: Pascal VOC

Table 5.4: Description of convolutional neural network (CNN) architecture used in Pascal VOC dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	224x224x3	64	3x3x3	1x1	1x1	224x224x64
Conv2+BN+Relu	224x224x64	32	3x3x64	1x1	0x0	224x224x32
Conv3+BN+Relu	224x224x32	64	3x3x32	1x1	1x1	224x224x64
Conv4+BN+Relu	224x224x64	32	3x3x64	1x1	0x0	224x224x32

```
+----------------+------------+
|    Modules     | Parameters |
+----------------+------------+
| seq1.0.weight  |    1728    |
|  seq1.0.bias   |     64     |
| seq1.1.weight  |     64     |
|  seq1.1.bias   |     64     |
| seq1.3.weight  |    2048    |
|  seq1.3.bias   |     32     |
| seq1.4.weight  |     32     |
|  seq1.4.bias   |     32     |
| seq1.6.weight  |   18432    |
|  seq1.6.bias   |     64     |
| seq1.7.weight  |     64     |
|  seq1.7.bias   |     64     |
| seq1.9.weight  |    2048    |
|  seq1.9.bias   |     32     |
| seq1.10.weight |     32     |
|  seq1.10.bias  |     32     |
+----------------+------------+
Total Trainable Params: 24832
```

Layer Description of Proposed Approach 1 (Appendix)

Dataset: CIFAR10

Table 6.1: Segmentation network for CIFAR10 dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	32x32x3	64	3x3x3	1x1	1x1	32x32x64
Conv2+BN+Relu	32x32x64	128	3x3x64	1x1	1x1	32x32x128
Conv3+BN+Relu	32x32x128	64	3x3x128	1x1	0x0	32x32x64
Conv4+BN+Relu	32x32x64	128	3x3x64	1x1	0x0	32x32x128

Table 6.2: Classification network for CIFAR10 dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	32x32x128	12	3x3x12	1x1	1x1	32x32x12
Conv2+BN+Relu	32x32x12	12	3x3x12	1x1	1x1	32x32x12
Max Pool2d	32x32x12	1	2x2	2x2	0	16x16x12
Conv3+BN+Relu	16x16x12	24	3x3x24	1x1	1x1	16x16x24
Conv4+BN+Relu	16x16x24	24	3x3x24	1x1	1x1	16x16x24
Dropout (P=0.2)	16x16x24	-	-	-	-	16x16x24
FC Layer	6144	-	-	-	-	10

```
+---------------+------------+
|    Modules    | Parameters |
+---------------+------------+
|  seq.0.weight |    1728    |
|   seq.0.bias  |     64     |
|  seq.1.weight |     64     |
|   seq.1.bias  |     64     |
|  seq.3.weight |   73728    |
|   seq.3.bias  |    128     |
|  seq.4.weight |    128     |
|   seq.4.bias  |    128     |
|  seq.6.weight |    8192    |
|   seq.6.bias  |     64     |
|  seq.7.weight |     64     |
|   seq.7.bias  |     64     |
|  seq.9.weight |    8192    |
|   seq.9.bias  |    128     |
| seq.10.weight |    128     |
|  seq.10.bias  |    128     |
+---------------+------------+
Total Trainable Params: 92992
```

```
+-------------------+------------+
|      Modules      | Parameters |
+-------------------+------------+
|    conv1.weight   |   13824    |
|     conv1.bias    |     12     |
| batchnorm2.weight |     12     |
|  batchnorm2.bias  |     12     |
|    conv2.weight   |    1296    |
|     conv2.bias    |     12     |
| batchnorm3.weight |     12     |
|  batchnorm3.bias  |     12     |
|    conv3.weight   |    2592    |
|     conv3.bias    |     24     |
| batchnorm4.weight |     24     |
|  batchnorm4.bias  |     24     |
|    conv4.weight   |    5184    |
|     conv4.bias    |     24     |
|     fc.weight     |   61440    |
|      fc.bias      |     10     |
+-------------------+------------+
Total Trainable Params: 84514
```

Layer Description of Proposed Approach 1 (Appendix)

Dataset: STl10

Table 6.3: Segmentation network for STL10 dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	32x32x3	64	3x3x64	1x1	1x1	32x32x64
Conv2+BN+Relu	32x32x64	128	3x3x128	1x1	1x1	32x32x128
Conv3+BN+Relu	32x32x128	256	3x3x256	1x1	1x1	32x32x256
Conv4+BN+Relu	32x32x256	96	3x3x96	1x1	0x0	32x32x96

Table 6.4: Classification network for STL10 dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	96x96x96	32	3x3x3	1x1	1x1	96x96x32
Conv2+BN+Relu	96x96x32	64	3x3x64	1x1	1x1	96x96x64
Max Pool2d	96x96x64	1	2x2	2x2	0	48x48x64
Conv3+BN+Relu	48x48x64	128	3x3x128	1x1	1x1	48x48x128
Dropout(P= 0.05)	48x48x128	-	-	-	-	48x48x128
Max Pool2d	48x48x128	1	2x2	2x2	0	24x24x128
Conv4+BN+Relu	24x24x128	256	3x3x256	1x1	1x1	24x24x256
Conv5+BN+Relu	24x24x256	256	3x3x256	1x1	1x1	24x24x256
Max Pool2d	24x24x128	1	2x2	2x2	0	12x12x256
Dropout (P= 0.1)	12x12x256	-	-	-	-	12x12x256
FC Layer	36864	-	-	-	-	10

Modules	Parameters
seq.0.weight	1728
seq.0.bias	64
seq.1.weight	64
seq.1.bias	64
seq.3.weight	73728
seq.3.bias	128
seq.4.weight	128
seq.4.bias	128
seq.6.weight	294912
seq.6.bias	256
seq.7.weight	256
seq.7.bias	256
seq.9.weight	24576
seq.9.bias	96
seq.10.weight	96
seq.10.bias	96

Total Trainable Params: 396576

Modules	Parameters
conv_layer.0.weight	27648
conv_layer.0.bias	32
conv_layer.1.weight	32
conv_layer.1.bias	32
conv_layer.3.weight	18432
conv_layer.3.bias	64
conv_layer.4.weight	64
conv_layer.4.bias	64
conv_layer.7.weight	73728
conv_layer.7.bias	128
conv_layer.8.weight	128
conv_layer.8.bias	128
conv_layer.11.weight	147456
conv_layer.11.bias	128
conv_layer.12.weight	128
conv_layer.12.bias	128
conv_layer.15.weight	294912
conv_layer.15.bias	256
conv_layer.16.weight	256
conv_layer.16.bias	256
conv_layer.18.weight	589824
conv_layer.18.bias	256
conv_layer.19.weight	256
conv_layer.19.bias	256
fc_layer.1.weight	737280
fc_layer.1.bias	20

Total Trainable Params: 1891892

Layer Description of Proposed Approach 1 (Appendix)

Dataset: Pascal VOC

Table 6.5: Segmentation network for Pascal VOC dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	32x32x3	64	3x3x64	1x1	1x1	32x32x64
Conv2+BN+Relu	32x32x64	128	3x3x128	1x1	1x1	32x32x128
Conv3+BN+Relu	32x32x128	256	3x3x256	1x1	1x1	32x32x256
Conv4+BN+Relu	32x32x256	96	3x3x96	1x1	0x0	32x32x96

Table 6.6: Classification network for Pascal VOC dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	224x224x96	64	3x3x64	1x1	0x0	224x224x64
Dropout (P=0.05)	224x224x64	-	-	-	-	224x224x64
Conv2+BN+Relu	224x224x64	32	3x3x132	1x1	0x0	224x224x32
Dropout (P=0.05)	224x224x32	-	-	-	-	224x224x32
Conv3+BN+Relu	224x224x32	3	3x3x3	1x1	0x0	224x224x3
Resnet50 [74]						

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[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				

Total Trainable Parameters: 23557597

Modules	Parameters
seq.0.weight	1728
seq.0.bias	64
seq.1.weight	64
seq.1.bias	64
seq.3.weight	73728
seq.3.bias	128
seq.4.weight	128
seq.4.bias	128
seq.6.weight	294912
seq.6.bias	256
seq.7.weight	256
seq.7.bias	256
seq.9.weight	24576
seq.9.bias	96
seq.10.weight	96
seq.10.bias	96

Total Trainable Params: 396576

Layer Description of Proposed Approach 2 (Appendix)

Dataset: CIFAR10

Table 7.1: Segmentation network for CIFAR10 dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	32x32x3	64	3x3x3	1x1	1x1	32x32x64
Conv2+BN+Relu	32x32x64	32	3x3x64	1x1	0x0	32x32x32
Conv3+BN+Relu	32x32x32	64	3x3x32	1x1	1x1	32x32x64
Conv4+BN+Relu	32x32x64	32	3x3x64	1x1	0x0	32x32x32

Table 7.2: Autoencoder network for CIFAR10 dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	32x32x32	12	4x4x12	2x2	1x1	16x16x12
Conv2+BN+Relu	16x16x12	24	4x4x24	2x2	1x1	8x8x24
Conv3+BN+Relu	8x8x24	48	4x4x48	2x2	1x1	4x4x48
Conv4+BN+Relu	4x4x48	96	4x4x96	2x2	1x1	2x2x96
ConvTr1+BN+Relu	2x2x96	48	4x4x48	2x2	1x1	4x4x48
ConvTr2+BN+Relu	4x4x48	24	4x4x24	2x2	1x1	8x8x24
ConvTr3+BN+Relu	8x8x24	12	4x4x12	2x2	1x1	16x16x12
ConvTr4+BN+Relu	16x16x12	3	4x4x3	2x2	1x1	32x32x3

```
+----------------+------------+
|    Modules     | Parameters |
+----------------+------------+
| seq1.0.weight  |    1728    |
|  seq1.0.bias   |     64     |
| seq1.1.weight  |     64     |
|  seq1.1.bias   |     64     |
| seq1.3.weight  |    2048    |
|  seq1.3.bias   |     32     |
| seq1.4.weight  |     32     |
|  seq1.4.bias   |     32     |
| seq1.6.weight  |   18432    |
|  seq1.6.bias   |     64     |
| seq1.7.weight  |     64     |
|  seq1.7.bias   |     64     |
| seq1.9.weight  |    2048    |
|  seq1.9.bias   |     32     |
| seq1.10.weight |     32     |
|  seq1.10.bias  |     32     |
| seq2.0.weight  |    6144    |
|  seq2.0.bias   |     12     |
| seq2.1.weight  |     12     |
|  seq2.1.bias   |     12     |
| seq2.3.weight  |    4608    |
|  seq2.3.bias   |     24     |
| seq2.4.weight  |     24     |
|  seq2.4.bias   |     24     |
| seq2.6.weight  |   18432    |
|  seq2.6.bias   |     48     |
| seq2.7.weight  |     48     |
|  seq2.7.bias   |     48     |
| seq2.9.weight  |   73728    |
|  seq2.9.bias   |     96     |
| seq2.10.weight |     96     |
|  seq2.10.bias  |     96     |
| seq2.12.weight |   73728    |
|  seq2.12.bias  |     48     |
| seq2.13.weight |     48     |
|  seq2.13.bias  |     48     |
| seq2.15.weight |   18432    |
|  seq2.15.bias  |     24     |
| seq2.16.weight |     24     |
|  seq2.16.bias  |     24     |
| seq2.18.weight |    4608    |
|  seq2.18.bias  |     12     |
| seq2.19.weight |     12     |
|  seq2.19.bias  |     12     |
| seq2.21.weight |    576     |
|  seq2.21.bias  |     3      |
+----------------+------------+
Total Trainable Params: 225883
```

Layer Description of Proposed Approach 2 (Appendix)

Dataset: STL10

Table 7.3: Segmentation network for STL10 dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	32x32x3	64	3x3x3	1x1	1x1	32x32x64
Conv2+BN+Relu	32x32x64	32	3x3x64	1x1	0x0	32x32x32

Table 7.4: Autoencoder Part fir STL10 dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	96x96x32	12	4x4x12	2x2	1x1	48x48x12
Conv2+BN+Relu	48x48x12	24	4x4x24	2x2	1x1	24x34x24
Conv3+BN+Relu	24x24x24	48	4x4x48	2x2	1x1	12x12x48
Conv4+BN+Relu	12x12x48	96	4x4x96	2x2	1x1	6x6x96
ConvTr1+BN+Relu	6x6x96	48	4x4x48	2x2	1x1	12x12x48
ConvTr2+BN+Relu	12x12x48	24	4x4x24	2x2	1x1	24x24x24
ConvTr3+BN+Relu	24x24x24	12	4x4x12	2x2	1x1	48x48x12
ConvTr4+BN+Relu	48x48x12	3	4x4x3	2x2	1x1	96x96x3

Modules	Parameters
seq1.0.weight	1728
seq1.0.bias	64
seq1.1.weight	64
seq1.1.bias	64
seq1.3.weight	2048
seq1.3.bias	32
seq1.4.weight	32
seq1.4.bias	32
seq2.0.weight	6144
seq2.0.bias	12
seq2.1.weight	12
seq2.1.bias	12
seq2.3.weight	4608
seq2.3.bias	24
seq2.4.weight	24
seq2.4.bias	24
seq2.6.weight	18432
seq2.6.bias	48
seq2.7.weight	48
seq2.7.bias	48
seq2.9.weight	73728
seq2.9.bias	96
seq2.10.weight	96
seq2.10.bias	96
seq2.12.weight	73728
seq2.12.bias	48
seq2.13.weight	48
seq2.13.bias	48
seq2.15.weight	18432
seq2.15.bias	24
seq2.16.weight	24
seq2.16.bias	24
seq2.18.weight	4608
seq2.18.bias	12
seq2.19.weight	12
seq2.19.bias	12
seq2.21.weight	576
seq2.21.bias	3

Total Trainable Params: 205115

Layer Description of Proposed Approach 2 (Appendix)

Dataset: Pascal VOC

Table 7.5: Segmentation network of Pascal VOC dataset

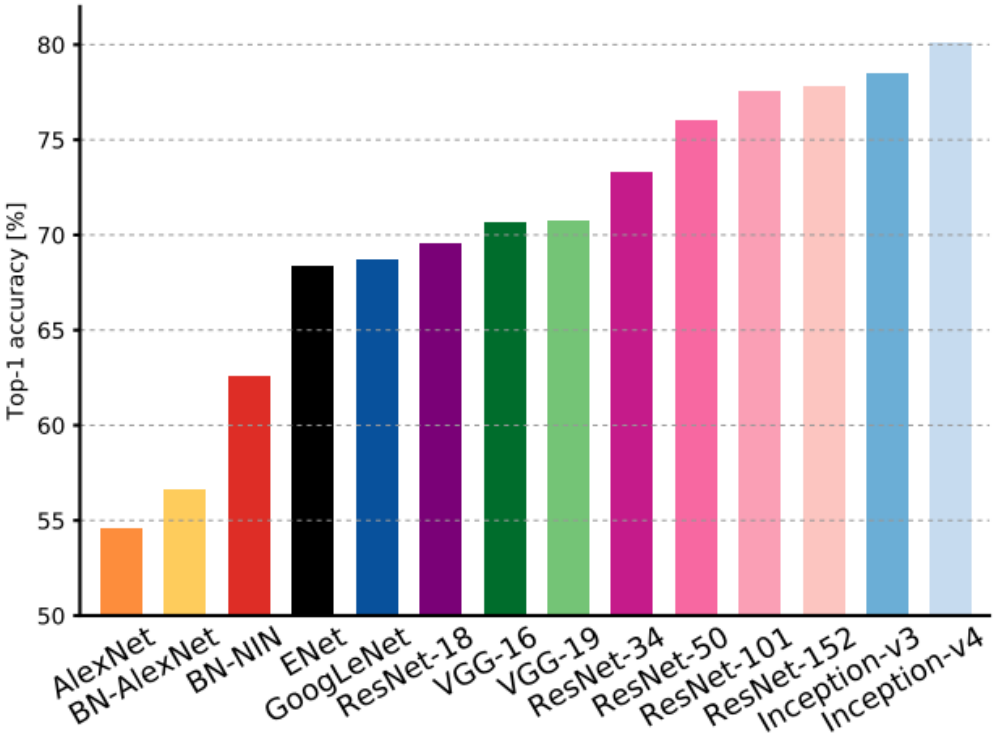
Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	32x32x3	64	3x3x64	1x1	1x1	32x32x64
Conv2+BN+Relu	32x32x64	128	3x3x128	1x1	1x1	32x32x128
Conv3+BN+Relu	32x32x128	256	3x3x256	1x1	1x1	32x32x256
Conv4+BN+Relu	32x32x256	96	3x3x96	1x1	0x0	32x32x96

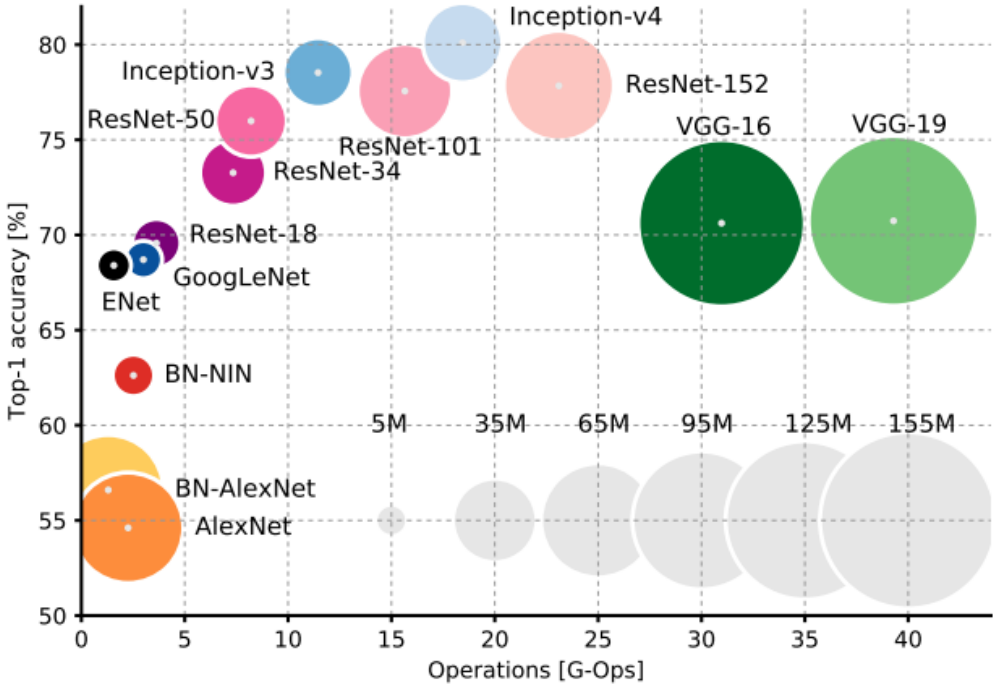
Table 7.6: Autoencoder part of Pascal VOC dataset

Operation Layer	Input Size (HxWxC)	Filter Numbers	Each Filter Size	Stride Value	Padding Value	Size of Feature Map (HxWxC)
Conv1+BN+Relu	224x224x32	12	4x4x12	2x2	1x1	112x112x12
Conv2+BN+Relu	112x112x12	24	4x4x24	2x2	1x1	56x56x24
Conv3+BN+Relu	56x56x24	48	4x4x48	2x2	1x1	28x28x48
Conv4+BN+Relu	28x28x48	96	4x4x96	2x2	1x1	14x14x96
ConvTr1+BN+Relu	14x14x96	48	4x4x48	2x2	1x1	28x28x48
ConvTr2+BN+Relu	28x28x48	24	4x4x24	2x2	1x1	56x56x24
ConvTr3+BN+Relu	56x56x24	12	4x4x12	2x2	1x1	112x112x12
ConvTr4+BN+Relu	112x112x12	3	4x4x3	2x2	1x1	224x224x3

```
+----------------+------------+
|    Modules     | Parameters |
+----------------+------------+
| seq1.0.weight  |    1728    |
|  seq1.0.bias   |     64     |
| seq1.1.weight  |     64     |
|  seq1.1.bias   |     64     |
| seq1.3.weight  |   73728    |
|  seq1.3.bias   |    128     |
| seq1.4.weight  |    128     |
|  seq1.4.bias   |    128     |
| seq1.6.weight  |   294912   |
|  seq1.6.bias   |    256     |
| seq1.7.weight  |    256     |
|  seq1.7.bias   |    256     |
| seq1.9.weight  |   24576    |
|  seq1.9.bias   |     96     |
| seq1.10.weight |     96     |
|  seq1.10.bias  |     96     |
| seq2.0.weight  |   18432    |
|  seq2.0.bias   |     12     |
| seq2.1.weight  |     12     |
|  seq2.1.bias   |     12     |
| seq2.3.weight  |    4608    |
|  seq2.3.bias   |     24     |
| seq2.4.weight  |     24     |
|  seq2.4.bias   |     24     |
| seq2.6.weight  |   18432    |
|  seq2.6.bias   |     48     |
| seq2.7.weight  |     48     |
|  seq2.7.bias   |     48     |
| seq2.9.weight  |   73728    |
|  seq2.9.bias   |     96     |
| seq2.10.weight |     96     |
|  seq2.10.bias  |     96     |
| seq2.12.weight |   73728    |
|  seq2.12.bias  |     48     |
| seq2.13.weight |     48     |
|  seq2.13.bias  |     48     |
| seq2.15.weight |   18432    |
|  seq2.15.bias  |     24     |
| seq2.16.weight |     24     |
|  seq2.16.bias  |     24     |
| seq2.18.weight |    4608    |
|  seq2.18.bias  |     12     |
| seq2.19.weight |     12     |
|  seq2.19.bias  |     12     |
| seq2.21.weight |    576     |
|  seq2.21.bias  |     3      |
+----------------+------------+
Total Trainable Params: 609915
```

Reason for choosing ResNet50 (Appendix)





Source: AN ANALYSIS OF DEEP NEURAL NETWORK MODELS FOR PRACTICAL APPLICATIONS
Alfredo Canziani & Eugenio Culurciello

Âppendix
Architecture for Unsupervised Approach

Convolutional Operation

$$s[t]=(x\star w)[t]=\sum_{a=-\infty}^{a=\infty} x[a]w[a+t]$$

Feature map Input kernel

Input Feature Map

3	5	2	8	1
9	7	5	4	3
2	0	6	1	6
6	3	7	9	2
1	4	9	5	1

Convolutional Filter

1	0	0
1	1	0
0	0	1

*Figure 4a. **Left**: A 5x5 input feature map (depth 1). **Right**: a 3x3 convolution (depth 1).*

Input Feature Map

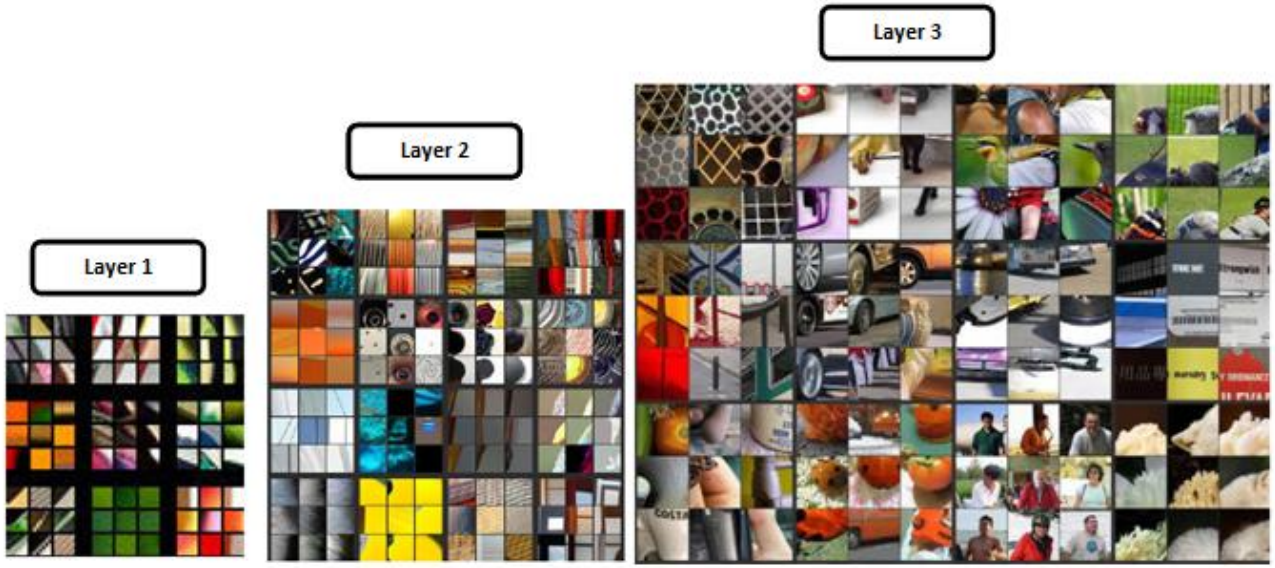
3×1	5×0	2×0	8	1
9×1	7×1	5×0	4	3
2×0	0×0	6×1	1	6
6	3	7	9	2
1	4	9	5	1

3+0+0+9+7+0+0+0+6 →

Output Feature Map

25	18	17
18	22	14
20	15	23

*Figure 4b. **Left**: The 3x3 convolution is performed on the 5x5 input feature map. **Right**: the resulting convolved feature. Click on a value in the output feature map to see how it was calculated.*



Feature Visualization of Convolutional net trained on ImageNet (from Matthew D. Zeiler and Rob Fergus)

Source:Matthew D. Zeiler and Rob Fergus}, {Visualizing and Understanding Convolutional Networks},2013

- Filter extracts local features
- In practice, a number of filters are used to extract different types of local features