SEMANTIC SEGMENTATION

P Bhaskar Tarun
IIT Ropar
Nangal Road, Ropar, Punjab
2019eem1013@iitrpr.ac.in

Jatoth Jayakrishna
IIT Ropar
Nangal Road, Ropar, Punjab
2019eem1008@iitrpr.ac.in

Abstract

Semantic segmentation, also called scene labeling, refers to the process of assigning a semantic label (e.g. car, people, and road) to each pixel of an image. It is an essential data processing step for robots and other unmanned systems to understand the surrounding scene. Despite decades of efforts, semantic segmentation is still a very challenging task due to large variations in natural scenes. In this paper, Firstly, we briefly summarize the traditional methods as well as datasets related to the segmentation, and then we comprehensively investigate recent methods based on Deep neural networks (DNN). Finally we will propose our updated model for better accuracy.

1. Introduction

For the last three decades, one of the most difficult problems in computer vision has been semantic segmentation. Semantic segmentation is difficult form image classification or object recognition in that it is not necessary to know what the visual concepts or objects are beforehand. To be specific, an object classification will only classify objects that it has specific labels for such car, building, tree, motor cycle. An ideal semantic segmentation algorithm will also segment unknown objects which are new or unknown. There are numerous applications where semantic segmentation could be used to improve existing algorithms from cultural heritage preservation to image copy detection to satellite imagery analysis to on the fly visual search and human access to segmentations would allow the problem to be approached at a semantic level. For example, in content based image retrieval, each image could be segmented as it is added to the database. When a query is processed, it could be segmented as it is added to the database. When a query is processed, it could be segmented and allow the user to query for similar segments in the database. In human computer interaction, every part of each video frame would be segmented so that the user could interact at a finer level with other humans and objects in the environment. In the context of an airport, for example, the security team is typically interested in any unattended baggage, some of which could hold dangerous materials. It would be beneficial to make queries for all objects which were left behind by a human.

Given a new image, an image segmentation algorithm should output which pixels of the image belong together semantically. For example, in Fig 1 a) the input image consists of a dog sleeping on the chair. In Fig 1 b) we see the ideal segmentation which clusters the pixels by the semantic objects, all of the pixels belonging to the dog are colored yellow to show that they belong together, similarly the chair are colored blue and the background objects all are colored red.

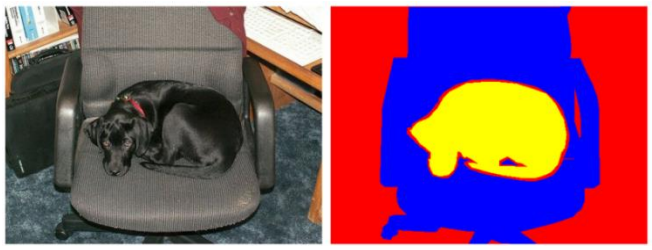


Fig 1.a) Dog on the chair image b) Segmentation for dog on chair

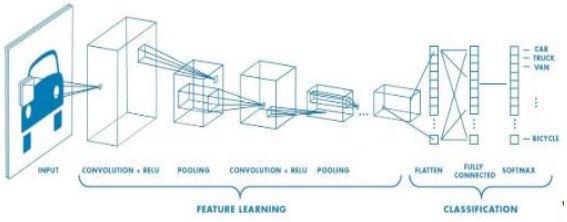
2. Convolution Neural Networks

Like most of the other applications, using a CNN for semantic segmentation is the obvious choice. When using a CNN for semantic segmentation, the output is also an image rather than a fixed length vector. Usually, the architecture of the model contains several convolutional layers, non-linear activations, batch normalization, and pooling layers. The initial layers learn the low-level concepts such as edges and colours and the later level layers learn the higher level concepts such as different objects.

At a lower level, the neurons contain information for a small region of the image, whereas at a higher level the neurons contain information for a large region of the image. Thus, as we add more layers, the size of the image keeps on decreasing and the number of channels keeps on increasing. The down sampling is done by the pooling layers.

For the case of image classification, we need to map the spatial tensor from the convolution layers to a fixed length

vector. To do that, fully connected layers are used, which destroy all the spatial information.



. Fig 2. CNN

.

2.1. Layers of CNN

2.1.1 Convolution

Convolution is one of the main building blocks of a CNN. The term convolution refers to the mathematical combination of two functions to produce a third function. It merges two sets of information. In the case of a CNN, the convolution is performed on the input data with the use of a **filter** or **kernel** (these terms are used interchangeably) to then produce a **feature map.** We execute a convolution by sliding the filter over the input. At every location, a matrix multiplication is performed and sums the result onto the feature map.

2.1.2 Activation Function

It is used to determine the output of a neural network (like yes/no).It can also be attached between two different neural networks. The popular activation functions are RELU, softmax etc.

2.1.3 Batch Normalization

Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

2.1.4 Dropout

Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel. During training some number of layer outputs are randomly ignored or "dropped out". This has the effect of marking the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with different "view" of the configured layer.

2.1.5 Pooling

It is responsible for reducing the spatial size of the convoluted feature. Pooling combines the output of a neuron cluster at one layer into a single neuron in the next layer. Max pooling uses the maximum value from each cluster of neurons at the prior layer. Average pooling uses the average value from clusters of neurons at the prior layer.

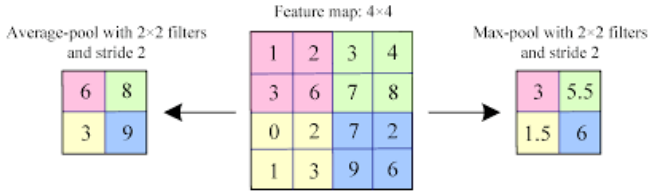


Fig 3. Pooling

2.1.6 Transposed Convolution

Transposed Convolution is the most preferred choice to perform up sampling, which basically learns parameters through back propagation to convert a low resolution image to a high resolution image. However, on a high level, transposed convolution is exactly the opposite process of a normal convolution i.e., the input volume is a low resolution image and the output volume is a high resolution image.

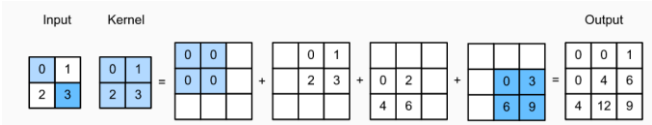


Fig 4. Transposed Convolution

2.2 CNN Architectures for Semantic segmentation

Basically, segmentation is a process that partitions an image into regions. It is an image processing approach that allows us to separate objects and textures in images**.** Segmentation is especially preferred in applications such as remote sensing or tumor detection in biomedicine**.**

There are many traditional ways of doing this. For example: point, line, and edge detection methods, thresholding, region-based, pixel-based clustering, morphological approaches, etc. Various methods have been developed for segmentation with convolutional neural networks (a common deep learning architecture), which have become indispensable in tackling more advanced challenges with image segmentation. In this post, we'll take a closer look at one such architecture: **u-net**.

2.2.1 U-net

U-Net is more successful than conventional models, in terms of architecture and in terms pixel-based image segmentation formed from convolutional neural network layers. It's even effective with limited dataset images. The presentation of this architecture was first realized through the analysis of biomedical images.

These layers are intended to increase the resolution of the output. For localization, the sampled output is combined with high-resolution features throughout the model. A sequential convolution layer then aims to produce a more precise output based on this information.

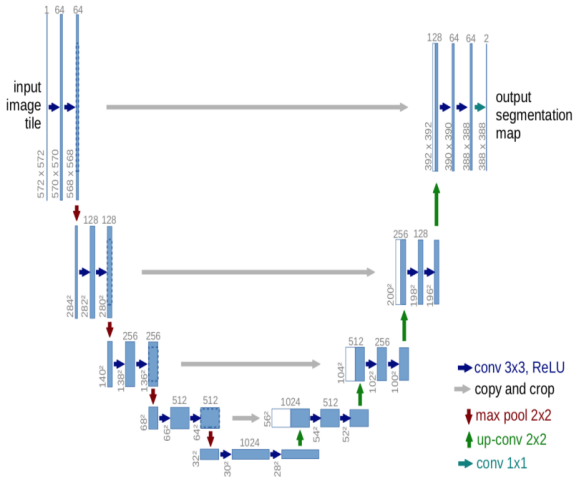


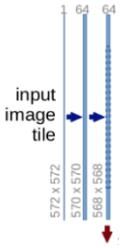
Fig 5. U-net

U-Net takes its name from the architecture, which when visualized, appears similar to the letter *U*, as shown in the Fig 5 above. Input images are obtained as a segmented output map. The most special aspect of the architecture in the second half. The network does not have a fully-connected layer. Only the convolution layers are used. Each standard convolution process is activated by a ReLU activation function. U-Net is unique; it might be helpful to quickly compare it to a different traditional approach to image segmentation: the autoencoder architecture. In classical autoencoder architecture, the size of the input information is initially reduced, along with the following layers. At this point, the encoder part of the architecture is completed and the decoder part begins. Linear feature representation is learned in this section, and the size gradually increases. At the end of the architecture, the output size is equal to the input size.
Now let's break down the implementation line by line and maps to the corresponding parts on the image of UNet architecture for better explanation.

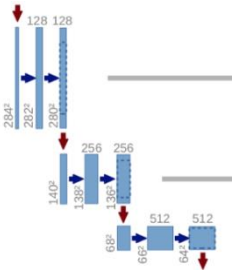
- **Contracting Path**

The contracting path follows the formula:
conv_layer1 > conv_layer2 > max_pooling > dropout(optional)

all convolution process are having same filter size as 3x3.



Notice that each process constitutes two convolutional layers**,** and the number of channel changes from 1 → 64, as convolution process (filter size 3x3) will increase the depth of the image. The red arrow pointing down is the max pooling process which halves down size of image(the size reduced from 572x572 → 568x568 is due to padding issues, but the implementation here uses padding= "same").The process is repeated 3 more times:

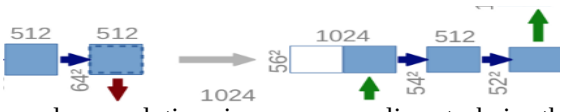


Now we reaches at the bottommost:

Still 2 convolutional layers are built, but with no max pooling**:** The image at this moment has been resized to 28x28x1024. Now let's get to the expansive path.

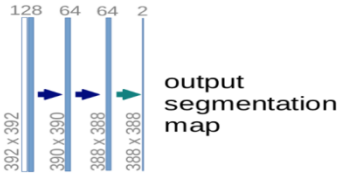
- **Expansive Path**

In the expansive path, the image is going to be upsized to its original size. The formula follows:
conv_2d_transpose > concatenate > conv_layer1 > conv_layer2



Transposed convolution is an upsampling technic that expands the size of images. Basically, it does some padding on the original image followed by a convolution operation.
After the transposed convolution, the image is upsized from 28x28x1024 → 56x56x512, and then, this image is concatenated with the corresponding image from the contracting path and together makes an image of size 56x56x1024.

The reason here is to combine the information from the previous layers in order to get a more precise prediction.
Same as before, this process is repeated 3 more times:
Now we've reached the uppermost of the architecture, the last step is to reshape the image to satisfy our prediction requirements.



The last layer is a convolution layer with 1 filter of size 1x1(notice that there is no dense layer in the whole network). And the rest left is the same for neural network training.

The U-net is convolution network architecture for fast and precise segmentation of images. Up to now it has outperformed the prior best method (a sliding-window convolution network) on the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks. It has won the Grand Challenge for Computer-Automated Detection of Caries in Bitewing Radiography at ISBI 2015 on the two most challenging transmitted light microscopy categories by a large margin.

3. Identified Problem and Data Source

The dataset is taken from Kaggle website:

https://www.kaggle.com/dansbecker/cityscapes-image-pairs

Cityscapes is a dataset consisting of diverse urban street scenes across 50 different cities at varying times of the year as well as ground truths for several vision tasks including semantic segmentation, instance level segmentation (TODO), and stereo pair disparity inference. For segmentation tasks Cityscapes provides dense pixel level annotations for 3475 images at pre-split into training (2975) and validation (500) sets. Label annotations for segmentation tasks span across 30+ classes commonly encountered during driving scene perception. Each image file is 256x512 pixels, and each file is a composite with the original photo on the left half of the image, alongside the labeled image (output of semantic segmentation) on the right half.

Cityscapes data (dataset home page) contains labeled videos taken from vehicles driven in Germany. This version is a processed subsample created as part of the Pix2Pix paper. The dataset has still images from the original videos, and the semantic segmentation labels are shown in images alongside the original image. This is one of the best datasets around for semantic segmentation tasks.

There are many segmentation labels in Cityscapes dataset images; it is very difficult to differentiate between them by segmentation mask as they are having lots of things in common. One example is that of such labels is difference Road and Footpath.
There is a problem in masking the object completely without any interruption of other color of mask (Obtain a plain mask of the object).
In dark areas in image it is very difficult to find an object even by humans. We will make such a CNN model that can overcome all this problems.

4. Existing Solutions

There are few solutions for this problem but there are some drawbacks with those solutions.

4.1 Fully convolutional networks

The approach of using a "fully convolutional" network trained end-to-end, pixels-to-pixels for the task of image segmentation. The paper's authors propose adapting existing, well-studied *image classification* networks (eg. AlexNet) to serve as the encoder module of the network, appending a decoder module with transpose convolutional layers to upsample the coarse feature maps into a full-resolution segmentation map.

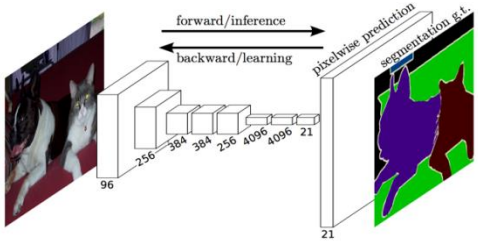


Fig 6. Fully convolutional networks

However, because the encoder module reduces the resolution of the input by a factor of 32, the decoder module **struggles to produce fine-grained segmentations.**

4.2 Skip connections

The authors address this tension by slowly upsampling (in stages) the encoded representation, adding "skip connections" from earlier layers, and summing these two feature maps.

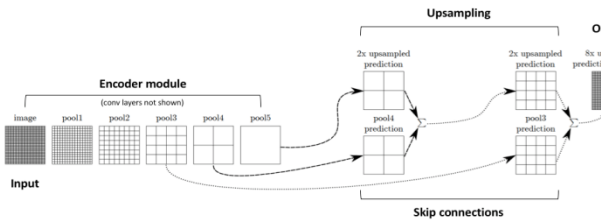


Fig 7. Skip Connection

These skip connections from earlier layers in the network (prior to a downsampling operation) should provide the necessary detail in order to reconstruct accurate shapes for segmentation boundaries. Indeed, we can recover more fine-grain detail with the addition of these skip connections. Improve the "fully convolutional" architecture primarily through ***expanding the capacity of the decoder*** module of the network.

More concretely, they propose the **U-Net architecture** which "consists of a contracting path to capture context and a ***symmetric*** expanding path that enables precise localization." This simpler architecture has grown to be very popular and has been adapted for a variety of segmentation problems.

5. Exploratory Data Analysis

On analyzing the data we found that the data is split into two categories. Those are training and validation set. The training set consists of 2975 and validation set consists of 500 images. The image is of size 256x516 pixels, and each file is a composite with the original photo on the left half of the image, alongside the labeled image (output of semantic segmentation) on the right half as shown in Fig 8.

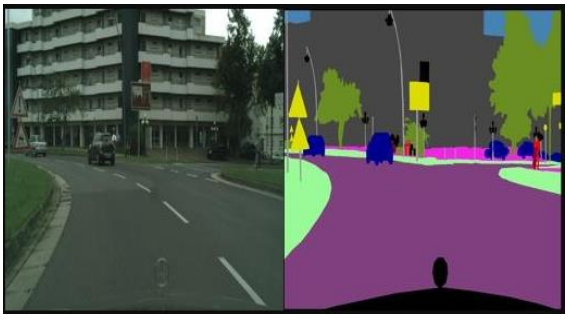


Fig 8. Dataset image

The dataset consists of around 2975 images with almost 30 categories in which some of the main object categories which i observed as

Category	Label color
Buildings	Dark gray
Cars	Dark blue
Humans	Red
Trees	Dark green
Footpath	Pink
Road	Purple
Traffic signs	Yellow
Street light poles	Gray
Grass	Light green
Traffic lights	Orange
Motor cycle	Brown
Unknown objects	Black

Form the object categories in the image and from my analysis it's very easy to detect big size objects in image such as Buildings, Cars, Trees, Road, Footpath, etc. But it's very difficult to detect small size objects in image such as Traffic signs, Traffic lights, etc.

On analyzing the dataset we got to know that the categories are more so we need to design a big model, so that it can analyze each and every object in the image.

6. Proposed Machine Learning Models

In order to achieve highest accuracy and best segmented image, we tried different combination of convolution, pooling, regularization (Dropout) and different types of random weight initializers for training the Cityscapes dataset.

```
Model:"model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected
to
=================================================================
input_3(InputLayer)[(None,256,256,3)      0         _____________
_________________________________________________________________
lambda_2(Lambda)          (None,256,256,3)     0       input_3[0][0]
_________________________________________________________________
conv2d_23(Conv2D)         (None,256,256,16)    448     lambda_2[0][0]
_________________________________________________________________
dropout_11(Dropout)       (None,256,256,16)    0       conv2d_23[0][0]
_________________________________________________________________
conv2d_24(Conv2D)         (None,256,256,16)    2320    dropout_11[0][0]
_________________________________________________________________
max_pooling2d_5           (None,128,128,16)    0       conv2d_24[0][0]
(MaxPooling2D)
_________________________________________________________________
conv2d_25                 (None,128,128,16)    2320      max_pooling2d
(Conv2D)                                                   _5[0][0]
_________________________________________________________________
dropout_12(Dropout)       (None,128,128,16)    0       conv2d_25[0][0]
_________________________________________________________________
conv2d_26(Conv2D)         (None,128,128,16)    2320    dropout_12[0][0]
_________________________________________________________________
max_pooling2d_6           (None,64,64,16)      0       conv2d_26[0][0]
(MaxPooling2D)
_________________________________________________________________
conv2d_27                 (None,64,64,32)      4640    max_pooling2d
(Conv2D)                                                   _6[0][0]
_________________________________________________________________
dropout_13(Dropout)       (None,64,64,32)      0       conv2d_27[0][0]
_________________________________________________________________
conv2d_28(Conv2D)         (None,64,64,32)      9248    dropout_13[0][0]
_________________________________________________________________
max_pooling2d_7           (None,32,32,32)      0       conv2d_28[0][0]
(MaxPooling2D)
_________________________________________________________________
conv2d_29                 (None,32,32,64)      18496     max_pooling2d
(Conv2D)                                                   _7[0][0]
```

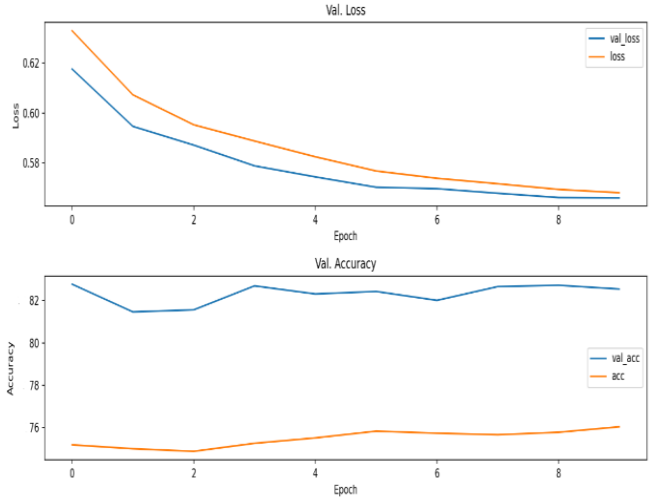
```
dropout_14(Dropout)     (None,32,32,64)     0     conv2d_29[0][0]
conv2d_30(Conv2D)       (None,32,32,64)     36928 dropout_14[0][0]
max_pooling2d_8         (None,16,16,64)     0     conv2d_30[0][0]
 (MaxPooling2D)
conv2d_31               (None,16,16,128)    73856 max_pooling2d
(Conv2D)                                          _8[0][0]
dropout_15(Dropout)     (None,16,16,128)    0     conv2d_31[0][0]
conv2d_32(Conv2D)       (None,16,16,128)    147584 dropout_15[0][0]
max_pooling2d_9         (None,8,8,128)      0     conv2d_32[0][0]
(MaxPooling2D)
conv2d_33               (None,8,8,256)      295168 max_pooling2d
(Conv2D)                                          _9[0][0]
dropout_16(Dropout)     (None,8,8,256)      0     conv2d_33[0][0]
conv2d_34(Conv2D)       (None,8,8,256)      590080 dropout_16[0][0]
conv2d_transpose_5      (None,16,16,128)    131200 conv2d_34[0][0]
(Conv2DTrans)
concatenate_5           (None,16,16,256)    0     conv2d_transpose_5[0][0]
(Concatenate)                                     conv2d_32[0][0]
conv2d_35               (None,16,16,128)    295040 concatenate_5[0][0]
(Conv2D)
dropout_17(Dropout)     (None,16,16,128)    0     conv2d_35[0][0]
conv2d_36(Conv2D)       (None,16,16,128)    147584 dropout_17[0][0]
conv2d_transpose_6      (None,32,32,64)     32832 conv2d_36[0][0]
(Conv2DTrans)
concatenate_6           (None,32,32,128)    0     conv2d_transpose_6[0][0]
(Concatenate)                                     conv2d_30[0][0]
conv2d_37               (None,32,32,64)     73792 concatenate_6[0][0]
(Conv2D)
dropout_18(Dropout)     (None,32,32,64)     0     conv2d_37[0][0]
conv2d_38(Conv2D)       (None,32,32,64)     36928 dropout_18[0][0]
conv2d_transpose_7      (None,64,64,32)     8224  conv2d_38[0][0]
(Conv2DTrans)
concatenate_7           (None,64,64,64)     0     conv2d_transpose_7[0][0]
(Concatenate)                                     conv2d_28[0][0]
conv2d_39(Conv2D)       (None,64,64,32)     18464 concatenate_7[0][0]
dropout_19(Dropout)     (None,64,64,32)     0     conv2d_39[0][0]
conv2d_40(Conv2D)       (None,64,64,32)     9248  dropout_19[0][0]
conv2d_transpose_8      (None,128,128,16)   2064  conv2d_40[0][0]
(Conv2DTrans)
concatenate_8           (None,128,128,32)   0     conv2d_transpose_8[0][0]
(Concatenate)                                     conv2d_26[0][0]
conv2d_41               (None,128,128,16)   4624  concatenate_8[0][0]
(Conv2D)
dropout_20(Dropout)     (None,128,128,16)   0     conv2d_41[0][0]
conv2d_42(Conv2D)       (None,128,128,16)   2320  dropout_20[0][0]
conv2d_transpose_9      (None,256,256,16)   1040  conv2d_42[0][0]
(Conv2DTrans
conv2d_43(Conv2D)       (None,256,256,16)   4624  concatenate_9[0][0]
ropout_21(Dropout)      (None,256,256,16)   0     conv2d_43[0][0]
conv2d_44(Conv2D)       (None,256,256,16)   2320  dropout_21[0][0]
conv2d_45(Conv2D)       (None,256,256,3)    51    conv2d_44[0][0]
==================================================================
Total params: 1,953,763
Trainable params: 1,953,763
Non-trainable params: 0
```

After testing several models I finally finalized this model as the best model for segmentation.
The model initially downsampled by Convolution (3x3) > Dropout (10%) > Convolution (3x3) > maxpooling.
This process is repeated 4 times, later it is again 2 times Convoluted (3x3) and later upsampled by several Transpose convolution (3x3) and obtained the same size as that of the input.
We also used callback function '**EarlyStopping**' to achieve optimum accuracy and used batch normalization to increase the training speed. The kernel initializer which gives better performance in segmentation model is '**he_normal**'. The optimizer we used in thid model is '**Adam**' and loss calculations by '**binary_crossentropy**' and matrics by '**accuracy**'. The model has been trained for 10 epochs.

7. Performance Evaluation

The Segmentation Model has achieved a training accuracy of 76 percent and a validation accuracy of 82.5 percent at epoch number 10.

The plots shown below, y-axis represents either accuracy or loss, and the x-axis represents the number of epochs for which the model has been trained.



The segmentation images obtained this model gives better performance and classify the objects accurately. The Fig 9 shows the actual image, actual image mask and our predicted image mask.

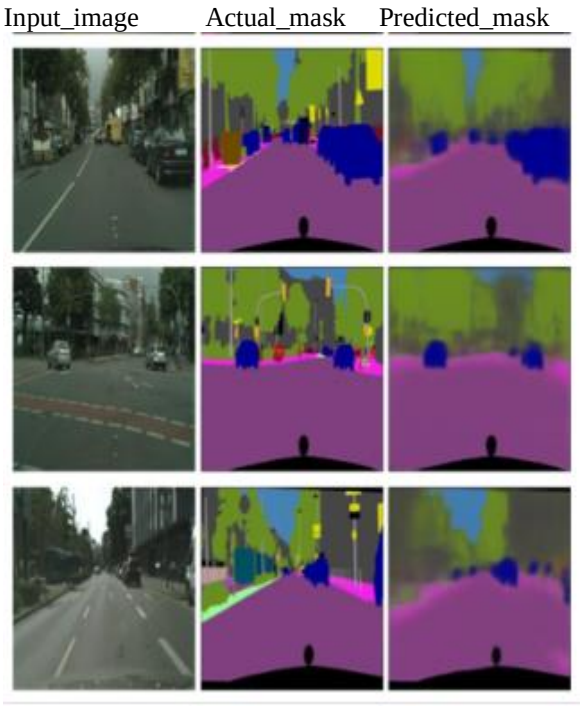


Fig 9. Model output(segmented image)

As shown is the above Fig 9 our model can label the objects very accurately such as cars, buildings, roads etc.

8. Conclusion

In the work of this paper, in order to meet the needs of higher precision in image segmentation tasks, we proposed a Deep neural network model. The network structure proposed in this paper is designed based on a U-net architecture, and it's all part contains Dropout of 10% and 20% and there are several other layers have been included. This design guarantees the integrity of semantic information to the greatest extent possible. The architecture makes the results of semantic segmentation of Cityscapes images clearer and more accurate through a redesigned neural network structure. We haven't used any pre-training model to transfer learning and ImageDataGeneratorin our proposed model to cope with the problem; this is because our training data set very large, so there is no requirement for pre-trained model and ImageDataGenerator. We put the convolution structure at the starting and ending of the network model to apply the deeper network to our model, and use the long connections in the network structure to effectively couple the semantic information of different levels, effectively improving the performance of the model in the semantic segmentation of satellite images. In the future, we will continue to focus on the application of combining different levels of semantic information in multiple image processing directions. Upon analyzing the training and validation accuracies, and predicted segmented image compared with the actual segmented image, we can clearly say that Model is a better model for Cityscapes dataset. Hence, proposed Model is the model that is best suited to our Cityscapes dataset.

References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox "U-Net: Convolutional Networks for Biomedical Image Segmentation" arXiv:1505.04597v1 [cs.CV] 18 May 2015.

[2] NARINDER SINGH PUNN and SONALI AGARWAL, "Inception U-Net Architecture for Semantic Segmentation to Identify Nuclei in Microscopy Cell Images," ACM Trans. Multimedia Comput. Commun. Appl., Vol. 16, No. 1, Article 12. Publication date: February 2020.

[3] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A.L., 2014. "Semantic image segmentation with deep convolutional nets and fully connected"crfs. arXiv preprint arXiv:1412.7062.

[4] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C. and Torr, P.H., 2015. Conditional random fields as recurrent neural networks. In Proceedings of the IEEE International Conference on Computer Vision..

[5] Hu, Ronghang, Marcus Rohrbach, and Trevor Darrell. "Segmentation from Natural Language Expressions." arXiv preprint arXiv:1603.06180 (2016).

[6] Arbelaez, Pablo, et al. "Contour detection and hierarchical image segmentation." IEEE transactions on pattern analysis and machine intelligence, 2011.

[7] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, Alan Yuille. CVPR, 2014.

[8] Shotton, Jamie, et al. "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context." International Journal of Computer Vision, 2009.

[9] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep

convolutional neural networks." Advances in neural information processing systems. 2012.

[10] Mostajabi, Mohammadreza, Payman Yadollahpour, and Gregory Shakhnarovich. "Feedforward semantic segmentation with zoom-out features." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.