Project report submitted in partial fulfillment
of the requirements for the degree of

Bachelor of Technology

By

Vineet Singh - 15UCS159
Vikash Kumar - 15UCS156
Prince Jain - 15UCS101

Under Guidance of
Prof. Anukriti Bansal

Department of Computer Science
The LNM Institute of Information Technology, Jaipur

August 2019

The LNM Institute of Information Technology
Jaipur, India

CERTIFICATE

This is to certify that the project entitled Image Segmentation using Deep Learning, submitted by Vineet Singh (15ucs159), Vikash Kumar (15ucs156) and Prince Jain (15ucs101) in partial fulfillment of the requirements of degree in Bachelors of Technology (B. Tech), is a bonafide record of work carried out by them at the Department of Computer Science, The LNM Institute of Information Technology, Jaipur, (Rajasthan) India, during the academic session 2018-2019 under my supervision and guidance and the same has not been submitted elsewhere for award of any other degree. In my/our opinion, this thesis is of standard required for the award of the degree of Bachelors of Technology in (B. Tech).

August 2019

..... **.....**

Date **Adviser: Prof. Anukriti Bansal**

Dedicated to My Family and Friends and My Mentor.

Acknowledgments

We would like to express our gratitude to our BTP Mentor, Prof. Anukriti Bansal, The LNM Institute of Information and Technology, Jaipur (LNMIIT) for her valuable guidance, encouragement and co-operation during the course of the project and its presentation.

We would like to thank our BTP Panel for giving us this great opportunity to work on the topic (Predicting Movie Success: Using Data Analysis and Machine Learning) which led us to do a lot of research and gave us knowledge about many new things while learning about the project.

We would also like to thank our family and friends who helped us a lot in finalizing and completing the project in a limited time period.

Abstract

In our BTP Project, we try to and successfully perform Image Segmentation for road images particularly the ones that will be fed into our model at some point of time in the future (input sample on which we desire to obtain the semantic segmented results).

For the data selection part of the project, we have taken the CamVid Dataset from the University of Cambridge for training our model. The CamVid dataset is a collection of images containing street-level views captured while driving. This dataset has pixel-level labels for 32 semantic classes including car, pedestrian, road, etc. For the purpose of our BTP project we have taken into account for 12 labelled classes of the available 32. So, in layman terms we try to semantically segment an input image into 12 possible labels.

Link to data: http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/
https://idd.insaan.iiit.ac.in/

A semantic segmentation network works effectively by classifying each and every pixel in an image (from the maximum possible labels), resulting in an image which is semantically segmented by class. Some of the major applications for semantic segmentation in real world scenarios are namely, road segmentation for achieving autonomous driving, segmentation of cancer cells used in medical diagnosis.

Convolutional Neural Networks (CNNs) are driving major advances in many computer vision tasks, such as image classification, object detection and semantic image segmentation.
For performing the image segmentation, we have used the U-Net architecture in which we use RESNET-34 as encoder and the next 34 up-sampling layers to give us pixel wise segmented output of an input image. When we say that we have used pretrained RESNET-34 model as the encoder which was trained on image net dataset, we mean to say that we have trained the CamVid images on a RESNET-34 model and used them to further train on the U-Net architecture for logging high accuracy. To conclude, this FCN with many matrix multiplication, activation functions (ReLU), pooling layers, up-sampling layers, and skip connections gives us a robust model which can segment among 12 classes with 94% accuracy.

Contents

Chapter 1

Introduction

1.1 Area of Work

The rapid advancement in Computer Vision has led to major developments in image classification and object detection techniques. Deep Learning has added to this trend and led the world into the era of AI. Convolutional Neural Networks (CNNs) are the fulcrum of this breakthrough, and have proven to be the most trusted and widely used technique to formulate typical computer vision models and systems that lead to state of the art image classification, object detection and image segmentation.

The major grounds on which image segmentation are performed are labels such as roads, vehicles, buildings, pedestrians, traffic lights, trees, sidewalks, etc. The model tries to understand and then segment the input image pixel wise to obtain a fully segmented image.

For our model to achieve high accuracy and thus make accurate segmentation, some careful analysis and techniques have been accounted for, like, pretraining the dataset on a RESNET-34 architecture, using U-Net architecture for segmentation, IoU evaluation matrix, Adam Optimizer for optimizing our loss function, etc. These techniques of hyperparameter tuning and optimization have been implemented after careful testing and evaluation, helping in achieving higher accuracy brackets.

1.2 Problem Addressed

Via this report, we intend to perform a segmentation task on input images (never seen before by the model) thus classifying the desired image into maximum possible labels (classes). This is done via training a computer vision model using deep neural networks specifically CNNs (and their state-of-the-art versions like R-CNN, etc.) to construct a system robust enough to segment through defined classes with industry level accuracy and standards.

We did a deep research on the theoretical aspects of the U-Net architecture - mainly the process and ideas behind various matrix compressions and bottleneck, the mathematical intuition behind our evaluation metric (IoU), Tradeoff(s) in our model, different optimization approaches to effectively minimize the loss function without having to have our gradient suffer the state of forever approaching towards global minimum.

This type of highly tuned models help to impact high performance in end to end AI products like automated cars, Medical diagnosis - cancer cells detection, etc.

1.3 Existing System

Various researchers and scientists have developed various approaches and techniques to achieve high performance computer vision models capable to do image segmentation with high accuracy. Factually speaking, the community has majorly used the performance of Convolutional Neural Networks to solve this problem. To narrow down, the key problem to overcome is to build systems robust and efficient enough to handle large size image sets and to be able to perform segmentation with high standard accuracy. Building a model is one thing and building a model which can be scaled to requirements and meets real world demands is another. We have kept a standard throughout our project to meet the real world demands.

We have used the U-Net architecture with image sets pretrained on RESNET-34 to build our segmentation system,

The Architecture in particular is :

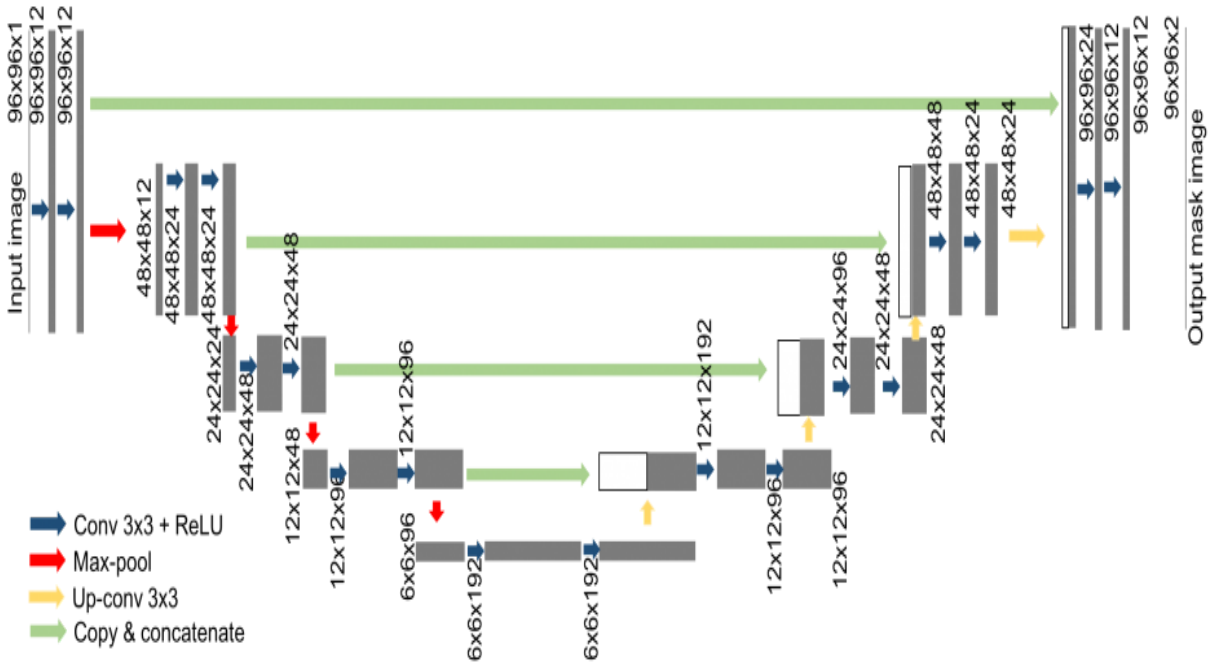


Fig: UNet for semantic seismic image.

Chapter 2

Literature Survey

2.1 Introduction-Deep Learning

Deep Learning is a branch of Machine Learning which completely relies on the concept and performance of Artificial Neural Networks (ANNs). Neural Networks were influenced by the concept of neurons in our brain, as neural network is going to mimic the human brain so we can say that deep learning also tries to mimic the human brain.

The concept of deep learning is not new, it has been around for a couple of years now. Today it sits right on top of the hype list, but much of that credit goes to the exponential increase in the processing power and the abundance of Data.

Formal Definition,

" *Deep Learning is a particular kind of Machine Learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.* "

The human brain is an example of complex design and architecture so the real question is, how do we recreate these neurons in a computer system?

Deep Learning solves this problem by letting us design and create artificial neuron like structures called artificial neural networks, where we have artificial nodes for neurons. Usually, the ANNs are designed as a structure having some neurons for input (X) and some for predicting the output value (Y) and in between, there may be lots of neurons interconnected in the hidden layer of the network topology.

2.2 Architectures

Deep Learning has various underlying applications, majorly in the field of Computer Vision and Natural Language Processing. But the architectures used to compute such problems are the same. To name a few,

1. Deep Neural Network

A neural network with a certain level of complex topology, like having multiple hidden layers in between input and output layers. Deep NNs are capable of handling and processing large non-linear relationships.

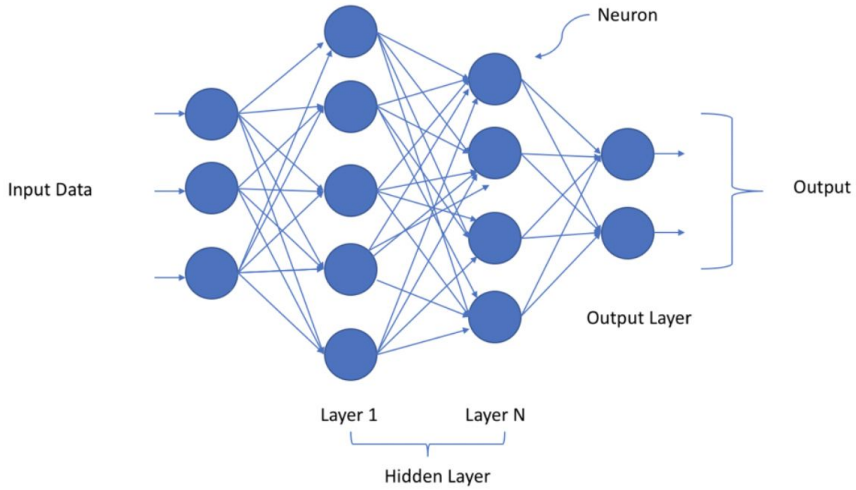


Fig: A deep neural network with N hidden layers

In a simplified version of deep neural network, represented as a hierarchical organization of neurons with connections to other neurons. These neurons pass a signal to other neurons based on the input data, and form a complex network capable of learning using some feedback mechanism (backpropagation).

2. Convolutional Neural Networks

CNNs for short, they are the most popular choice of NNs for performing performance driven Computer Vision tasks. The name 'convolution' is derived from a mathematical operation involving the convolution operation on different functions.

The 4 major stages involved in CNN are, Convolution, Subsampling, Activation, and Fully connected.

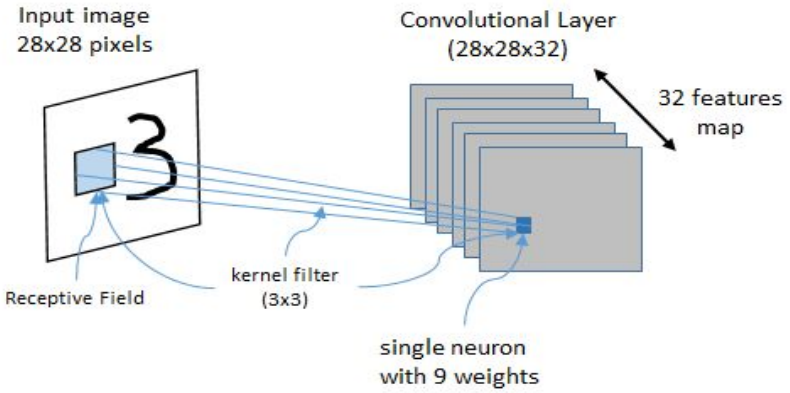


Fig: sample CNN in action by Giancarlo Zaccone

One Advantage - once a segment present in a distinct section of an image is learned, the CNN can recognize that segment be it present anywhere in the image.

One Disadvantage - it depends a lot on the size and quality of the training data and thus are highly susceptible to noise.

3. Recurrent Neural Networks

RNNs are termed 'recurrent' because they perform uniform task for every single element in a sequence, where the input is computationally dependent on the previous steps. Can be visualized as networks having memory, capable of capturing, storing and predicting from the input information.
They have a lot of real-world applications involving the problems where the sequence of the information is crucial, domains such as natural language processing, speech synthesis and machine translation.
Development over the years,

- **Bidirectional RNN**
- **Deep RNN**

One Advantage - unlike a traditional neural net, an RNN maintains the same parameters across all the steps, this significantly reduces the number of parameters needed to learn.

One Disadvantage - cannot be stacked into very deep models, due to the activation function used which makes the gradient decay over multiple layers.

4. Autoencoders

Autoencoders use the principle of 'backpropagation' for the problem of unsupervised nature. They usually represent data via multiple hidden layers such that the output signal has a close resemblance to the input signal.

Some of the major applications of Autoencoders are, Anomaly detection - example detecting fraud in financial transactions, Recommendation systems, etc.

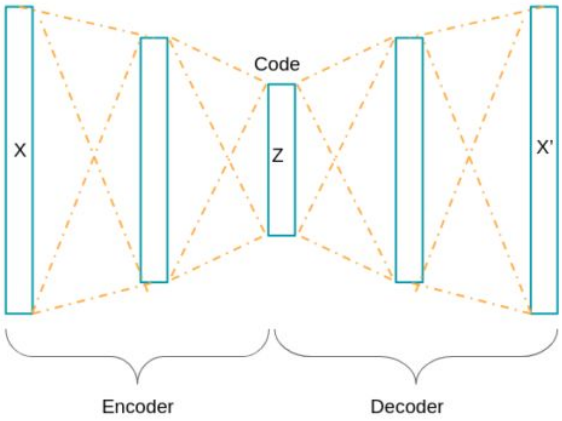


Fig: basic representation of autoencoder

The 4 major types of Autoencoders are,

- Vanilla autoencoder
- Multilayer autoencoder
- Convolutional autoencoder
- Regularized autoencoder

One Advantage - they give models which are primarily based on the data rather than the predefined filters, resulting in less complexity.

One Disadvantage - some types, especially the variational autoencoders, cause a deterministic bias in the model.

5. Generative Adversarial Networks

GANs, a very recent breakthrough in the field of deep learning, finds large and important applications in Computer Vision, especially image generation.

The basic idea of GANs is to train two deep learning models simultaneously. These deep networks try and compete with each other - one model is called the **generator**, and it tries to generate new instances from the given input. The other model is called the **discriminator**, and it tries to classify if a particular instance is from the training data or from the generator.

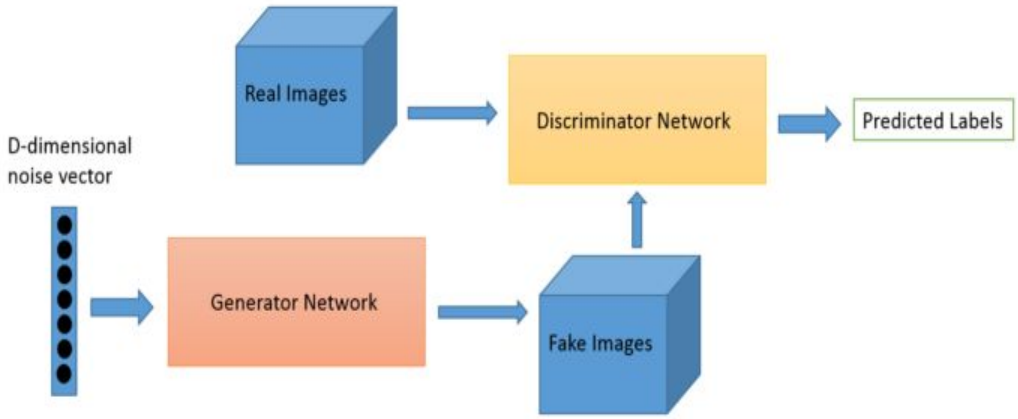


Fig: general architecture of GAN

One Advantage - GANs do not introduce any deterministic bias unlike variational autoencoders.

One Disadvantage - both generator and discriminator are seperate systems and are trained with different loss functions, hence the training time is high.

6. ResNets

CNNs are useful for solving image classification and visual recognition tasks. As the tasks get complex, training of neural network starts to get a way more difficult, requiring additional deep layers to compute the model and enhance its accuracy. Residual Learning is a concept designed to tackle this very problem, and the resulting architecture is popularly known as ResNet. It consists of various residual modules - where each module represents a layer, and each layer consists of functions to be executed on the input.

One Advantage - They are highly modular, that is thousands of residual layers can be added to create a network.

One Disadvantage - If layers in ResNet too deep - errors get harder to detect and cannot be propagated back correctly. If layers get too narrow - the learning loses efficiency.

7. Special Mentions

- **LSTMs**
- **SqueezeNet**
- **CapsNet**
- **SegNet**
- **Seq2Seq**

2.3 Working - Deep Neural Network

Building and working with deep neural networks is a systematic approach. Firstly, we identify the actual use case behind the problem to model a right solution and it should be understandable. One should also check for the feasibility of the problem, whether it should be solved using deep learning or not.

Second, we need to identify the data with distribution covering all the possible scenarios to learn, that is the data should represent the problem well. If we train on data covering some of the learning parameters, then it is highly possible we perform well on the train set but end up having poor score on the test set.

Third, one should carefully consider all possible approaches and choose the most appropriate deep learning algorithm to model the problem. Different approaches and parameter values can have significant impact on the model performance.

Fourth, is to use the chosen algorithm throughout the training process. This helps to evaluate and benchmark the trained results among different algorithms.

Fifth, the final testing of the model should be done on the test set derived from the same data distribution as the train and/or dev set.

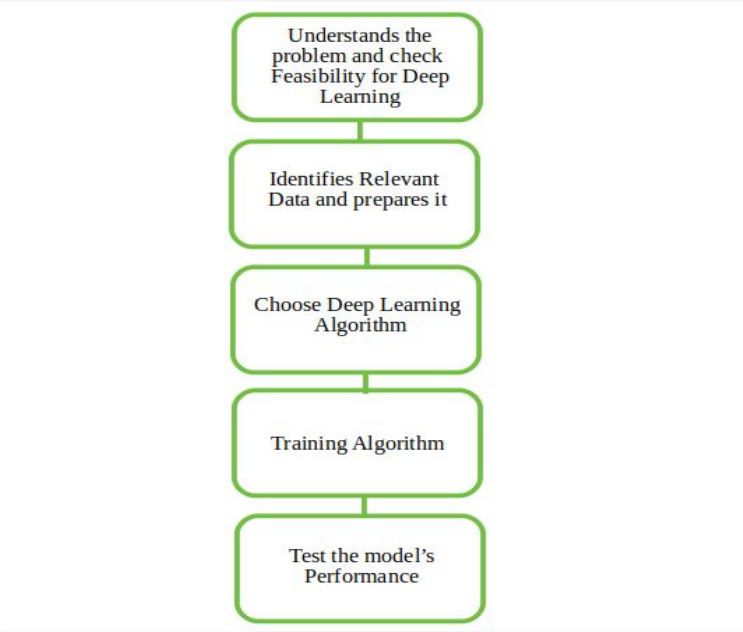


Fig: Working - Deep NNs

Advantages -

- **Best in-class performance on problems.**
- **Reduces the need for feature engineering.**
- **Eliminates unnecessary costs.**
- **Identifies difficult to detect defects with ease.**

Disadvantages -

- **Large volume of data required.**
- **Computationally expensive to train.**
- **No strong theoretical foundations.**

2.3.1 Learning Process of a Neural Network

A neural network is made up of neurons connected to each other; at the same time, each connection of our neural network is associated with a value 'weight' that emphasizes the importance of this relationship in the neuron when multiplied by the input value.

Each neuron has an 'activation function' that defines the output of that neuron. The activation function is used to introduce non-linearity in the processing capabilities of the network. There are several activation functions to choose from.

The most genuine part of Deep Learning is the process of '**Training**' our neural network. Training involves learning the values of our parameters (weights wij and bj biases). This learning process in a neural network is an iterative process of - forward and backward pass, performed by the layers of neurons.

The first phase of learning, called **forward propagation** (the forward pass) occurs when the neural net is exposed to the training data and passed across the entire neural network, for their predictions to be calculated. When the data has crossed all the layers, and all its neurons have made their calculations, the final layer will have result of label predictions for those input examples.

We use a '**loss-function**' to estimate the loss (error) and use to compare and draw conclusions about how good/bad our prediction result was in relation to the correct result (for supervised learning environment). Ideally, we want our cost to be zero, therefore as the model is being trained, the weights of the interconnections of neurons will gradually be adjusted until we obtain good predictions.

Once the loss is calculated by the forward pass, this information is propagated backwards. As the name '**backpropagation**' suggests, starting from the output layer, that loss information propagates to all the neurons in the hidden layer that contribute directly to the output. This process is repeated layer by layer, until all the neurons in the network hace received a loss value that describes their relative contribution to the total loss.

Now that this information has spread backwards, we can adjust the weights of connections between the neurons accordingly. Aim is to make loss as close to zero as possible for the next time we use the network for prediction.

The mathematical technique, '**gradient descent**' is used to make appropriate changes to weights in the form of small increments (done with the help of the calculation of the derivative/gradients of the loss function) allowing the optimization towards the direction of the global minimum.

These steps in general are done in batches (of data) in the successive epoch of all the data that we feed into the network in each iteration.

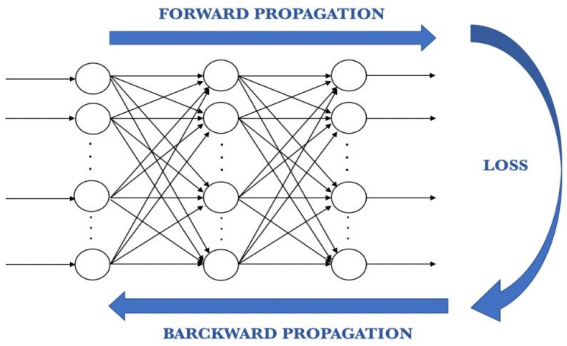


Fig : visual scheme of the above stages

2.4 Convolutional Neural Network

The advancements in COmputer Vision with Deep Learning have been constructed and perfected with time, primarily over a particular algorithm - a Convolutional Neural Network (CNN).

A Convolutional Neural Network or ConvNet is a Deep Learning algorithm capable of taking in an input image, assign importance (weights and biases) to various aspects of the image and can differentiate one from the other, developing an ability to learn these filters/characteristics. ConvNets require comparatively much lower pre-processing then to other classification algorithms.

2.4.1 Why ConvNets over Feed-Forward Neural Nets?

Classic Neural Nets, in cases of extremely basic binary images have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to capture the spatial and temporal dependencies in an image with high success through the application of relevant filters. This architecture performs outstanding fitting to image dataset due to the reduced number of learnable parameters involved. In layman terms, the network can be trained to understand the sophistication of the image better.

Primarily, the role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.

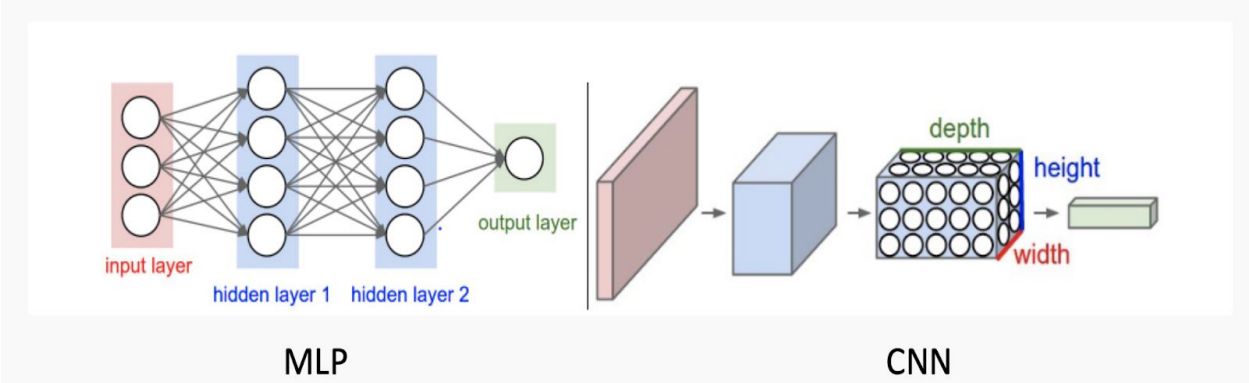


Fig: MLP architecture VS CNN architecture

An overview on MLP vs CNN:

MLPs -

- Do not scale well for images.
- Ignores the information like pixel position and correlation with neighbours.
- Fails to handle translations.

CNNs -

- **Pixel position and correlation with neighbours have semantic meanings.**
- **Elements of interest can be anywhere in the image.**

2.4.2 Different Layers in a CNN

There are three types of layers in a convolutional neural network, namely,

- **Convolution Layer**
- **Pooling Layer**
- **Fully Connected Layer**

Each of the layers performs different tasks on the input data due to each layer having different parameters to be optimized.

1. **Convolutional Layers**

The layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where the majority of user specified parameters are in the network (the most important parameters are the number of kernels and the size of the kernels).

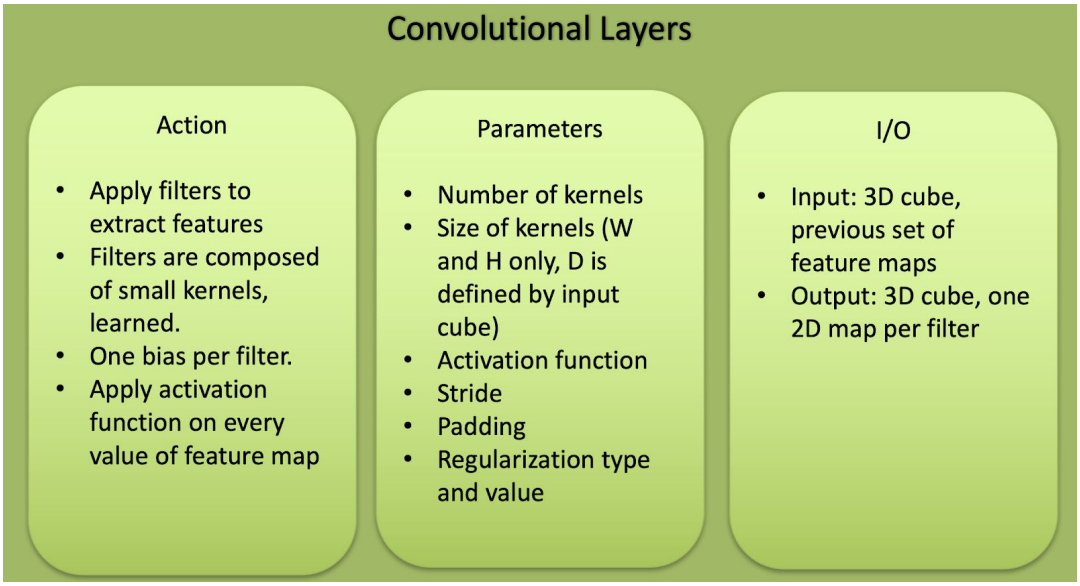


Fig: Features of a convolutional layer

2. **Pooling Layers**

They are very similar to the convolutional layers, except that they perform a specific function such as max pooling - which takes in the maximum value in certain filter region, or average pooling - which takes the average value in a filter region. Used to reduce the dimensionality of the network.

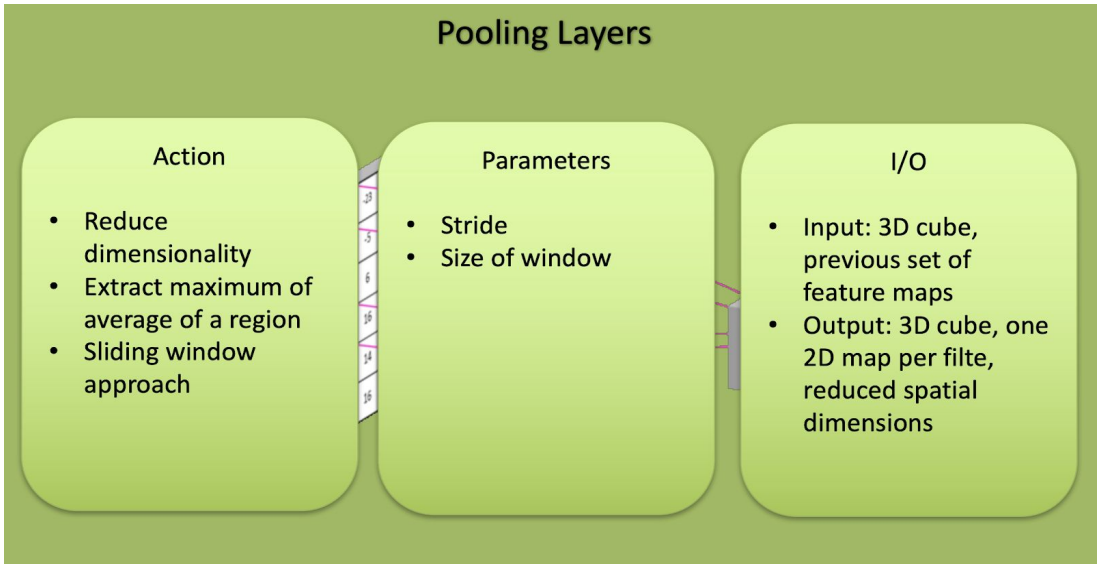


Fig: Features of a pooling layer

3. **Fully Connected Layers -**

Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP.

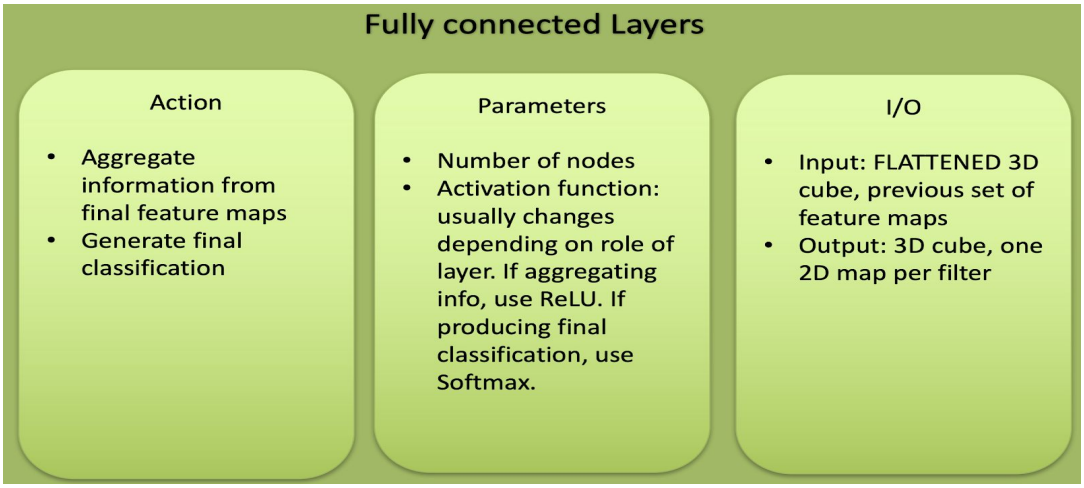


Fig: Features of a fully connected layer.

2.4.3 CNN layers - What do they learn?

- **Each CNN layer learns filters of increasing complexity.**
- **The first layers learn basic detection filters like corners, edges, etc.**
- **The middle layers learn filters that detect parts of an object. Example, for given faces, they can possibly learn to identify eyes, nose, mouth, etc.**
- **The last layers have higher representations to learn like learning to recognize full objects in different shapes and positions.**

Chapter 3

Plan of Work

3.1 Research Work & Experiments

The first step was to research algorithms which can solve this problem. We are able to build computer vision models to detect objects, determine their shape, predict the direction an object will enter, and many other things. You might have guessed it-this is the powerful technology behind self-driving cars!.

We have learned how deep learning and convolutional neural networks can be adapted to solve the problem of semantic segmentation in computer vision. Convolutional neural network semantic segmentation effectively means classifying each pixel in the image. So the idea is to create a map of the fully detected target area in the image. Basically, what we want is the output image in a slide, where each pixel has a label associated with it.The naive approach is to simplify the segmentation task into a classification task. The idea is based on the observation that when an image passes through a CNN, the activation map induced by the hidden layer can provide us with useful information about which pixels have more activations on which category. Our plan is to transform a normal CNN for classification into a fully convolutional neural network for segmentation. First, we obtain a pre-trained convolutional neural network, such as a pre-trained classification and ImageNet. We chose ResNet, and then converts the last fully connected layer to accept Layer of convolution layers. One by one.

When we do this, if we look at where there are more activations, we will get some form of localization.Next step is to fine-tune to the full convolutional network only in the segmentation task. The important thing to note here is that the loss function we use in this image segmentation scene is actually still the common loss function we use for classification, multi-class cross entropy, and not something like L2 loss, as we usually do An image is used when outputting. This is because although you might think, in reality we just assign a class to each output pixel, so this is a classification problem. The problem with this approach is that by doing this alone, we lose some resolution, because activation can be downsized in many steps. There are examples of cyclists on the slide. Different methods for solving semantic segmentation through deep learning are based on a downsampling-upsampling architecture, where the left and right parts have the same size in terms of the number of trainable parameters. This method is also called an encoder-decoder architecture. The main idea is to obtain an input image of size n times m, compress it through a series of convolutions, and then decompress it to obtain an output of the original size of size n times m. What should we do? In order to save the information, we can use skip connections or preserve all convolution and pooling layers by applying depooling and transposition convolution operations in the

decoder section, but at the same place as where max pooling and convolution is applied in convolutional part or encoder part of the network.While possessing many learnable parameters, the model performed well for road signs classification on the CamVid dataset.

Our main goal was to get better accuracy on indian road images so trained this network on IDD Dataset , but results were not that satisfying so we look forward toward better segmentation approach but slower than previous architecture.

Mask R-CNN is basically an extension of Faster R-CNN. Faster R-CNN is widely used for object detection tasks. For a given image, it returns the class label and bounding box coordinates for each object in the image.

The Mask R-CNN framework is built on top of Faster R-CNN. So, for a given image, Mask R-CNN, in addition to the class label and bounding box coordinates for each object, will also return the object mask.

- Faster R-CNN first uses a ConvNet to extract feature maps from the images
- These feature maps are then passed through a Region Proposal Network (RPN) which returns the candidate bounding boxes
- We then apply an RoI pooling layer on these candidate bounding boxes to bring all the candidates to the same size
- And finally, the proposals are passed to a fully connected layer to classify and output the bounding boxes for objects

3.1.1 Backbone Model

Similar to the ConvNet that we use in Faster R-CNN to extract feature maps from the image, we use the ResNet 101 architecture to extract features from the images in Mask R-CNN. So, the first step is to take an image and extract features using the ResNet 101 architecture. These features act as an input for the next layer.

3.1.2 Region Proposal Network (RPN)

Now, we take the feature maps obtained in the previous step and apply a region proposal network (RPM). This basically predicts if an object is present in that region (or not). In this step, we get those regions or feature maps which the model predicts contain some object.

3.1.3 Region of Interest (RoI)

The regions obtained from the RPN might be of different shapes, right? Hence, we apply a pooling layer and convert all the regions to the same shape. Next, these regions are passed through a fully connected network so that the class label and bounding boxes are predicted.

Till this point, the steps are almost similar to how Faster R-CNN works. Now comes the difference between the two frameworks. In addition to this, Mask R-CNN also generates the segmentation mask.

For that, we first compute the region of interest so that the computation time can be reduced. For all the predicted regions, we compute the Intersection over Union (IoU) with the ground truth boxes. We can computer IoU like this:

IoU = Area of the intersection / Area of the union

Now, only if the IoU is greater than or equal to 0.5, we consider that as a region of interest. Otherwise, we neglect that particular region. We do this for all the regions and then select only a set of regions for which the IoU is greater than 0.5.

Once we have the RoIs based on the IoU values, we can add a mask branch to the existing architecture. This returns the segmentation mask for each region that contains an object. It returns a mask of size n X n for each region which is then scaled up for inference.

We used git repo https://github.com/multimodallearning/pytorch-mask-rcnn to train model on our dataset containing classes like-

'Sky', 'Building', 'Road_marking', 'Road', 'Pavement', 'Tree', 'SignSymbol', 'Fence', 'Car', 'Pedestrian', 'Bicyclist', 'Unlabelled'.

3.2 Training

We evaluate our method on CamVid dataset. We trained our models using pretrained resnet-34 without using any extra-data nor post processing module.

- We evaluated our model with Intersection over Union (IoU) metric and the global accuracy (pixel-wise accuracy on the dataset).

- For a given class c, predictions (oi) and targets (yi), the IoU is defined by,
  IoU(**c**) = **Pi** (**oi** == **c** ⋀ **yi** == **c**) /**Pi** (**oi** == **c** ⋁ **yi** == **c**) ,
  where ⋀ is logical and operation, while ⋁ is logical or operation. We compute IoU by summing over all the pixels i of the dataset.

- We used a pretrained model as initialization for down sampling path and used Xavier initialization for the rest of our architecture.

- Used cross-entropy loss function which has turned out to be very good in classification tasks. **cel**=−∑**y***log(**p**) ,where y-actual, p-predicted

- Train our model with Adam optimiser, with an initial learning rate of 3e − 3.

- We decreased the learning rate by factor of 10 as loss function starts decreasing slowly and added exponential weight decay of 0.02 after some time of training so that model won't overfit.

- We have trained it over 30 epochs with two different data/image size.

- All models are trained on data augmented with random crops and vertical flips, lightning, warping.

- We used a batch size of 32 as per our gpu size.

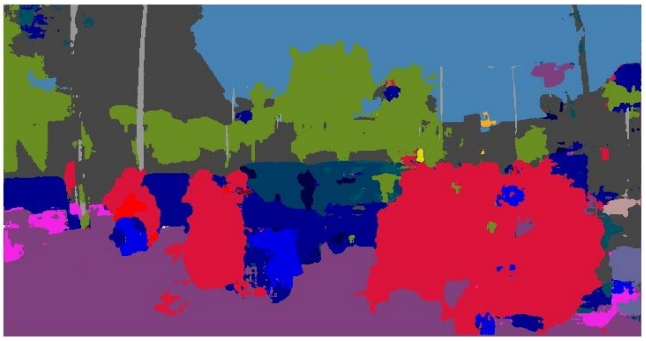
3.3 Results

Some of the observations and inferences made by studying and intellectually inferring the model we have trained are,

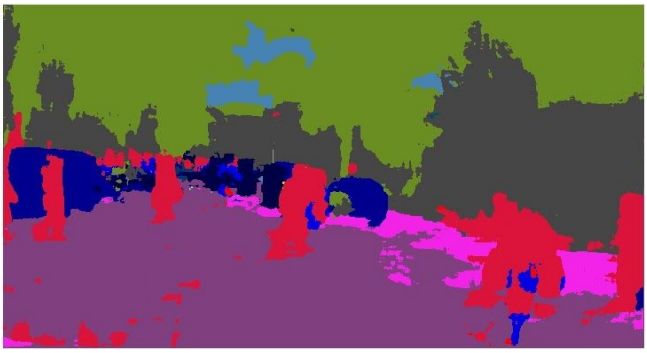
- **We trained our model over 30 epochs and with two different image sizes to get accuracy nearly about 94% which is greater than all the previous results in this data set.**
- **The resulting network is very deep (68 layers). Moreover, it improves the performance on challenging urban scene understanding datasets (CamVid), without neither additional post-processing, nor including temporal information.**
- **Previous best result on this data was 91.2% using FCN of 103 layers.**
- **We have implemented our code using python libraries like fastai, pytorch, numpy, PIL etc.**
- **Mask RCNN is slower than UNet in making inferences but it gives much better results than the Masked RCNN.**

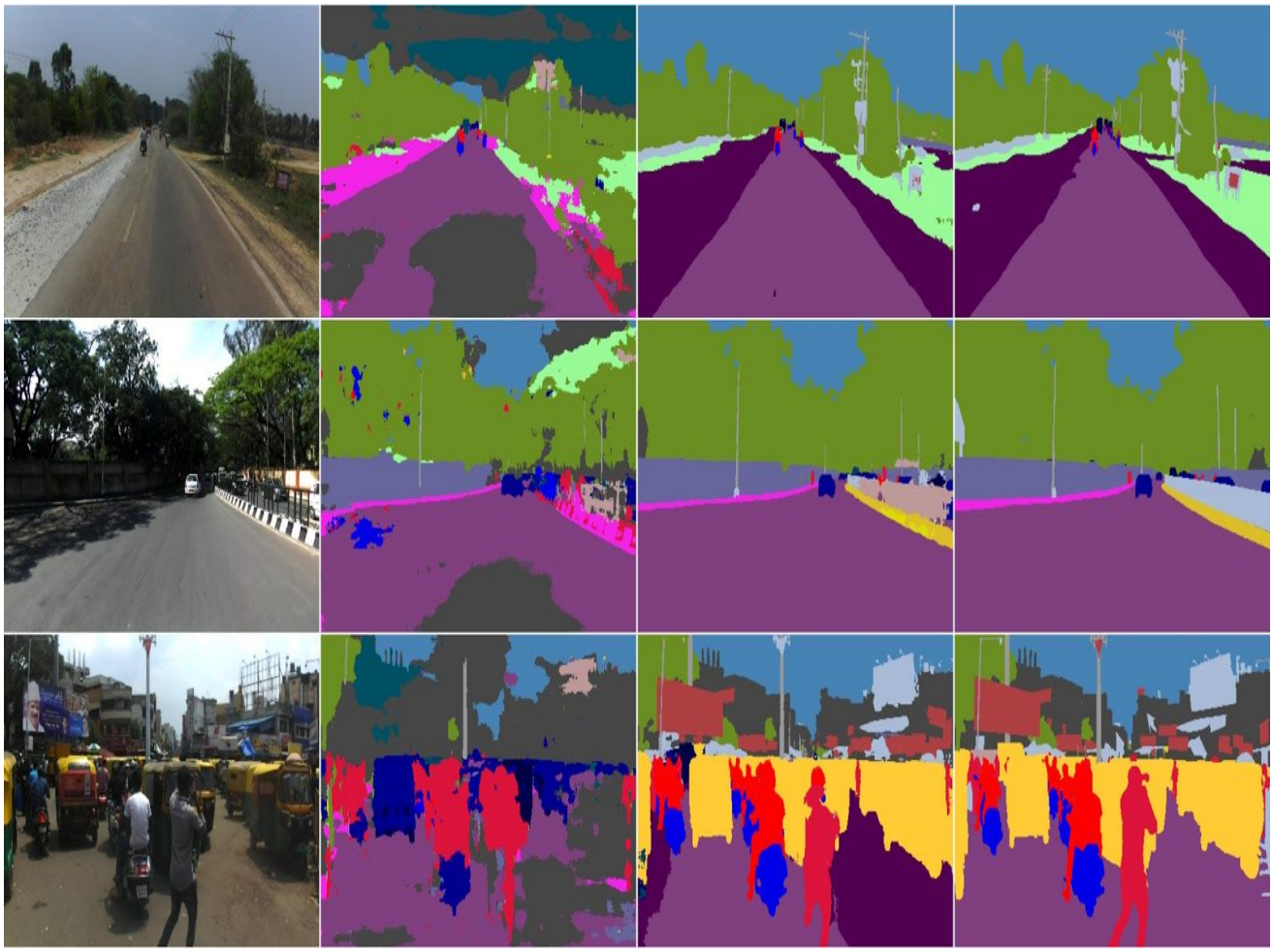
3.3.1 Ground Truth/ Predictions

Output sample for some images taken directly from our testing data set. The images are specifically from Indian road images. The successful segmented results show that the model performs with best efficiency with Indian images used as input data.




TITAN
SAMSUNG
Galaxy
XEROX
fastrack



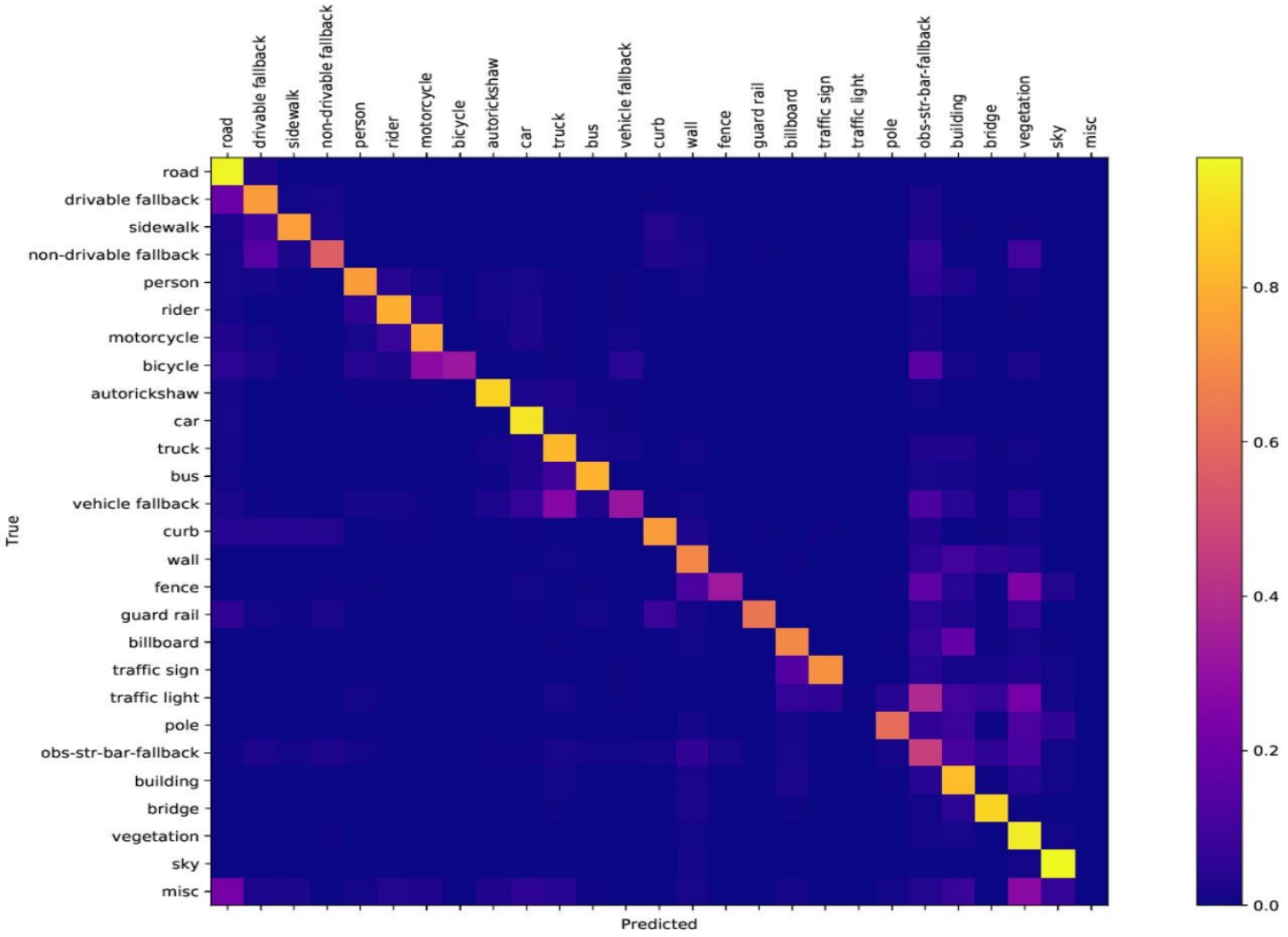


We have tested our model on many different data sets, but it still needs improvement for such diverse conditions of Indian roads.

Majorly, our model confused between,

- **Motorcycle & Bicycle**
- **Billboard & Traffic sign**
- **Obs-str-bar-fallback, Vegetation & Traffic lights**
- **Buildings & Billboards**
- **Vegetation & Wall, Pole, Fence**
- **Drivable, Non-drivable & Vegetation**

3.2.2 The Confusion Matrix



The Confusion Matrix is a table that is generally used to describe the performance behaviour of a classification model (which is what we are trying to achieve with our model). The matrix is used on test data for which the true values are known. Above is the Confusion matrix for 27-class classification by our model.

Chapter 4

Conclusion

The advancement in the field has given us ample options (in types of architecture, use case-specific solutions, customized approach, etc.) to choose from and to apply into our own working solutions. Keeping this in mind, we designed our conclusion to be in similar taste.

In this paper we have explained and implemented the best approach (by implementing a few ourselves) to design a classification model, capable of performing semantic segmentation into the input images with high accuracy.

Two of the most established and used frameworks we dealt with in this paper are,

- UNet Architecture
- Masked R-CNNs

4.1 Which is better ? : UNet vs Masked R-CNN

Released in 2018, Masked R-CNN, developed by Kamming He and his team at FAIR, is one of the world's most powerful algorithms to perform instance segmentation. This algorithm out performed the UNet model in terms of accuracy (although UNet is faster than Masked R-CNN).

Therefore, the answer is, **Masked R-CNN.**

4.1.1 Our Observation

The IoU accuracy for the two models is,

- **Masked R-CNN**
  IoU of 91.2% in our training data & 89.7% in validation data.
- **UNet**
  IoU of 83% in validation data.

4.2 Masked R-CNN

Mask R-CNN is an extension of Object detection algorithm Faster R-CNN with an extra mask head.

The extra mask head allows us to pixel wise segment each object and also extract each object separately without any background (which is not possible by semantic segmentation).

How Successful is Mask R-CNN?

- **Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.**
- **Mask R-CNN outperformed all existing, single model entries on every task, including the COCO 2016 challenge winners.**
- **It gives bounding boxes and segmentation masks on each and every object leading to instance segmentation.**

Chapter 5

Bibliography

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation
[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition
[3] https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html
[4] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick, Mask R-CNN
[5] https://idd.insaan.iiit.ac.in/
[6] Girish Varma, Anbumani Subramanian, Anoop Namboodiri, Manmohan Chandraker, and C V Jawahar, IDD: A Dataset for Exploring Problems of Autonomous Navigation in Unconstrained Environments
[7] Simon Jegou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio, The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation
[8]https://www.pyimagesearch.com/2018/09/03/semantic-segmentation-with-opencv-and-deep-learning/
[9]https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c
[10]https://towardsdatascience.com/computer-vision-instance-segmentation-with-mask-r-cnn-7983502fcad1
[11]https://medium.com/analytics-vidhya/image-segmentation-using-fastai-ddded25f811e