# 

CS-405 Deep Learning Fall 2023

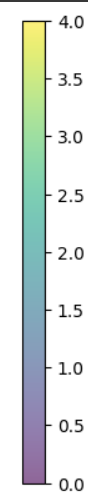
Lab 07

Semantic Image Segmentation

|  |  |
| --- | --- |
| NAME | RAJA HAIDER ALI |
| CMS ID | 346900 |
| GROUP | 2 |

Date: 29th Oct 2023

**Question 1: In the cell below, paths to the images and their corresponding masks are being set. Please analyse the mask of an image and check how does it store the information regarding classes (i.e. different parts) of a vehicle?**

**Answer:** There is a color scale with range between 0-4. 0 is the color purple and 4 is the color yellow and in between these values, there is a mix of color purple and yellow representing the different classes.

* color corresponding to the value 0 is background.
* color corresponding to the value 1 is car body.
* color corresponding to the value 2 is wheel.
* color corresponding to the value 3 is light.
* color corresponding to the value 3 are windows.

The mask covers all the classes by representing them by above mentioned colors from the color range.

**Question 2.1: Please observe the dataset class given in the cell below. When we want to retrieve items using an instance of this class, which function is generally called?**

**Answer:** \_\_getitem\_\_() method is called.

**Question 2.2: What does the data consist of that we obtain when we use an instance of this class for item retreival?**

**Answer:** we obtain a tuple containing two elements: an image and a mask tensor. The image tensor contains the image data, and the mask tensor contains segmentation information. These two tensors are returned as the output of the \_\_getitem\_\_().

**Question 3: In the cell below, from where did we pick up the values of mean and std of RGB values for input normalization?**

**Answer:** These are commonly used pre-computed values from a standard dataset like ImageNet. These values are used as default values, which work well for many computer vision tasks.

**Question 4: Which augmentations are we using in this case?**

**Answer:** Following augmentations are being applied to t\_train.

* **A.Resize(256, 256, interpolation=cv2.INTER\_NEAREST):** Resizes the image to 256x256 pixels with nearest-neighbor interpolation.
* **A.HorizontalFlip():** Apply horizontal flipping.
* **A.VerticalFlip():** Apply vertical flipping.
* **A.GaussNoise():** Add Gaussian noise.

Following augmentations are being applied to t\_val.

* **A.Resize(256, 256, interpolation=cv2.INTER\_NEAREST):** Resized the image to 256x256 pixels with nearest-neighbor interpolation.

**Question 5: Please observe the SegNet model below and answer the following questions:**

**5.1: In SegNet we studied that during MaxPooling, the indices of the maxpooled values are stored during encoding stage and these values are then used in unpooling operations in the decoding stage. How is this being done in the code below? Which functions are being used in this process?**

**Answer:** This process is being done using **MaxPool2d** and **MaxUnpool2d layers.**

For encoding,

self.MaxEn = nn.MaxPool2d(2, stride=2, return\_indices=True)

return\_indices=True ensures that the pooling indices are saved during the max-pooling operation.

For decoding,

self.MaxDe = nn.MaxUnpool2d(2, stride=2)

The MaxUnpool2d layer uses the saved pooling indices (obtained during the encoding stage) to perform the unpooling operation during decoding.

**5.2: How many convolution blocks or stages are being used in the encoder and the decoder?**

**Answer:** 5 blocks are being used in both.

**5.3: End the end of it all, what does the dim=1 in x = F.softmax(x, dim=1) signify?**

**Answer:** In the above code, dim =1 signifies that the softmax operation is applied along the second dimension of the tensor x.

**Question 6: Please observe the UNet model below and answer the following questions:**

**6.1: What does the DoubleConv model do? It is built up using the Sequential method of building up models, what is this method?**

**Answer:** The DoubleConv model is a component used in the UNet model, and it performs a series of operations in a sequential manner. It is a sequence of two 3x3 convolution layers, each followed by batch normalization (BN) and ReLU activation function.

self.double\_conv = nn.Sequential(

            nn.Conv2d(in\_channels, mid\_channels, kernel\_size=3, padding=1, bias=False),

            nn.BatchNorm2d(mid\_channels),

            nn.ReLU(inplace=True),

            nn.Conv2d(mid\_channels, out\_channels, kernel\_size=3, padding=1, bias=False),

            nn.BatchNorm2d(out\_channels),

            nn.ReLU(inplace=True)

        )

The Sequential method is a way of packing layers together in which each layer is applied one after the other in a sequential manner. As in the above code,

nn.Conv2d(in\_channels, mid\_channels, kernel\_size=3, padding=1, bias=False)

will be applied first , then its output will be given as input to the next layer which is

nn.BatchNorm2d(mid\_channels)

and so on upto the last , which is

nn.ReLU(inplace=True)

**6.2: What operations does the Down module constitute? What is its basic purpose?**

**Answer:** Its purpose is to downscale the input features by applying

 nn.MaxPool2d(2)

which reduces the spatial dimensions of the input by a factor of 2 and then its output is passed through double convolution layers to increase the number of channels/depth.

DoubleConv(in\_channels, out\_channels)

So, the basic purpose of this module is to reduce the spatial dimension and increase the depth of its input.

**6.3: What is the purpose of the Up module? What operations does it consists of?**

**Answer:**  Its purpose is to upscale the feature maps and then combine it with the feature maps from a previous layer to form a skip connection.

self.up = nn.ConvTranspose2d(in\_channels, in\_channels // 2, kernel\_size=2, stride=2)

This layer increases the spatial dimension of the input by a factor of 2. The upscaled feature maps are then concatenated with feature maps from a previous layer, padding is applied and finally, the concatenated feature maps are passed through DoubleConv(in\_channels,out\_channels) which consist of 2 convolution layers with Batch Norm and ReLU activation function.

**Question 7: The segmentation\_models\_pytorch is a useful library for using pretrained models for the task of semantic segmentation. If we are using the Unet model, then in this case we can choose between different pre-trained models as the encoder for the UNet. In this cell below we have used "vgg16\_bn".Please visit the link given below and mention 3 other encoders listed in the repository.**

|  |  |  |
| --- | --- | --- |
| **Encoder** | **Weights** | **Params, M** |
| vgg11\_bn | imagenet | 9M |
| mit\_b1 | imagenet | 13M |
| resnet34 | imagenet | 21M |

**Question 8: In the 2 cells given below, we are computing Pixel Accuracy as well as mean Intersection over Union evaluation metrics. Whats the difference between the two? Which one is better and why?**

**Answer:**

|  |  |  |
| --- | --- | --- |
| Metric | Pixel Accuracy | Intersection over Union |
| Description | Measures overall pixel-wise classification accuracy | Measures the overlap between predicted and ground truth segmentation |
| Formula | Correctly Classified Pixels / Total Pixels | Intersection Area / Union Area |
| Use Cases | Global view of model performance; treats all classes equally; simple metric | Class-specific evaluation; mean IoU provides an average IoU score for all classes; informative metric |

It depends on the specific use and nature of the dataset to choose which one performs better. If the dataset has a balanced class distribution and we want a simple and global evaluation, then pixel accuracy is a good choice. On the other hand, if dataset is imbalanced and we want the model's performance on a as-per class basis then IoU is a better choice.

**Question 9: If you had to use a pretrained encoder other than the VGG16\_BN for the UNet model, which one would you prefer to use, and why? (Note: There is no single correct answer to this question. Follow your intuition and current knowledge base that we have developed in the classes.)**

**Answer:** If I had to use a pretrained encoder other than VGG16\_BN for the UNet model, I would prefer to use ResNet-50. ResNet-50 is a strong choice because it has demonstrated excellent performance in various computer vision tasks, and its skip connections can enhance feature extraction, which aligns well with UNet's architecture for image segmentation.

**Question 10: Why are we creating the CarsTestDataset class? Why not use the same dataset class that we had defined earlier? What is the difference between the two?**

**Answer:** For train data, we created CarDataset class. Now, we are creating a different class CarsTestDataset and the reason is we want different data processing and transformation methods for test data to ensure proper evaluation of the model’s performance to the new unseen data. We want to see whether our model generalizes to the new data or not.

The differences between the two is that CarsTestDataset applies a different data transformation compared to the CarsDataset class and the data composition is also different.

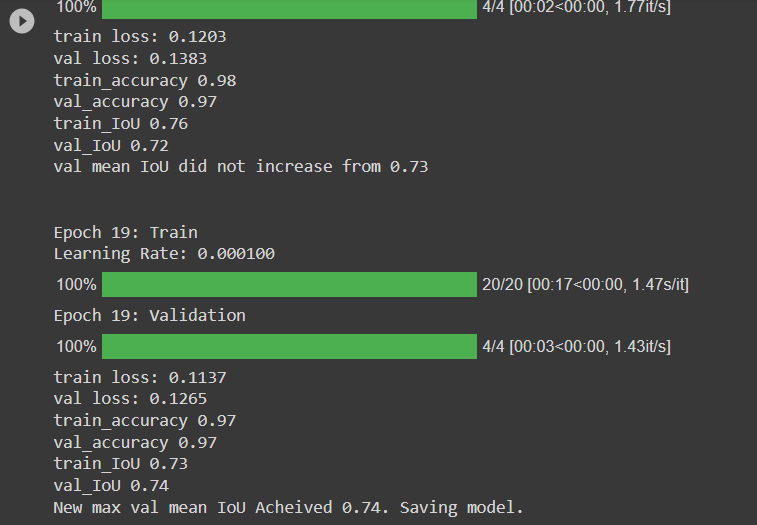
**Tasks:**

**Please use the training script below and fine-tune/train the pre-trained UNet-VGG16 encoder model on the cars dataset. Try to experiment with the hyperparameters to achieve maximum validation IoU.**

**Answer:**

|  |  |
| --- | --- |
| **Hyperparameter** | **Value** |
| Learning rate | 0.0001 |
| Step\_size | 60 |
| Epochs | 20 |

Using the above hyperparameter values, I achieved a maximum validation IoU value of 74.



**Discussion on the training plots obtained at the end of the training, and the validation and test results achieved.**

**Answer:**

* The training has progressed well as the training loss decreased and the training accuracy increased after each epoch. The training IoU also improved with each epoch.
* The validation loss decreased over the epochs which indicates that the model generalized well to the unseen data. This is also evident form the relatively high validation accuracy.
* The validation IoU is also quite high, with the best value achieving 0.74. This means that the model is effectively segmenting objects in the validation data.
* The validation mean IoU starts at 0.33 but gradually improves over epochs. The best mean IoU obtained is 0.74, indicating that the model is becoming good at object segmentation.
* There is no separate test set used. To achieve a more accurate assessment of the model, a separate testing set can be used.