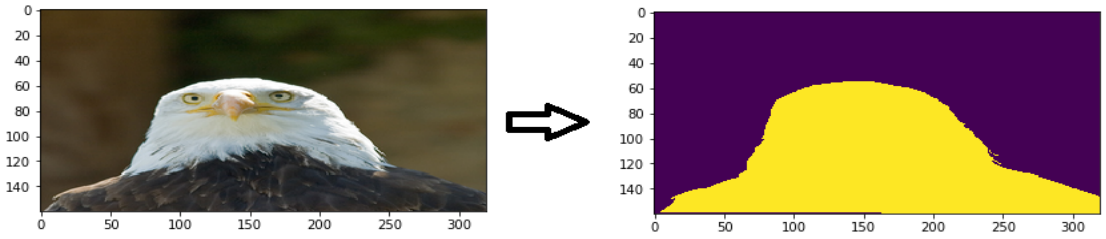
Capstone Project PRATIK KUMAR

Machine Learning Engineer Nanodegree

Definition

Project Overview:

Images are one of the most widely used forms of communication in today's world. Everyday millions of images are created. Since quite a time there has been active research going on in the field of image understanding and object detection. In the project I have tried to achieve an algorithm which can find active boundaries of various objects in the image and label them. Formally the task is known as 'image segmentation' or 'semantic segmentation'. Basically the purpose of this project is to device a classifier that can look at the image & tell which classes are present in the images & their pixel –wise locations. In layman terms it can be explained as : In an image containing bird if we replace all pixels of bird with one value & all other pixels with another value, it becomes immensely easy to find bird in that modified image. The new image is then called a segmented image.



I here in the project, am working on VOC Pascal Dataset [2]. It contains 20 classes like bird, bicycle, person, etc. for identification purpose.

'Bounding box detection' is a similar task in which we recognize the object in image & surround it with a box, as implemented in paper[8].

In this paper they have designed a deep neural network architecture to implement object localization. They present a simple and yet powerful formulation of object detection as a regression problem to object bounding box masks. They also have evaluated their model on VOC Pascal dataset.

Problem Statement:

Image segmentation is the operation of partitioning an image into a collection of connected sets of pixels. The purpose of the image segmentation is to partition an image into meaningful regions which is easier to understand. Normally image segmentation is an initial & vital step in understanding the images. The aim of the project is to understand how image segmentation can be achieved using neural network. In the project, I have tried to create a Convolutional Neural Network model that can determine semantic boundaries. My aim is to recognize 20 classes - bicycle, bird, boat, chair, cow, dog, human, etc.

Metrics:

In this project I am using ***categorical accuracy*** and ***mean IU*** as the evaluation metrics. I am using keras to implement the model.

For categorical accuracy I am using pixel-wise accuracy i.e. number of number of pixels correctly predicted out of total number of pixels.

Keras provides an inbuilt metrics '*categorical_accuracy*' to calculate the accuracy.

For computing '*categorical_accuracy*', output should be in the form of one hot encoded vector. In the project I have 20 classes & 1 background. So output will be 21 sized vector.

Mean IU is the ratio of true positive and (true positive + false positive + false negative).

In the project I have defined my own implementation of computeIU function to calculate the mean IU accuracy.

The justification for using these metrics is that in my model I am doing pixel-wise prediction, so these metrics are the good measure of the model's pixel-wise performance.

Analysis

Data Exploration:

Dataset used for the project is VOC Pascal dataset from the Visual Object Classes Challenge 2012 (VOC2012) [2]. VOC Pascal dataset provides images for various image recognition competitions for tasks like classification, detection, segmentation, action classification. In my project I am only doing the task of class segmentation.

The dataset consists of 2913 training images of various sizes and their corresponding segmented images. Separate test images are provided but since ground truth is not available, so I didn't use them to evaluate my model performance. The idea which I have adopted in my project is splitting the training data of 2913 images into train/validation/test data, so that I can get a good measure of accuracy of the model. I am training the model on 2500 images, performing validation on 113 images & testing the model on rest 300 images.

As seen in the above image pair, we can see both the person and the motorbike have been segmented as 2 different classes.

Furthermore, any class can occur any number of times in an image, all the occurrences of multiple objects of the same class have been given the same label. This can be viewed with the help of the below image pair.

Here we can see that there are two persons & two aeroplanes in the original image. While all the objects are segmented, both the aeroplanes have got same colored label and both the persons have got same colored label different from the aeroplanes.

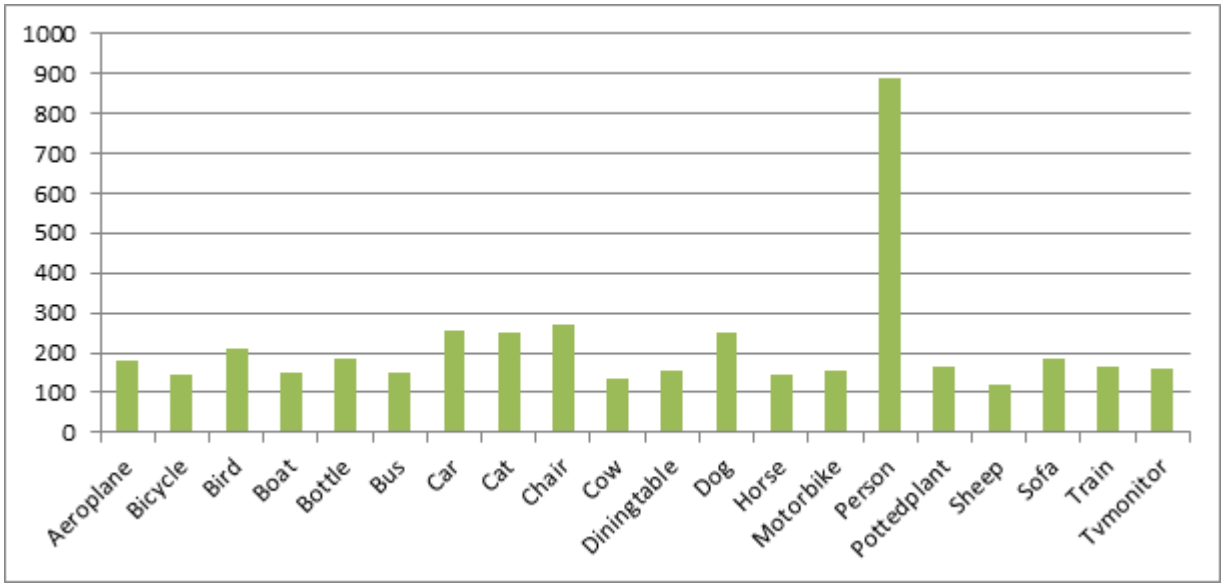
The segmented images consist of 20 classes & 1 background. So I have 21 classes in total. In segmented image each pixel indices correspond to classes in alphabetical order (1=aeroplane, 2=bicycle, 3=bird, 4=boat, 5=bottle, 6=bus, 7=car, 8=cat, 9=chair, 10=cow, 11=dining table, 12=dog, 13=horse, 14=motorbike, 15=person, 16=potted plant, 17=sheep, 18=sofa, 19=train, 20=tv/monitor).
Dataset can be directly downloaded from the PASCAL VOC website[2].

Following are few images from the dataset.

Exploratory visualization:

The dataset consists of 20 classes and 1 background which contains 2913 images in total.

The below visualization shows the distribution of the **classes** in the dataset. For example ‘person’ class has occurred in 887 images. Similarly ‘cow’ has occurred in 135 images.

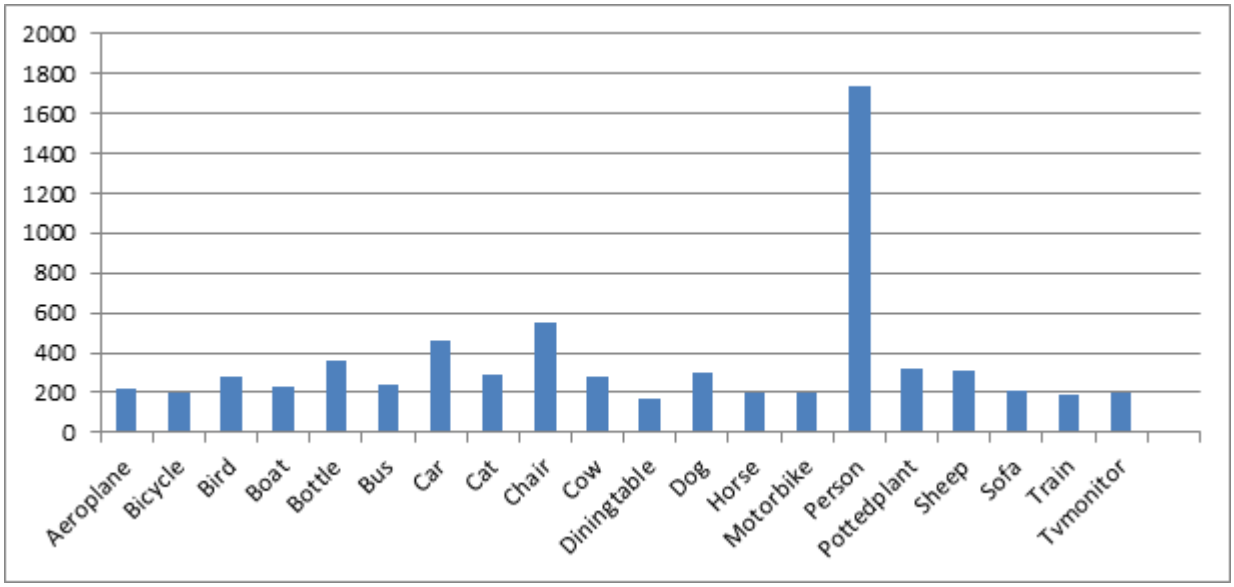


There are 2 important things to note about the dataset:

1. An image can contain one or more classes.
2. Furthermore, any class can occur any number of times in an image.

In my project I am determining classes not objects/instances. So the model will label all the occurrence of a class object with the same label.

The below visualization summarizes the distribution of the **instances** in the dataset. For example ‘person’ class has occurred in 1733 times in dataset. Similarly ‘cow’ has occurred 284 times in dataset.



Algorithm & Techniques:

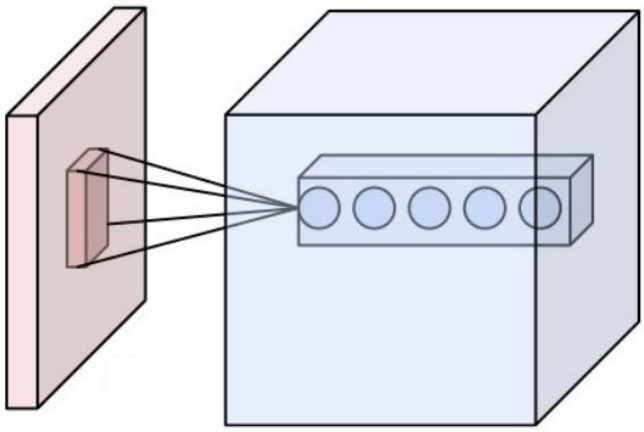
Convolutional neural networks are used in the project to achieve the task of image segmentation. In the last 10 years, neural networks have evolved exponentially & have been proven as state of the art algorithm for most of the image processing task.

The Convolutional Neural Networks (CNNs) have several different filters/kernels consisting of trainable parameters depending on the depth and filters at each layer of a network, which can convolve on a given input volume (the first input being the image itself) spatially to create some feature/activation maps at each layer. During this process, (through back-propagation) they learn by adjusting those initial values to capture the correct magnitude of a spatial feature on which they are convolving. These high numbers of filter essentially learn to capture spatial features from the input volumes based on the learned magnitude. Hence they can successfully boil down a given image into a highly abstracted representation which is easy for predicting.

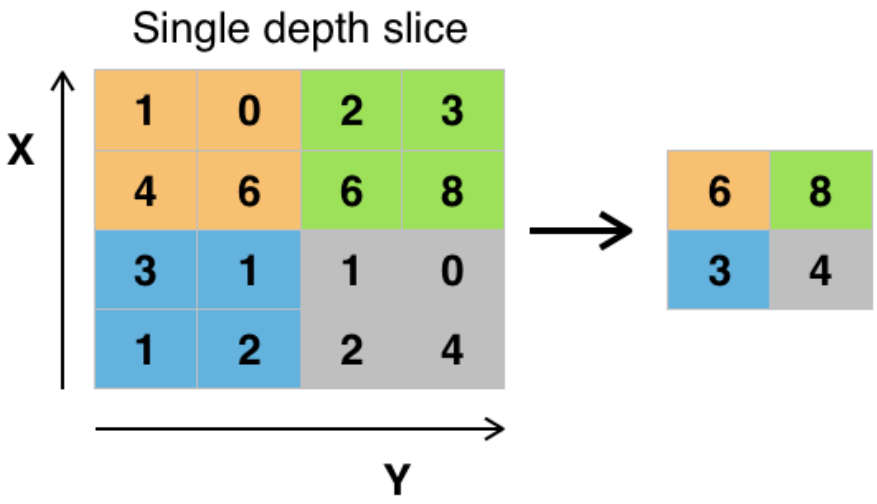
Two of the most important layers of CNN are Convolutional and Pooling layer.

Convolutional layer- The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the

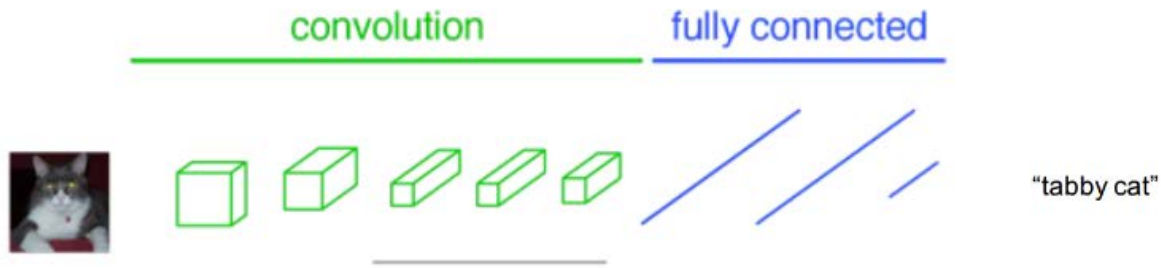
input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.



Pooling layer- Sometimes when the images are too large, we would need to reduce the number of trainable parameters. It is then desired to periodically introduce pooling layers between subsequent convolution layers. Pooling is done for the sole purpose of reducing the spatial size of the image. Pooling is done independently on each depth dimension, therefore the depth of the image remains unchanged. The most common form of pooling layer generally applied is the max pooling. Max pooling 2D is the application of a moving window across a 2D input space, where the maximum value within that window is the output.

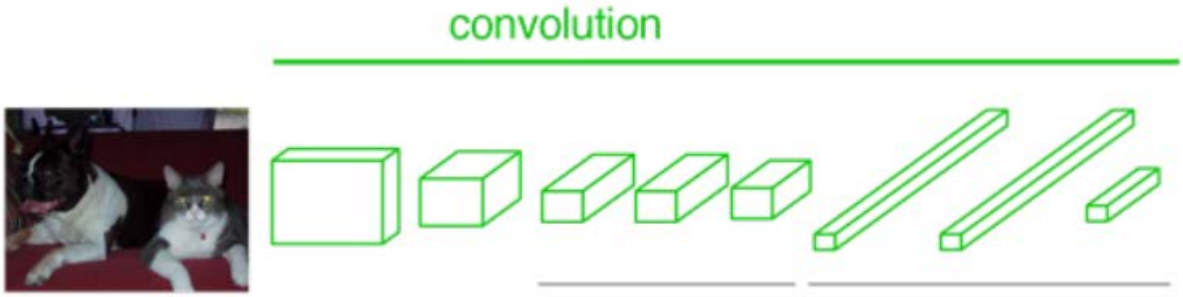


Generally CNN are used for classification purposes. But here in this project, I have extended it for pixel-wise prediction of classes.



The above figure summarizes a basic CNN which has images as input & various classes as output. The model predicts which of the classes are most likely to be present in the image.

The way I move forward in my project is to transform a normal Convolutional Neural Network (CNN) to a Fully Convolutional neural network (FCN). This is done by replacing the Fully Connected (FC) layers with the Convolutional layers.



In this project I have exploited another technique of "skip layers" in which we directly connect initial layer outputs to the later layer inputs. The basic intuition behind this technique is that there is some information that is captured in the initial layers and is required for reconstruction during the up-sampling. So the information in the primary layers can be fed explicitly to the later layers using the skip architecture. The skip layers can actually help in fine tuning the model prediction and can help in getting the model prediction to the level of state of art for image segmentation.

Benchmark model:

The benchmark model used for the project is FCN-8s as proposed in "Fully Convolutional Networks for Semantic Segmentation [1]" . The benchmark model performs similar task of image segmentation on various datasets like VOC 2011, VOC 2012, NYU-Depth v2, etc. However in my project I am solely using VOC 2012 for all training & testing purposes. The benchmark model proposes a skip architecture that combines semantic information from a deep, coarse layer with appearance information from shallow, fine layer to produce accurate and detailed segmentations.

There are many traditional methods to do image segmentation like Region growing, Clustering, split & Merge, but in this project I have used neural networks for classifying pixels into classes, thereby fulfilling the image segmentation task.

Methodology

Data Preprocessing:

Dataset used for the project is VOC Pascal dataset from the Visual Object Classes Challenge 2012 (VOC2012) [2]. It consists of 2913 images of various dimensions. So the first step I am taking in preprocessing is reshaping it to a smaller size in which I will be able to do the training keeping in mind the memory & computation limitations. The size which I have chosen for the model is 320*160. So I am resizing both the input & output data to 320*160. As a standard practice, I am splitting the training data of 2913 images into train/validation/test data. The training tensor will be sized 2500*320*160*3. Similarly, validation tensor will be sized 113*320*160*3 and test tensor will be of size 300*320*160*3.

The next step in preprocessing is to make the dataset to zero centered. As the images are in RGB format, with each pixel value from 0 to 255, I am calculating the channel-wise (RGB channel) mean. After subtracting the mean RGB components from each pixel value of the dataset, I am feeding this input to the model for learning.

Now let's discuss about the training output of the model. VOC2012[2] provides 2913 segmented images. Let's call them trainClass images. I am doing a similar resizing of the trainClass images to 320*160 sized. In the images provided in the dataset, each pixel takes values from 0 to 20 & 255. '255' value stands for an ambiguous class & is generally used for the boundaries of the objects. So first step I am taking is to convert this '255' to '0', thereby converting all the ambiguous classes to background class. So now the trainClass image is 320*160 and each pixel can take 21 values from 0 to 20 (20 inclusive, 0 stands for background). As I want the model to predict each pixel of input image to one of these 21 classes & they are categorical data, so I have done a one-hot encoding for the trainClass. For each pixel I define a 21 sized vector in which only one element will be '1' corresponding to that particular class & rest all will be zero. The training tensorClass will be sized 2500*320*160*21. The validation tensorClass will be sized 113*320*160*21.and the test tensorClass will be of size 300*320*160*21. This is in accordance to the convention I have taken for train/val/test split of dataset.

So basically I will feed 320*160*3 sized input to model & will get 320*160*21 sized output from the model. Later on I will take argmax to find the corresponding class label for each pixel.

Implementation:

In the project I have implemented two architectures using CNN for image segmentation task.

The layers I have used in the architecture are as follows:

Conv2D: This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. The kernel size in the project is 3x3.

MaxPooling2D: Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. The pool size in the project is 2x2.

Activation: Activation function is used to bring non-linearity in the model. 'Relu' and 'softmax' are few commonly used in the project.

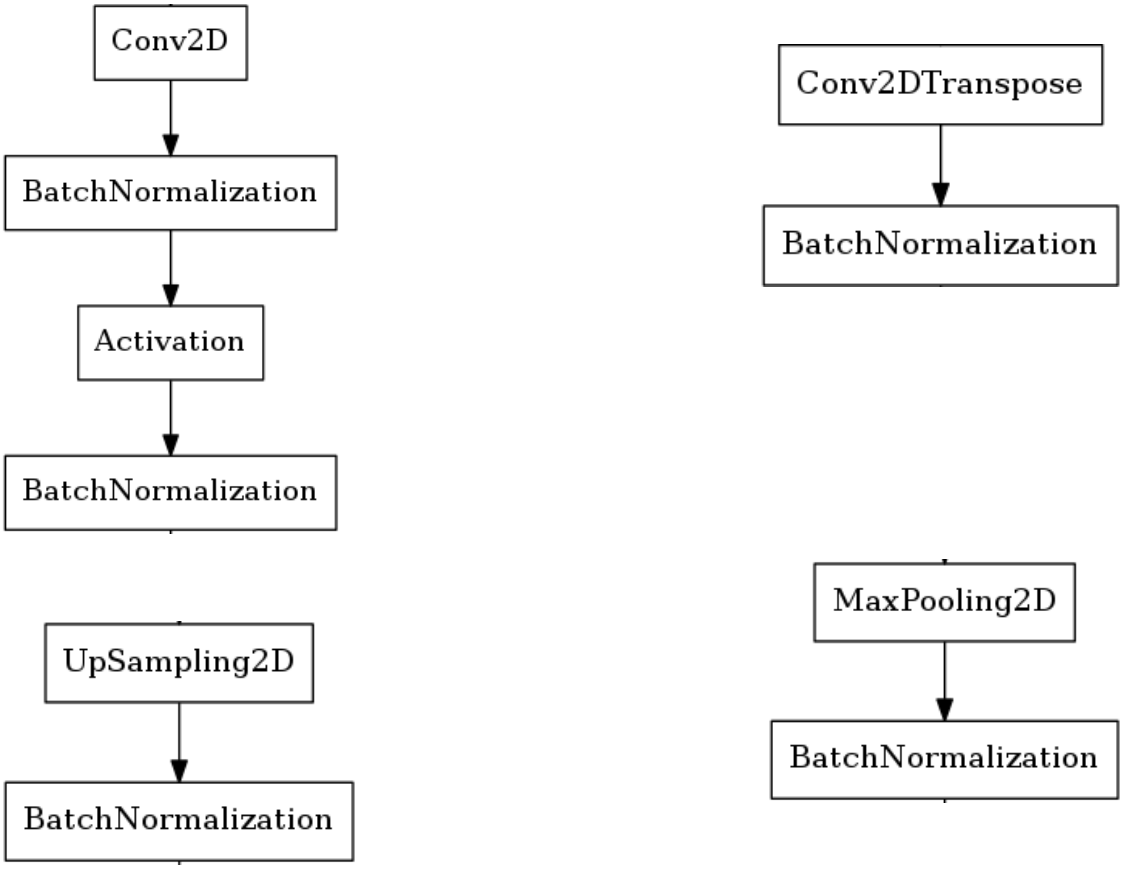
BatchNormalization: It normalizes the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

UpSampling2D: It repeats the rows and columns of the data by size specified in the parameter. The kernel size in the project is 2x2

Conv2DTranspose: It applies convolution with fractional stride. Typically used for upsampling. The kernel size in the project is 3x3

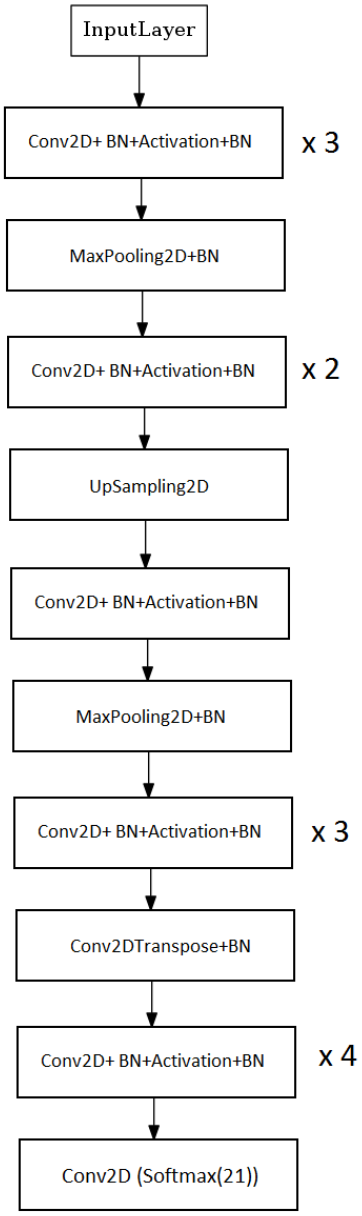
In the initial architecture I have implemented a simple FCN network. I have also made few changes in the architecture like instead of continuously downsampling & then upsampling, doing it in a more continuous fashion of constantly downsampling & upsampling.

Blocks in the architecture looks like the below.



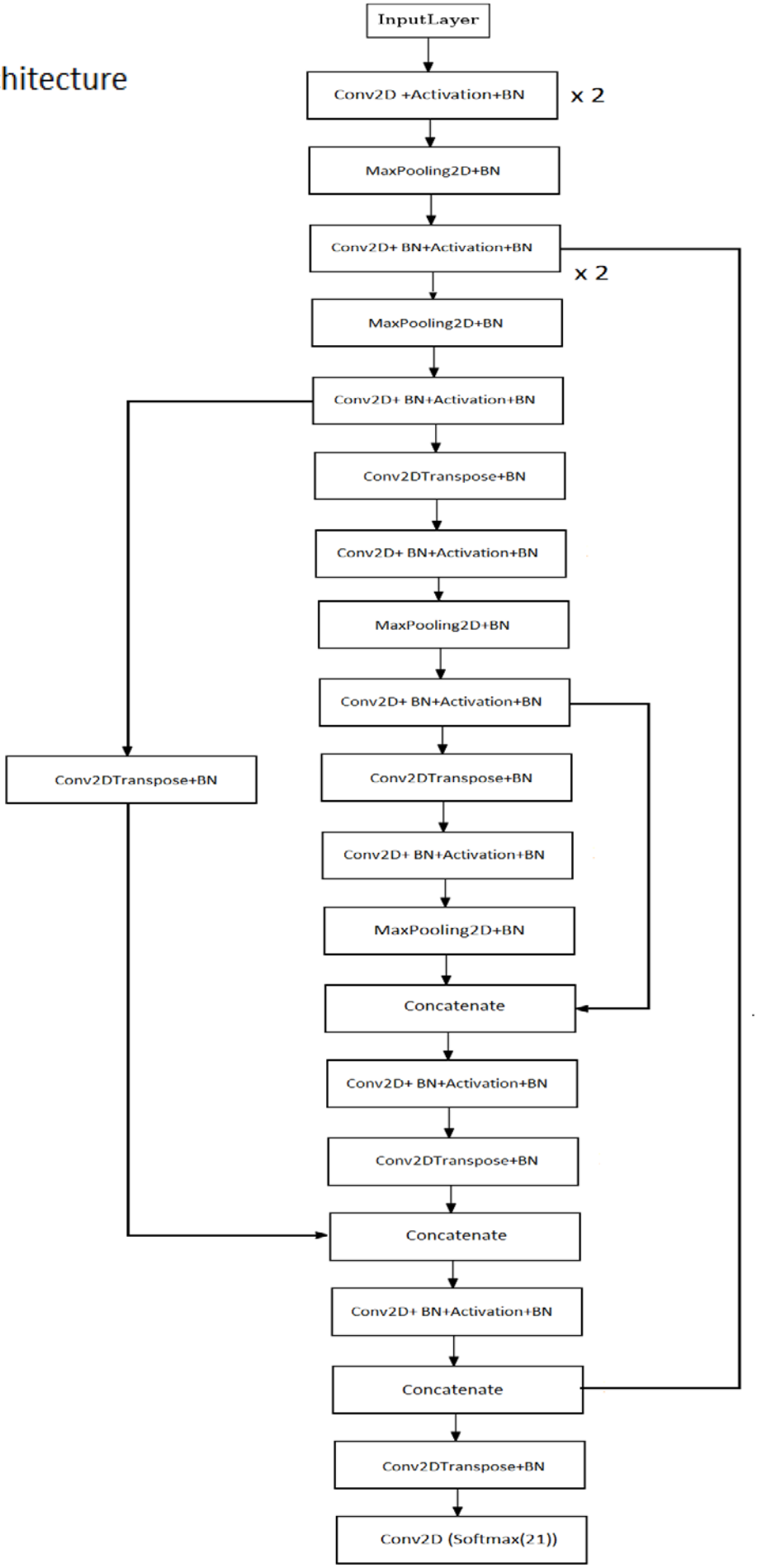
I have added Batch Normalization layers between each subsequent layers so that the output of layers don't fall on the dead 'relu' region.

The first architecture looks like the below:



The intuition behind this architecture was "passing the input images through a set of downsampling & upsampling layers to effectively gain knowledge about the objects in image & to not lose much information because of too much downsampling.

Second architecture



The basic intuition behind second architecture is the usefulness of skip layers in retaining the spatial information.

Few complications faced during the process.

First one of them was choosing an activation function. Initially ‘softmax’ was used in few layers, but it was computationally very expensive & was causing the slowness in training the model. It was overcome by using ‘relu’ which very well brings non-linearity in the model without being computationally expensive. In the last layer as we wanted the probability ‘softmax’ was used.

Another one is the need of ‘BatchNormalization’. As we are using ‘relu’ as activation function, a lot of neurons output may fall in dead relu region, making it tougher for the model to learn. BatchNormalization effectively solves this problem by making output of previous layer zero mean.

Another one was choosing the size of input images. While having larger image at input helps in better training of the model, but it increases the size of output of each layer which in turn makes the network manifold bulkier.

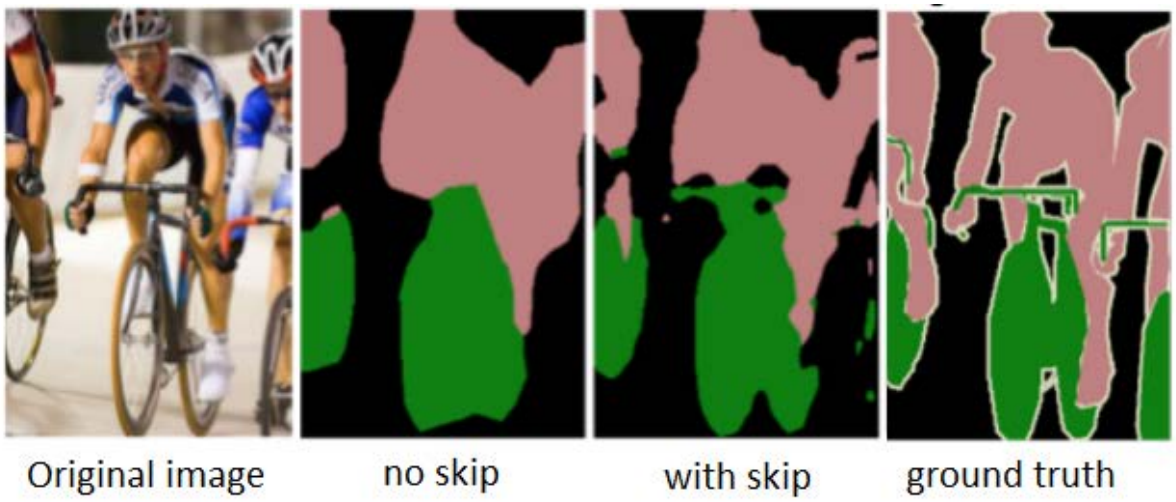
Refinement:

In this project, I have implemented two architectures using CNN for image segmentation task. The first one is a normal CNN used for pixel-wise prediction. The second architecture is the refinement of the first one with the use of skip layers. The problem with the first approach is that we lose some resolution by just doing pooling because the activations were downscaled on a lot of steps. To solve this problem we also get some activation from previous layers and sum/interpolate them together. This process is called "skip" from the creators of this algorithm.

The categorical accuracy & the meanIU accuracy with the first architecture is 0.6855 & 0.2901 respectively whereas with the second architecture is 0.7923 & 0.5161 respectively.

The second architecture beats the first one by considerable margin in the segmentation task. The reasoning that can be given is that the skip layers can actually help the later layers with fine tune details by providing more details which the initial layers had captured. There was some information that was

captured in the initial layers and was required for reconstruction during the up-sampling done using the FCN layer. If we would not have used the skip architecture that information would have been lost as can be seen in first architecture. So the information contained in the primary layers when fed explicitly to the later layers using the skip architecture resulted in better model prediction.



As we can see in the above image, adding a skip layer can fine tune the segmentation task for better accuracy.

Results

Model Evaluation & Validation:

I have used two parameters for model evaluation – categorical accuracy & MeanIU. While the former focuses on the pixel to pixel matching for accuracy computation, in the later I am considering the intersection over the union for accuracy computation.

For calculating the categorical accuracy I am using the keras inbuilt metrics ‘categorical_accuracy’.

The initial analysis was done using sequential architecture as proposed in architecture 1.

It has categorical_accuracy = 0.6855 & MeanIU = 0.29012 on test dataset. The major analysis is done on the second architecture with skip layers as it provides significant improvements on prediction.

I am using keras model_checkpointer to save the model after epoch runs. Keras provides mechanism to give monitor parameter, so as to only update the saved model if the monitor parameter is getting better. I have given two model parameters - 'loss' & 'val_loss'.

Results with monitor 'loss'

Categorical accuracy for 2500 train Images: 0.87811522588729862
Categorical accuracy for 113 validation Images: 0.69706495673255586
Categorical accuracy for 300 test Images: 0.73273944457372031

MeanIU accuracy for 2500 train Images: 0.611870685412
MeanIU accuracy for 113 validation Images: 0.432627511306
MeanIU accuracy for 300 test Images: 0.468261566863

Results with monitor 'val_loss'

Categorical accuracy for 2500 train Images: 0.80629220895767217
Categorical accuracy for 113 validation Images: 0.76924848609266028
Categorical accuracy for 300 test Images: 0.79235182046890262

MeanIU accuracy for 2500 train Images: 0.57963485633
MeanIU accuracy for 113 validation Images: 0.508213861083
MeanIU accuracy for 300 test Images: 0.516137304973

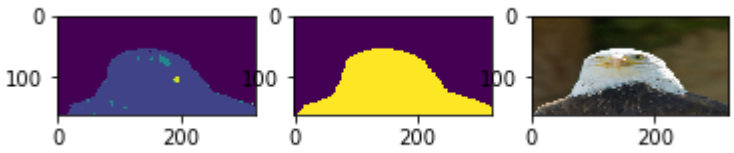
As we can see that model with monitoring parameter 'loss' is performing better on train images & poorly on val/test images, whereas model with monitoring parameter 'val_loss' performing better on train images & poorly on val/test images. The first one relates more to over-fitting than generalization.

Justification:

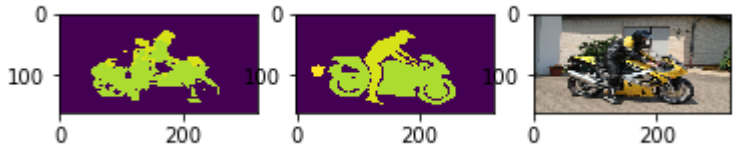
The results obtained here in the project are less than the benchmark model of FCN-8s model. The reason for that are the number of training parameters & epochs. For this project, due to the restricted resources available, I have configured the network with only 1.7M parameters & approx. 300 epochs.

In the skip architecture I have saved the models on 2 monitor parameters ‘loss’ & ‘val_loss’ . We can see that though the ‘loss’ model performs better on train data, ‘val_loss’ model outperforms in test & val dataset. The model has a prediction accuracy of 0.79235 on the test data which is acceptable .

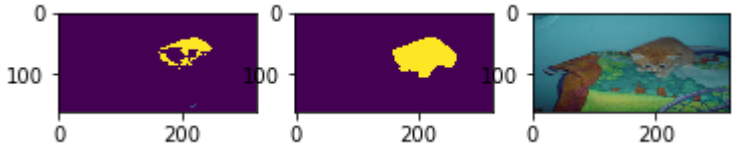
On train data the predicted output is:



On validation data the predicted output is:



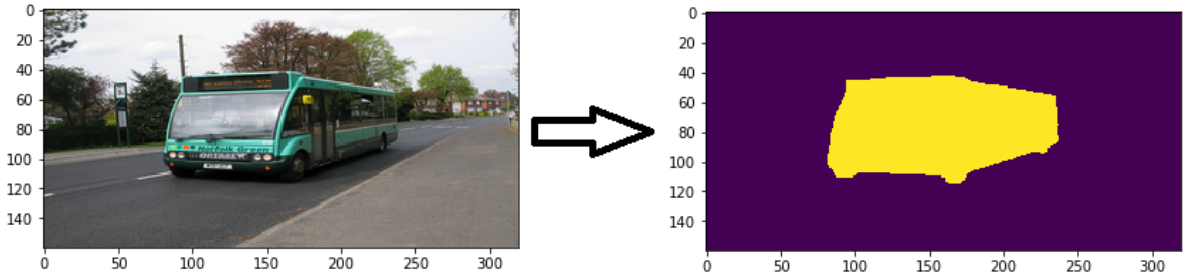
On test data the predicted output is:



Conclusion

Free form visualization:

In this project, I started with the task of image segmentation.



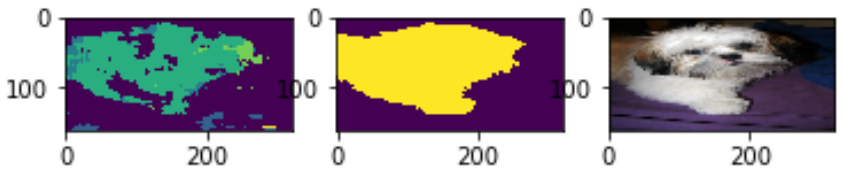
I have trained the model on 2500 images containing 20 classes. Few classes have high occurrence in the images like person, few have very low occurrence like dining table. Accordingly model has learnt few classes better than the other. One issue with the model is the prediction of other classes along with the correct class. In predicted image sometimes, there are other classes too which are not present in the image.

For example in the below image pairs, where the left image is model's predicted image, middle one is ground truth, right one is original image -

```
model predicted clasess- [ 0  6 11 15]
ground truth classes- [ 0 15]
```

In above pair we can see classes 6 and 11 are predicted wrongly.

```
model predicted clasess- [ 0  6  8 11 12 15 19]
ground truth classes- [ 0 12]
```



In above pair we can see classes 6, 8, 11, 15, 19 are predicted wrongly.

We can see in the above two pairs the model is more confident in the image containing human than that of image containing which can be seen by the number of false positive classes detected in dog image. The way it can be mitigated by having a balanced dataset or preprocessing the dataset by few techniques like random under-sampling or random over-sampling.

Reflection:

The core difference between the image detection & image segmentation is that we have to recreate the whole image again in segmented form along with classification of each pixel. This can result in a bulky network with lot of parameters which may require huge hardware. In my project I have tried to implement one such architecture. I have used MaxPooling to reduce the dimensions. Later, I used skip architecture to combine semantic information from a deep, coarse layer with appearance information from shallow, fine layer. Concatenate layer is used for this purpose of combining outputs of two layers.

One interesting thing to learn was analysis of performance of both models - the one with better 'val_loss' or one with better 'loss'. Clearly one with better loss was very accurate with train data but fails with validation / test data. This brings the topic of over-fitting the model and how a model which performs very well during training fails during real world situations-test data.

Another important aspect was how well the model is learning a particular class but not so well learning the other. As there are much more person images in dataset, it is learning person better than any other classes. It tells about the importance of data while training any model.

Improvements:

Many scenarios in real world like self-driving car, face identification etc. require image segmentation as the first & foremost task. Furthermore, they require a real-time model which can predict and at the same time learn the unknown data to constantly make the model better.

In this project, one major limitation was the amount of data & hardware resources. With a lot of training data, this model could have performed better.

This brings up a new concept of transfer learning that is being used extensively now a days. Transfer learning or inductive transfer is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. A lot of research is actively going on in this area, since creation of segmented train images is an expensive task [3].

Another recent advancement is Generative Adversarial Network .Adversarial training has been shown to produce state of the art results for generative image modeling. In this paper [4] , the author has showed the usage of adversarial networks for the sematic segmentation task.

References :

[1] E. Shelhamer, J. Long and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 4, pp. 640-651, April 1 2017.
doi: 10.1109/TPAMI.2016.2572683

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn,and A. Zisserman.The PASCAL Visual Object ClassesChallenge 2012 (VOC2012) Results.http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.htm

[3] Utkarsh Gaur, Matthew Kourakis, Erin Newman-Smith, William Smith, B.S. Manjunath, "Membrane segmentation via active learning with deep networks", *Image Processing (ICIP) 2016 IEEE International Conference on*, pp. 1943-1947, 2016, ISSN 2381-8549.

[4] Semantic Segmentation using Adversarial Networks by
Pauline Luc, Camille Couprie, Soumith Chintala, Jakob Verbeek

[5] https://keras.io/

[6] https://en.wikipedia.org/wiki/Image_segmentation

[7] https://courses.cs.washington.edu/courses/cse576/04sp/notes/ImageSegmentation.ppt

[8] Deep Neural Networks for Object Detection by Christian Szegedy, Alexander Toshev, Dumitru Erhan

[9] https://en.wikipedia.org/wiki/Convolutional_neural_network