TEST Engineering & Management

January - February 2020
ISSN: 0193 - 4120 Page No. 5485 - 5493

Object Detection Techniques Using Deep Learning: A Survey for Real-Time Applications

[1]Nalini C. Iyer, [2] Tejas Arlimatti, [3]Raghavendra M. Shet, [4]Preeti P, [5]Bhagyashree K
[1] Head of School, [2]Research Student,[3, 4, 5] Assistant Professor
School of ECE, KLE Technological University
Emails: nalinic@bvb.edu, raghu@kletech.ac.in

Article Info
Volume 82
Page Number: 5485 - 5493
Publication Issue:
January-February 2020

Article History
Article Received*: 18 May 2019*
Revised*: 14 July 2019*
Accepted*: 22 December 2019*
Publication*: 27 January 2020*

Abstract:
Technologies in the field of autonomous vehicles (AV) has seen great evolution in recent years. Many automakers are actively attempting to embed advanced technologies into their products, and it's testing at real world scenarios. This field of Autonomous Vehicles can arguably be said to be one of the most daunting topics of today in the field of Intelligent Transportation System (ITS), in particular the aspects like reliability, security, etc and as well as pushing forward for the world's transition towards a highly sustainable future. The sensor technologies of today, however, have several drawbacks; wherein there are high levels of complexity and set-up costs likewise. Thus, this paper aims to accomplish the objective of object detection using only Computer Vision, and specifically for real-time purposes. Thus, this paper aims to explore and compare all the available deep learning based object detection algorithms and arrive at the best model for real-time applications.

Keywords: *Object detection, bounding boxes, image classification, real-time detection.*

I. INTRODUCTION

It can be argued that Computer Vision is one of the hardest and most demanding fields in the domain of Artificial Intelligence;it has posed as a serious challenge to engineers and researchers for decades together. The elicitation of information from images and videos finds application in numerous areas like artificial intelligence, robotics, remote sensing, virtual reality, automation in industries, home etc.The fact that there are several accidents taking place due to driver errors/negligence is what has driven research into this field, and has gained immense popularity for the concepts of making cars that can drive by themselves. RADAR, computer vision GPS and LIDAR serve as popular techniques to facilitate autonomy in vehicles by sensing surrounding environment. This serves as the first state of autonomous functionality, wherein asensor array [1] work incoherence to see, observe and analyze the surrounding environment and thus constitute what is known as the perception module of the vehicle [10]. The very next stage is localization, in which the system knits together the several small, incomplete and unconnected information extracted from the different sensors that helpthe vehicle to know its relative position, velocity, andother physical states from the obstacles (including thedynamic obstacles). In

January - February 2020
ISSN: 0193 - 4120 Page No. 5485 - 5493

the end, system consists of a module which contains the planning stage, wherein the vehicle takes decisions in accordance to its assessment of the situation. The state-of-the-art prototypes currently built by major Industry players employ a LIDAR andRADAR for perception systems. Usually they provide afairly precise360 degree view of the vehicle, this makes it highly aware about the environment around it; sometimes even more than a humandriver. The main downside of these systemsis the cost involved in deploying [2] them. Thus, one cost effective solution could be obtained by using cameras and pairing it with computervision techniques to substitute these systems.

II. ARCHITECTURE SURVEY RELATED WORK

The very first object detection models started off with the slow and arduous process of region search, and then later performed the process of classification. One of the very first techniques that paved the way for modern deep learning models was the Region Convolutional Neural Network (R-CNN) model. In R-CNN, the developed selective search method [3]was an option in contrast to the more thorough search process in an image used to identify the exact location of an object. This model initializedand consolidated several tiny regions of the image using hierarchical or different leveled gathering. Here the last gathering is a box that contains the whole image.

Further, the identified regions are converged by different color spaces and closeness measurements. The yield comprised of some region proposals, which could contain an object by consolidating the smaller regions.

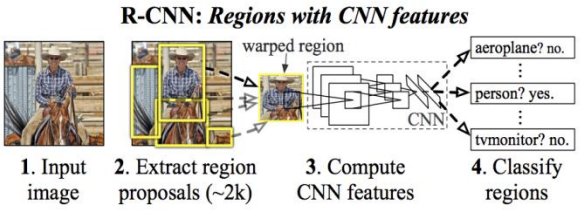

Fig 1: R-CNN architecture

The R-CNN architecture as shown in fig 1 combines the strategy of selective search for detection of region proposals, and utilizes deep learning technique to detect the objects in the selected regions. Every region proposal is then rescaled to coordinate with input dimensions of CNN, from where a 4096-measurement features vector is obtained. Several classifiers are fed with this feature vector to create probabilities that are paired with each class. A SVM classifier is applied on each one of these classifiers, which derive a probability to identify this object for a specific features vector. To minimize the effect of localization error a feature vector that was selected is been given as an input to a linear regressor block which helps in adapting to the shape of the bounding box for the proposed region.

The following improved strategy was called the Fast R-CNN[4] technique, which was intended to lessen the time expended due to huge multiple models that were important to properly investigate all region proposals. A primary CNN with various convolutional Layers acceptthe whole image as input,as opposed toutilizing a CNN for every region of proposals (R-CNN). The Selective Search Method is connected to the produced feature

January - February 2020
ISSN: 0193 - 4120 Page No. 5485 - 5493

maps to distinguish the RoI's. Basically, the extent of the feature maps is diminished by using a RoI pooling layer to acquire a substantial RoI with a fixed height and width as hyperparameters. Each RoI layer feeds each fully-connected layer, making a feature vector. This feature vector is utilized in predicting the object being referred to by utilizing a softmax classifier, and to adjust the bounding box restrictions to it with a linear regressor.

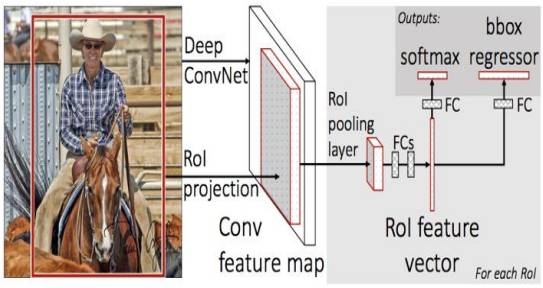

Fig 2: Fast R-CNN

The Region Proposal Network (RPN) was introduced in 2016 togenerate region proposals, generate bounding boxes and to detectobjects. The Faster Region-based Convolutional Network (Faster R-CNN) is a model that combines the RPN and the Fast R-CNN models.

In Faster R-CNN, the model takes in the entire image as input, and produces a set of feature maps accordingly. A 3x3 window slides through the entire feature maps, and then outputs a feature vector that is linked to two fully-connected layers; one for box-regression, and the other for box-classification.

Fig 3: Faster R-CNN

So far, the models discussed above handled the process of object detection as a classification task by building a pipeline that first generated the object proposals, and then send these to classification/regression algorithms. However, recent developments in the field have led to a few methods that pose detection as a regression problem and not classification as such. Two of the most popular ones currently are YOLO and SSD.

The R-CNN, Fast R-CNN, and Faster R-CNN algorithms are two stage detectors, that is they first generate the region proposals and then apply classifiers to the generated region proposals. This makes the models have very high inference times which make them unsuitable for real-time purposes.

The YOLO architecture is a one-stage model that provides for directly predicting the class probabilities and the bounding boxes in a single evaluation, using only one network. Real time predictions are thus made possible, thanks to its simplicity.

A full image is taken as input by this model. The image is further divided into an equally spaced SxS grid. 'B' number of bounding boxes with a certain score is predicted by each of the cells. The probability of detecting the object, multiplied by the IoU(Intersection over Union) between the predicted truth boxes and the ground truth is called the confidence.

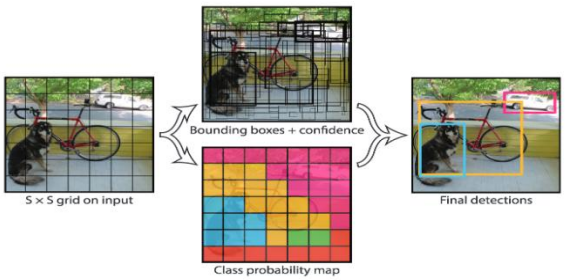

Fig 4. YOLO example

The CNN used in this model is inspired by the GoogLe Net[7] model, which introduced the

January - February 2020
ISSN: 0193 - 4120 Page No. 5485 - 5493

Inception module. The system has 24 convolutional layers which is trailed by 2 completely associated layers toward the end. Reduction layers with 1x1 channels are trailed by 3x3 convolutional layers, and replace the underlying inception modules. The Fast YOLO (Tiny YOLO) model is a lot lighter variant of the YOLO and comprises of just 9 convolutional layers, and also lesser number of channels. The greater part of the convolutional layers in this model is pre-trained utilizing the ImageNet dataset.

The last layer of the model (fully associated layer) givesa S*S*(C+B*5) tensor which relate to all predictions for all the cells of the matrix. C is the generated number of probabilities for each class. B refers to the number of anchor boxes per cell, each of these boxes beingrelated to 4 coordinates (center, width, and height and the confidence value).

In the older models, the generated bounding boxes generally contained a single object. The YOLO model, fig 5 though, can predict a high number of bounding boxes. Thus, there is the

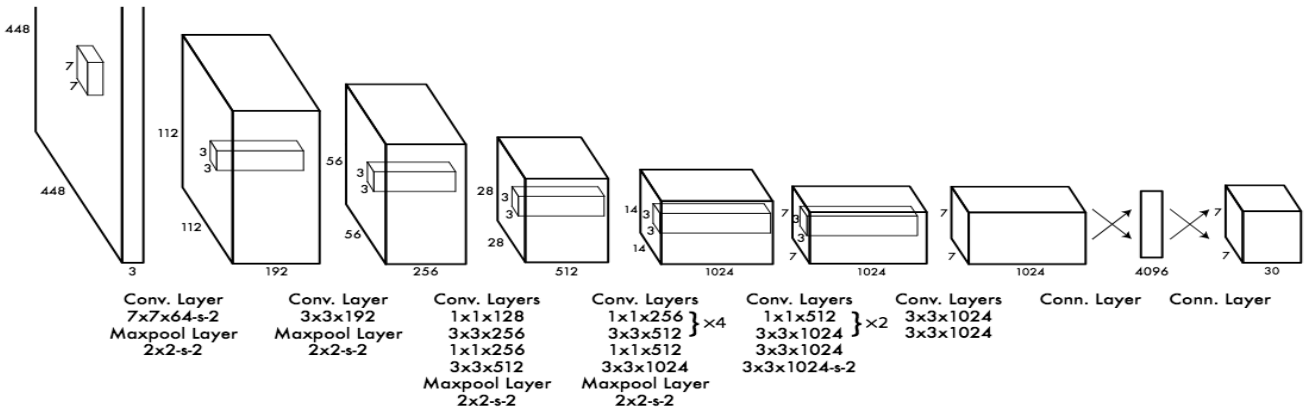

Fig 5. YOLO architecture

Possibility of there being several boundingboxes that contain no object. Thus, the Non-Maximum Suppression (NMS) method is used to counter this, and isapplied at the very end of the network. It provides for merging the highly-overlapping bounding boxes of the same object into a single bounding box.

Along the lines of the YOLO model, the Single-Shot Detector[8] (SSD) was developed in 2016 that can predict bounding boxes and class probabilities, all at once, utilizing a single end-to-end CNN architecture.

The SSD model, like the YOLO model, accepts the entire image as input, which then goes through numerous convolutional layers having distinctive channel sizes (10x10, 5x5 and 3x3). Feature maps from the convolutional layers at different positions of the system are utilized in predicting the bounding boxes. They are handled by a certain set of convolutional layers that consist of 3x3 channels, which are called additional element layers that assist in delivering a set of

January - February 2020
ISSN: 0193 - 4120 Page No. 5485 - 5493

bounding boxes that are like the anchor boxes of the Fast R-CNN model [11] [12].

Each of the produced boxes has 4 parameters; the co-ords of the center, the width, and the height. Additionally, in the meantime, it alsodelivers a vector of probabilities that correspond to the confidence over each class of objects.

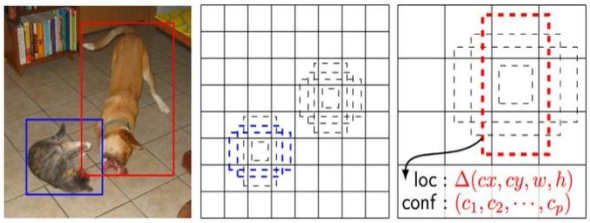

Figure 6. SSD Framework

The Non-Maximum Suppression strategy is utilized at the end of the SSD model like the YOLO, to preserve only the most relevant bounding boxes. The Hard Negative Mining (HNM) technique is then applied, on the grounds that for this situation the NMS boxes comprise additionally of a few negative boxes that are predicted. It comprises of choosing just a subpart of these boxes that were formed during the training. The boxes are then ordered by their confidence levels, and the best is chosen based upon the negative/positive ratio, which is generally at most 1/3.

A. Choosing the algorithm

The desired characteristics of a desired model should be:

- Should have high MAP (Mean Average Precision).
- Should have minimum inference time.

An ideal model that encompasses both of the above-mentioned properties is not possible, so one has to be chosen that has a judicious mixture of both.

Due to the high FPS requirements of the autonomous system, we have decided to choose the Tiny YOLO algorithm because on decent mid-tier GPU's, it can provide for around 45 FPS, which is suitable for real-time object detection purposes.

We've implemented the entire YOLO model using the PyTorch framework because it provides for easy GPU implementation, by utilizing the NVIDIA CUDA platform. The Tiny YOLO is considerably faster than the YOLO, with only around 10% decrease in mAP. The YOLO model can also be used for real-time applications, but it would require the use of an extremely high-end GPU like the NVIDIA Titan X, or Titan V, whereas the Tiny YOLO can provide for reasonable output FPS on a moderate GPU like the NVIDIA 1060. The architecture of Tiny YOLO is similar to that of YOLO, but it consists of only 9 convolution layers and much fewer filters, thus greatly decreasing the inference time. This model has been trained on the Pascal VOC dataset which consists of 80 objects. We have used this pre-trained model, since it's readily available and is license-free. The output for the classes of objects that one doesn't need for object detection can easily be masked, at no loss or gain in performance. Hence, we have left out only the classes of objects that we can expect for the autonomous car [13] to find in its path or around it, so that no incorrect detections are made that can disrupt the process.

January - February 2020
ISSN: 0193 - 4120 Page No. 5485 - 5493

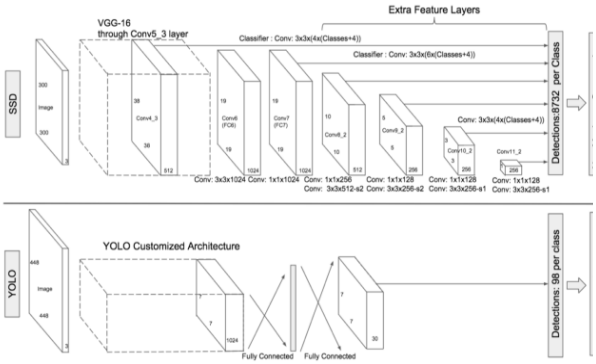

Fig 6 : Differences in architecture between SSD and YOLO

We propose for architecture based on the Tiny YOLO model, and implemented using the PyTorch framework for the object detection purpose. The live camera feed is fed to the CPU which splits the feed into frames, filling a frame buffer. This frame buffer is then used by the GPU that takes one frame at a time, and applies the convolutional network from the model to it. The softmax layer at the end predicts the class of the object, and if background is detected, it does not report it. Also, since YOLO model also runs regression techniques for localizing the objects, the bounding boxes for the object are obtained as soon as the object is detected.

III. EXPERIMENTAL RESULTS

The YOLO architecture was chosen over the SSD because the SSD adds extra feature maps from different layers on top of YOLO, which makes it a little slower, although it increases its accuracy by a small margin. This also results in higher memory usage of SSD over YOLO.

The other metric used in comparison was the MAP (Mean Average Precision)[10]. This is a useful metric in algorithms that predict the location of objects along with the classes. Hence, it can be used for evaluating Localizationmodels, Object Detection Models, and Segmentation Models.

Every image in an object detection problem could have multiple objects of different classes. As was mentioned before, the model has to be assessed both for classification and localization performance.

This is where MAP comes into the picture. It is basicallythe maximum precisions average at various recall values.

Thus, MAP for a collection of queries is mean of average precision scores for each of the queries.

$$\text{MAP} = \frac{\sum_{q=1}^{Q} \text{AveP(q)}}{Q}$$

where, Q is the number of queries.

The PASCAL (Pattern Analysis, Statistical Modelling and Computational Leaning) VOC (Visual Objects Classes) project is a standard dataset that is used to compute the standardized values for mAP by the various object detection algorithms. This dataset has also been used for training of objects by certain models like SSD, etc.

The YOLO model's got around 63.7% mAP score for around 2007 PASCAL VOC dataset and an approximate of 57.9% mAP score for 2012 PASCAL VOC dataset. The Fast YOLO (Tiny) model shows much lower scores, but has much better performance in the form of better FPS.

The SSD modelhas been proven to obtain mAP scores of 83.2% for around 2007

January - February 2020
ISSN: 0193 - 4120 Page No. 5485 - 5493

PASCAL VOC test dataset, and around 82.2% over 2012 PASCAL VOC test dataset. For the test-dev dataset of 2015 COCO (Common Objects in Context) challenge, which is another open-source dataset provided for object detection purposes with over 330,000 images, they've secured a score of 48.5% for an IoU of 0.5, 30.3% for an IoU of 0.75 and 31.5% for the official mAP metric.

Model	mAP	FPS	Real-time Speed
Fast-RCNN	70.0%	0.5	No
Faster-RCNN	73.2%	7	No
YOLO VGG-16	66.4%	12	No
YOLO	63.4%	21	No
Fast YOLO	52.7%	29	Yes

Fig 7.mAP and FPS differences

Thus, it is clear from the above discussion that Fast YOLO (Tiny YOLO) has the highest FPS amongst allmodern models, and is the best suited for real-time applications like object detection for autonomous vehicles.

The proposed object detection pipeline was carried out on a laptop having a 2.2Ghz Intel i5 processor with 8GB RAM, and a NVIDIA Ge-Force 940M 2GB VRAM GPU. The development was done entirely in Python, though there also exist Java implementations of YOLO, because NVIDIA
CUDA framework for accessing the GPU can be easily implemented in Python using the PyTorch framework. Frameworks that can instead be used are Google's Tensorflow, Caffe, among others, but they differ in their ease of incorporating GPU access to their code.

We obtained an output FPS of around 20 when themodel was run on the i5 laptop with GTX 940M, and around 29 on a laptop running the more advanced GTX 1050 Ti 4GB VRAM. The major point being that the Tiny YOLO model doesn't require as much memory as it does GPU CUDA cores and computing power.

Model	AverageFPS	Inference time
YOLOv3	12.8	78ms
Tiny YOLOv3	26.6	37ms

Figure 8.Observed FPS on GTX 940M

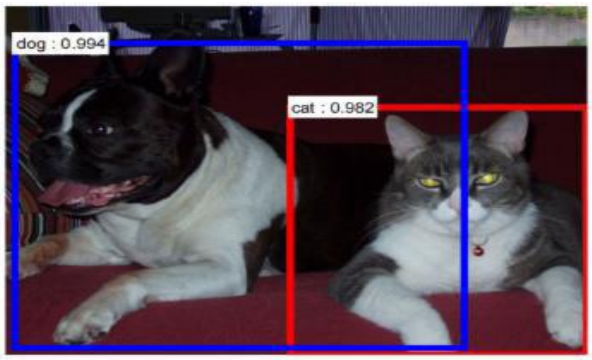

Figure 9. Multiple object detection

January - February 2020
ISSN: 0193 - 4120 Page No. 5485 - 5493

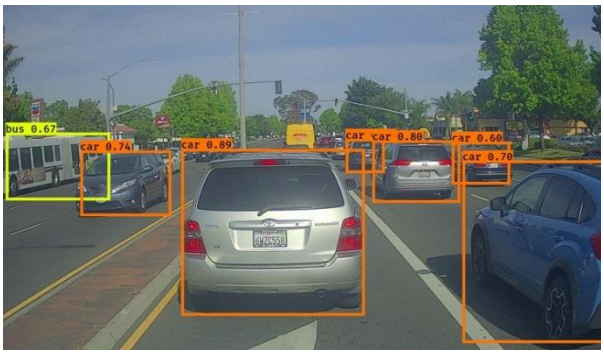

Figure 10.Vehicles on a freeway

IV. INFERENCES AND DISCUSSION

In this paper, we have demonstrated the evolution of object-detection algorithms, and after evaluating their performance metrics, have chosen the best method to implement the real-time object detection. It is observed that the YOLO models give better performance when compared to SSD, since they are more advanced and require lesser memory. The Tiny YOLO model can be implemented on a fairly medium-tier GPU, providing a good output FPS of around 30-40, which is sufficiently real-time for the object-detection systems of autonomous vehicles. The full YOLO model can also be used, provided they have the higher-end GPU's like the NVIDIA GTX 1080, NVIDIA GTX 1080 Ti, which can result in fairly reasonable FPS for detection purposes.

V. REFERENCES

1. Appiah, Naveen and NitinBandaru. "Obstacle detection using stereo vision for self-driving cars." (2015).
2. Häne, Christian & Sattler, Torsten&Pollefeys, Marc. (2015). Obstacle Detection for Self-Driving Cars Using Only Monocular Cameras and Wheel Odometry. 10.1109/IROS.2015.7354095.
3. Dhanakshirur R.R., Pillai P., Tabib R.A., Patil U., Mudenagudi U. (2019) A Framework for Lane Prediction on Unstructured Roads. In: Thampi S., Marques O., Krishnan S., Li KC., Ciuonzo D., Kolekar M. (eds) Advances in Signal Processing and Intelligent Recognition Systems. SIRS 2018. Communications in Computer and Information Science, vol 968. Springer, SingaporeR. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1440-1448. doi: 10.1109/ICCV.2015.169
4. S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017.
5. J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788.
6. M. Al-Qizwini, I. Barjasteh, H. Al-Qassab and H. Radha, "Deep learning algorithm for autonomous driving using GoogLeNet," 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, 2017, pp. 89-96.
7. Liu, Wei et al. "SSD: Single Shot MultiBox Detector." ECCV (2016).
8. J. Hosang, R. Benenson and B. Schiele, "Learning Non-maximum Suppression," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 6469-6477. doi: 10.1109/CVPR.2017.685
9. K. Li, Z. Huang, Y. Cheng and C. Lee, "A maximal figure-of-merit learning approach to maximizing mean average precision with deep neural network based classifiers," 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, 2014, pp. 4503-4507.
10. PillaiPreeti et al. Digital Signal Processing: An Abstract Mathematics to Real World Experience. Journal of Engineering Education Transformations, [S.l.], jan. 2016. ISSN 2394-1707. Available at:

January - February 2020
ISSN: 0193 - 4120 Page No. 5485 - 5493

<http://journaleet.org/index.php/jeet/article/view/85546>. Date accessed: 25 Oct. 2019. doi:10.16920/jeet/2016/v0i0/85546.

11. Maralappanavar S., Iyer N.C., Maralappanavar M. (2019) Pedestrian Detection and Tracking: A Driver Assistance System. In: Shetty N., Patnaik L., Nagaraj H., Hamsavath P., Nalini N. (eds) Emerging Research in Computing, Information, Communication and Applications. Advances in Intelligent Systems and Computing, vol 882. Springer, Singapore
12. JyotiPatil, SatishChikkamath, JyothiHalaharavi, BasavrajHosur, Sharadakabadagi, Nikita Kulkarni "RFID Loco Tracking Using IOT"International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
13. Bhagyashree K., Ramakrishna S., Kumar P. (2019) Analysis of PAPR for Performance QPSK and BPSK Modulation Techniques. In: Shetty N., Patnaik L., Nagaraj H., Hamsavath P., Nalini N. (eds) Emerging Research in Computing, Information, Communication and Applications. Advances in Intelligent Systems and Computing, vol 882. Springer, Singapore