**Case study: Lane Detection for Autonomous Vehicles using Computer Vision Algorithm**

|  |  |  |
| --- | --- | --- |
| **Course Code** | **19AI621** | |
| **Course Name** | **Computer Vision** | |
| **Course Instructor** | **Dr. Senthilkumar T** | |
|  | |  |
| **Team Members** | **Roll Number** | **Contributions** |
| Abhishek Gopinath | CB.EN.P2AID20002 | Dataset, Analytical/Analysis Questions, Block Diagram, Frequency Domain Filters |
| Alan Henry | CB.EN.P2AID20010 | Dataset, Analytical/Analysis Questions, Harris Corner Detection, |
| Jiss Joseph Thomas | CB.EN.P2AID20024 | Dataset, Analytical/Analysis Questions, Spatial Domain Filters |

**1. Problem Statement/Objective:**

To detect lanes on the given dataset of video or images of roads or as real-time, using computer vision algorithms which could be helpful in the proper implementation of autonomous driving.

**2. Dataset Description:**

The dataset given to our problem statement could be either as a set of images or as a video format. There are more than 1000 images for the dataset containing images and two or three videos for detecting the same.

|  |  |
| --- | --- |
| **Type** | Colour Image |
| **Size** | Multiple sizes |
| **Row \* Column** | NA |
| **Resolution** | NA |
| **Colour Model** | RGB |
| **Format** | PNG |
| **Data Acquisition** | Web, Camera |
| **No of classes** | 3 |
| **No of Images** | 853 |
| **Annotation** | PASCAL VOC format. |

**Sample Images: **

**3. Analytical Questions/Statistical Questions/Prediction level Question**

**Analysis**

1.How many vehicles in the frame

2. what are the objects detected in it

3. How many pedestrians are detected in it

4. Detecting road signs.

5. Objects on the interest area.

6. Lines inside the interest area.

7. Traffic signals detected

8. Wet area inside the interest area.

9. Dividers inside the frame.

10. How far the lanes to be detected.

**Analytics**

1. Why vehicles are less in a specific area

2. Why more pedestrians are found at a certain point

3. Why the road is wet?

4. What type of road we are travelling on.

5. What time do most people drive

6. Do people dim their light while passing

7. How often a certain route is taken.

8. How many speed breakers were detected to analyze whether there is a school nearby.

9. Why there is no horn sign detected

10. Why vehicle in the front is slowing down

**4. Block Diagram:**

****

**5. Preprocessing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Domain** | **Filter** | **Input Image** | **Output Image** |
| **Spatial Domain** | **Image Smoothing**  ***Average Filter*** |  |  |
| ***Weighted Average Filter*** |  |  |
| ***Gaussian Blurring*** |  |  |
| ***Median Filter*** |  |  |
| ***Image Sharpening*** |  |  |
| ***Roberts Filter*** |  |  |
| ***Sobel Filter*** |  |  |
| ***Gamma Transform*** |  |  |
| ***Log Transform*** |  | gamma = 0.8 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Domain** | **Filter** | **Input Image**  **(with added noise)** | **Output Image** |
| **Frequency Domain** | ***Low Pass Filter*** | **Decentralized Image:** |  |
| ***High Pass Filter*** | **Decentralized Image:** |  |
| ***Ideal Low Pass Filter*** |  |  |
| ***Ideal High pass Filter*** |  |  |
| ***Butterworth Low Pass Filter*** |  |  |
| ***Butterworth High Pass Filter*** |  |  |
| ***Gaussian Low Pass Filter*** |  |  |
| ***Gaussian High Pass Filter*** |  |  |

**Observation after Comparison Between Spatial and Frequency Domain Filters:**

**Spatial Domain:**

**Input -> Image Processing -> Output**

**Frequency Domain:**

**Frequency + Distribution -> Image Processing -> Inverse Transformation -> Output**

* The spatial domain deals with the image plane itself whereas the Frequency domain deals with the rate of pixel change.
* The spatial domain works based on direct manipulation of pixels whereas the Frequency domain works based on modifying Fourier Transform.
* The spatial domain takes less time to computer whereas the Frequency domain takes more time to compute.

**6. List of Features:**

|  |  |  |
| --- | --- | --- |
| **Feature Name** | **Purpose** | **Category**  **[Image/Vision]** |
| Edges | To detect lanes | Image or Video |
| Blobs | To detect vehicles | Image or Video |
| Corner | To identify the type of vehicles o object, present in the data | Image or Video |
| Ridges | To identify the shape and outline the objects | Image or Video |

**7. Feature Detection and Tracking:**

* ***Harris Corner Detection:***

1.Take the grayscale of the original image

2. Apply a Gaussian filter to smooth out any noise

3. Apply Sobel operator to find the x and y gradient values for every pixel in the grayscale image

4. For each pixel p in the grayscale image, consider a 3×3 window around it and compute the corner strength function. Call this its Harris value.

5. Find all pixels that exceed a certain threshold and are the local maxima within a certain window (to prevent redundant dupes of features)

6. For each pixel that meets the criteria in 5, compute a feature descriptor.

****

* ***SIFT******(Scale-Invariant Feature Transform):***

1. Scale-space Extrema Detection (Feature point (also called key point) detection)
   * Potential location for finding features.
2. Feature point localization
   * Accurately locating the feature key points.
3. Orientation assignment
   * Assigning orientation to key points.
4. Feature descriptor generation.
   * Describing the key points as a high dimensional vector.
5. Keypoint Matching



* ***SURF (Speeded-Up Robust Features):***

1. Feature Extraction
   * Integral Images
   * Hessian matrix-based interest points
   * Scale-space representation
2. Feature Description
   * Orientation Assignment
   * Extract Descriptor Components

* ***Multiscale Oriented Patches Descriptor (MOPS):***

|  |  |  |
| --- | --- | --- |
| **Multiscale Oriented Patches Descriptor (MOPS)** | ***Translation*** (***T = MT1)*** |  |
| ***Rotation*** (***T = MRMT1)*** |  |
| ***Affine Transformation*** |  |
| ***Perspective Transformation*** |  |
| ***Scaling*** |  |

#### **After Feature Detection and Tracking**:



**8. Scene:**

|  |  |  |
| --- | --- | --- |
| **Scene** | **Feature to be detected** | **Sample Image** |
| **Image of some cyclists on the road** | Faces (Pedestrian or Cyclists) |  |
| **Image of a road with lanes visible** | Lanes |  |
| **Image of a road with lanes and rarely visible cars** | Cars |  |

**9. List of objects in the scene and the features:**

|  |  |
| --- | --- |
| **Object Name** | **List of features for the object** |
| Lanes | Shape,  Colour,  Texture |
| Vehicles | Shape,  Colour,  Size |
| Pedestrians/Cyclists | Shape,  Colour,  Size |
| Signals and Signboards | Shape,  Colour,  Size |

**10. Face Detection**

**Viola-Jones Algorithm:**







**Number of faces detected: 9**

**Number of falsely detected faces: 1**

**False Positive Rate : 8/9**

**Detection Rate :7/14**

**11. Face detection algorithms and Deep learning architecture:**

|  |  |  |
| --- | --- | --- |
| Face Detection Algorithm | URL | Deep learning architecture |
| Facenet | <https://machinelearningmastery.com/how-to-develop-a-face-recognition-system-using-facenet-in-keras-and-an-svm-classifier/#:~:text=FaceNet%20is%20a%20face%20recognition,of%20face%20recognition%20benchmark%20datasets.&text=About%20the%20FaceNet%20face%20recognition,implementations%20and%20pre%2Dtrained%20models>. | Facenet |
| Histogram of Oriented Gradients using Dlib | https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/ |  |
| Haar Cascade Classifiers using OpenCV | https://becominghuman.ai/face-detection-using-opencv-with-haar-cascade-classifiers-941dbb25177 |  |
| CNN based face detector from dlib | https://towardsdatascience.com/cnn-based-face-detector-from-dlib-c3696195e01c | dlib |

**12. Performance Metrics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Category** | **Purpose** | **Formula** |
| Peak Signal to Noise Ratio (PSNR) | Spatial Domain | Gives the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. | Where is the maximum possible pixel value of the image; is the Mean Squared Error;  A PSNR value of 30 dB above is preferred and ideally higher it is that much better. |
| Mean Squared Error (MSE) | Spatial Domain | Measures the average squared difference between the estimated values and the actual value. It is a risk function, corresponding to the expected value of the squared error loss. | Where is the Mean Squared Error; is the number of data points; are the observed values; are the predicted values;  There is no correct value. The closer it is to zero better and if it's zero it means the model is perfect. |
| Structural Similarity Index (SSIM) | Spatial Domain | It is a perceptual metric that quantifies image quality degradation\* caused by processing such as data compression or by losses in data transmission. | Where is the Structural Similarity Index;  is the average of ; is the average of ; is the variance of ; is the variance of ; is the covariance of and; and are two variables to stabilize the division with a weak denominator;  Generally, it is ideal to get 1 which means high structural similarity and 0 if no structural similarity |
| Intersection Over Union (IoU) | Spatial Domain | It is essentially a method to quantify the percent overlap between the target mask and our prediction output. Used as object detection evaluation metrics. | Ideally, **IoU** > 0.5 is considered a good prediction. |
| Accuracy |  | It is the ratio of correctly predicted observation to the total observations | An ideal value for it is closer to 100. |
| Precision |  | The ratio of correctly predicted positive observations to the total predicted positive observations is the precision. | An ideal value for it is closer to 100. |
| Recall |  | Recall is the ratio of correctly predicted positive observations to all observations in actual class | An ideal value for it is closer to 100. |
| F1-Score |  | The weighted average of Precision and Recall is the F1 Score. | An ideal value for it is closer to 100. |

**13. Deep Learning Architectures**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Architecture Name | Category | Learning | Year | Applications |
| CNN (Convolutional neural networks) |  | Supervised Learning | 2006 | Image recognition, video analysis, and natural language processing |
| RNN (Recurrent neural networks) |  | Supervised Learning | 1980 | Speech recognition and handwriting recognition |
| SOM (Self-organizing Maps) |  | Unsupervised Learning | 1980 | Dimensionality reduction, clustering high-dimensional inputs to 2-dimensional output, radiant grade result, and cluster visualization |
| AE(Autoencoders) |  | Unsupervised Learning | 1980 | Dimensionality reduction, data interpolation, and data compression/decompression |
| LSTM (Long Short-Term Memory) | RNN | Supervised Learning | 1997 | Image and video captioning systems |
| GRU (Gated Recurrent Unit) | RNN | Supervised Learning | 2014 | Natural language text compression, handwriting recognition, speech recognition, gesture recognition, image captioning |
| RBM (Restricted Boltzmann Machines) | AE | Unsupervised Learning | 1986 | Dimensionality reduction and collaborative filtering |
| DBN (Deep Belief Networks) |  | Supervised Learning | 2006 | Image recognition, information retrieval, natural language understanding, and failure prediction |
| DSN (Deep Stacking Networks) |  | Supervised Learning | 2011 | Information retrieval and continuous speech recognition |
| LeNet | Spatial Exploitation | Supervised Learning | 1998 | Recognizing simple digit images,  recognition of handwritten zip code digits |
| AlexNet | Spatial Exploitation | Supervised Learning | 2012 | allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU |
| VGGNet | Spatial Exploitation | Supervised Learning | 2014 | Architecture included in the Keras library used for large scale image recognition |
| GoogleNet | Spatial Exploitation | Supervised Learning | 2014 | It achieves efficiency through reduction of the input image, whilst simultaneously retaining important spatial information. It is designed to be a powerhouse with increased computational efficiency compared to some of its predecessors or similar networks created at the time.  image classification and object detection |
| ResNet | Depth + Multi-Path | Supervised Learning | 2015 | Extremely *deep* networks can be trained using standard SGD (and a reasonable initialization function) through the use of residual modules.  builds on constructs known from pyramidal cells in the cerebral cortex |
| ResNext | Width | Supervised Learning | 2016 | ResNeXts is the adding of parallel towers/branches/paths within each module |
| DenseNet | Multi-Path | Supervised Learning | 2017 | Used to keep increasing the depth of deep convolutional networks.  ***require fewer parameters*** than an equivalent traditional CNN, as there is ***no need to learn redundant feature maps***.  ***DenseNets layers are very narrow*** (e.g. 12 filters), and they just ***add a small set of new feature maps***. |
| PolyNet | Width | Supervised Learning | 2017 | A Very **Deep PolyNet** is composed based on the module. Compared to Inception-ResNet-v2, **PolyNet** reduces the Top-5 validation error on single crops from 4.9% to 4.25%, and that on multi-crops from 3.7% to 3.45%. **PolyNet**, By using PolyInception module, better than Inception-ResNet-v2 |
| PyramidalNet | Width | Supervised Learning | 2017 | Used in Spectral-Spatial Hyperspectral Image Classification |
| YOLO (You Only Look Once) | Spatial Exploitation | Supervised Learning | 2015 | Object detection can help with image classification |
| SqueezeNet | Spatial Exploitation | Supervised Learning |  | It can obtain AlexNet-level accuracy (~57% rank-1 and ~80% rank-5) at only 4.9MB through the usage of “fire” modules that “squeeze” and “expand”. |
| SegNet | Spatial Exploitation | Supervised Learning |  | Autonomous driving, scene understanding |
| GAN (Generative Adversarial Network) | Spatial Exploitation | Supervised Learning |  | Generate Examples for Image Datasets. Generate Photographs of Human Faces. Generate Realistic Photographs. Generate Cartoon Characters |

**14. Training, Testing, and Validation**

* Training data is used to help your machine learning model make predictions. It’s the largest part of your dataset, forming at least 70-80% of the total data you’ll use to build your model.
* Validation data is primarily used to determine whether your model can correctly identify new data or if it’s overfitting to your original dataset.
* Testing data is used after both training and validation. It aims to test the accuracy of your final model against your targets.

**Rule of 10:**

It is a common rule of thumb that we need more than 10 times the data for the model than its degree of freedom. The degree of freedom can be anything it can be an attribute, a parameter, or a column in our data that affects the output of the model. The main aim is to compensate for most of the variability that our parameters may bring into the input of the model.

**15. Face Detection**

**Training Dataset Description:**

* An array of tuples. The first element of each tuple is a NumPy array representing the image. The second element is its classification (1 for face, 0 for non-face)
* 2429 face images, 4548 non-face images

**Testing Dataset Description:**

* An array of tuples. The first element of each tuple is a NumPy array representing the image. The second element is its classification (1 for face, 0 for non-face)
* 472 faces, 23573 non-face images

**Validation Dataset Description:**

**Search for “****Face detection with viola jones and deep learning**”, run the code, and present the results for your dataset. Understand the code related to face detection.

**16. Optical Flow**

* [**Sparse Optical Flow: Lucas-Kanade method**](https://docs.opencv.org/master/d4/d8b/group__datasets__ar.html)

All the neighboring pixels will have similar motions. Lucas-Kanade method takes a 3x3 patch around the point. So all the 9 points have the same motion. We can find () for these 9 points. So now our problem becomes solving 9 equations with two unknown variables which are over-determined. A better solution is obtained with the least square fit method. Below is the final solution which is two equation-two unknown problems and solves to get the solution.

****

* [**Horn–Schunck method**](https://github.com/lmiz100/Optical-flow-Horn-Schunck-method)

The Horn–Schunck method of estimating optical flow is a global method that introduces a global constraint of smoothness to solve the aperture problem. It assumes smoothness in the flow over the whole image. Thus, it tries to minimize distortions inflow and prefers solutions that show more smoothness.

|  |  |
| --- | --- |
|  |  |
| **C:\Users\Jiss\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\EF0E7D72.tmp** | |

* **[Dense Optical Flow](https://docs.opencv.org/master/d4/d8b/group__datasets__ar.html)**

Dense Optical Flow computes the optical flow for all the points in the frame. It is based on Gunner Farneback's algorithm which is explained in "Two-Frame Motion Estimation Based on Polynomial Expansion" by Gunner Farneback.



**17. Acton Recognition**

|  |  |  |
| --- | --- | --- |
| **Scene** | **Inference** | **Required Action** |
| A person walking through the side of the road/lane | Probability of him/her continue walking or cross the road | Slow down or stop the vehicle |
| The vehicle in the front is slowing down | Probability of heavy traffic in front or it is going to be parked or maybe turn the vehicle around | Slow down or stop the vehicle |
| Traffic signal turning to red/green |  | Stop/Proceed |
| Traffic signal turning to yellow |  | Slow down the vehicle |
| One vehicle hitting another vehicle | Probability of occurring an Accident | Verify manually and pass the message to Hospital and Police Station if required |
|  |  |  |
|  |  |  |

**18. Object Recognition: YOLO (You Only Look Once)**

YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region.

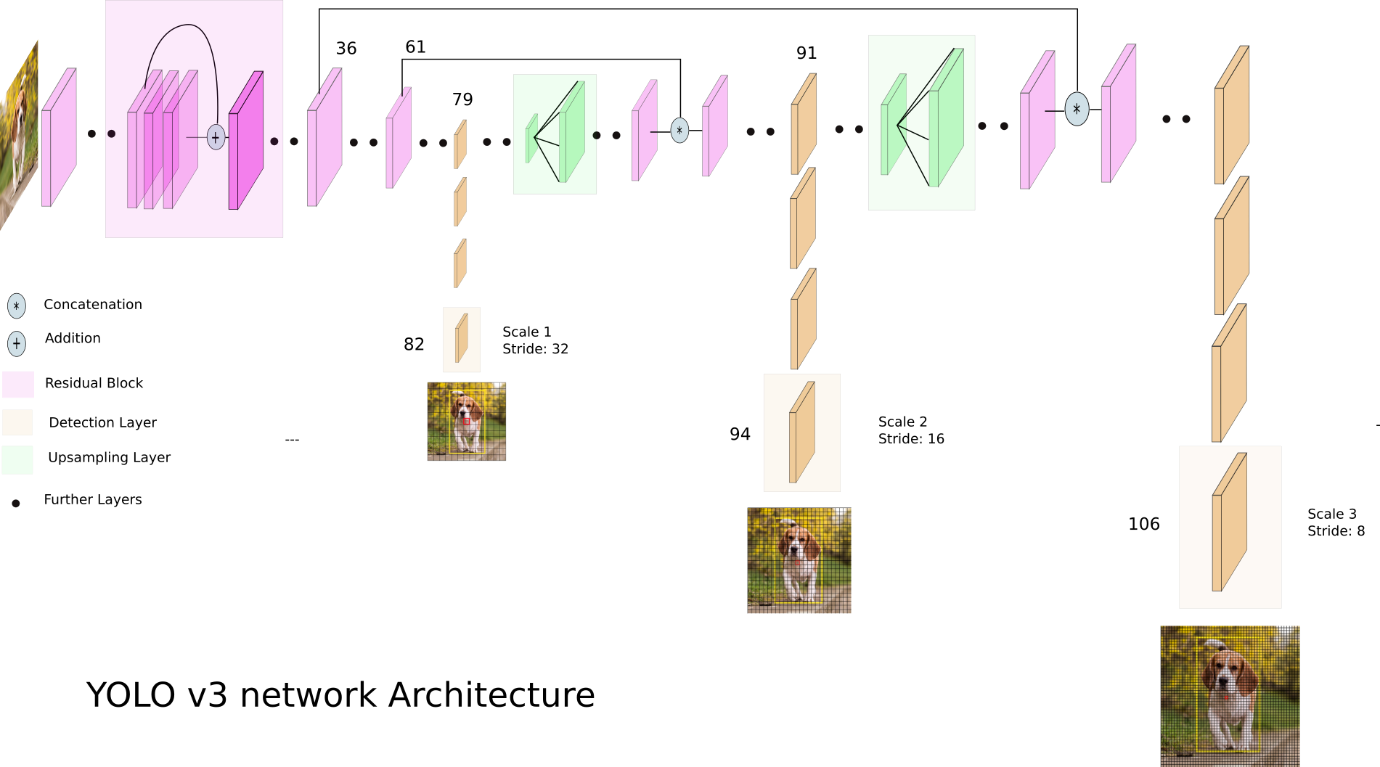
The algorithm “only looks once” at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.

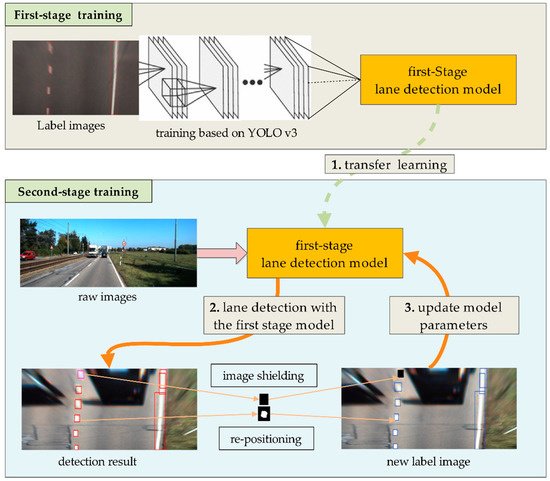
With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance.

YOLO learns generalizable representations of objects so that when trained on natural images and tested on the artwork, the algorithm outperforms other top detection methods.

YOLOv3 uses a variant of Darknet, a framework to train neural networks, which originally has 53 layers. For the detection task, another 53 layers are stacked onto it, accumulating to a total of a 106-layer fully convolutional architecture. This explains the reduction in speed in comparison with the second version, which only has 30 layers.

* **YOLOv3 Archiitecture Diagram:**



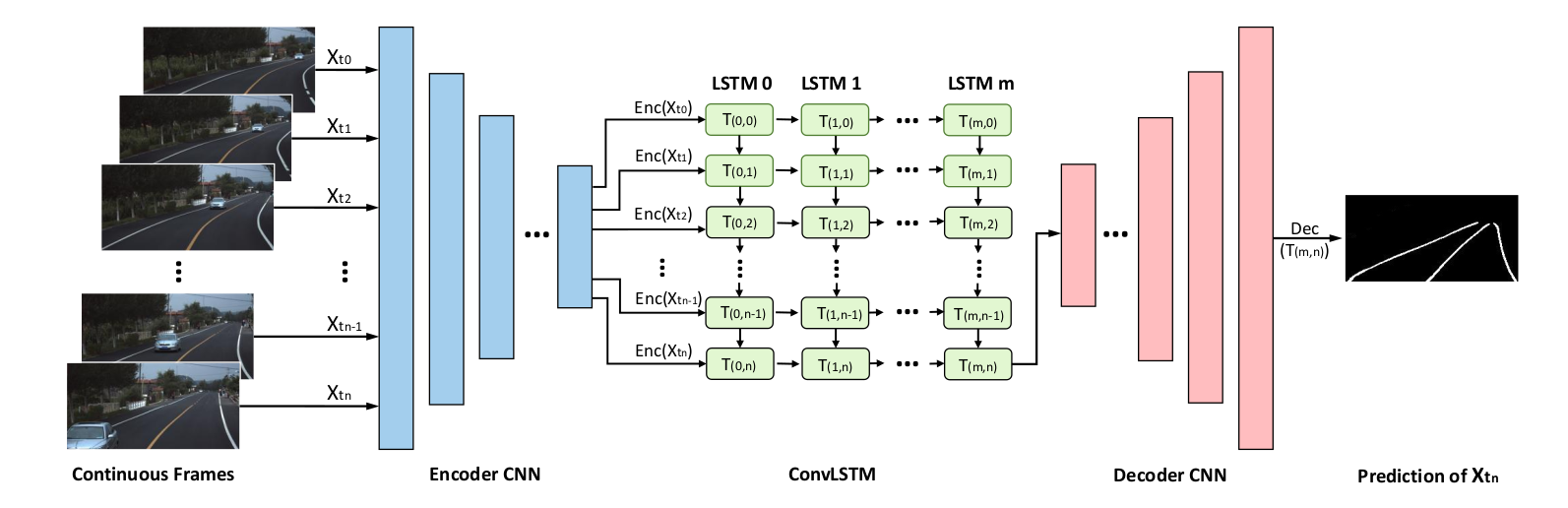
[](https://www.mdpi.com/sensors/sensors-18-04308/article_deploy/html/images/sensors-18-04308-g001-550.jpg)

YOLOv3 makes detections at three different scales. The detection is done by applying 1 x 1 detection kernels on feature maps of three different sizes at three different places in the network. The shape of the detection kernel is 1 x 1 x (B x (5 + C) ). Here B is the number of bounding boxes a cell on the feature map can predict, “5” is for the 4 bounding box attributes and one object confidence, and C is the number of classes. In YOLO v3 trained on COCO, B = 3 and C = 80, so the kernel size is 1 x 1 x 255. The feature map produced by this kernel has identical height and width to the previous feature map and has detection attributes along with the depth as described above.

|  |  |
| --- | --- |
| **Sample frame from the input video** | **Sample frame from the output video** |
|  |  |
|  |  |
|  |  |
|  |  |

**19. Deep Learning Architecture: LSTM (Long Short-Term Memory)**

Long Short-Term Memory (LSTM)s has the property of selectively remembering patterns for long durations of time.LSTM networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. It deals with the vanishing gradient problem encountered by traditional RNNs.

* **LSTM Archiitecture Diagram:**[](https://raw.githubusercontent.com/qinnzou/Robust-Lane-Detection/master/LaneDetectionCode/save/result/network.png)

|  |  |
| --- | --- |
| **Parameter Name**  **Purpose**  **A value that is generally provided** | **Hyper Parameter Name**  **Purpose**  **A value that is generally provided** |
|  |  |
|  |  |
|  |  |
|  |  |

**References**

1. <https://github.com/overtunned/lane_detection/tree/main/Dataset>
2. [GitHub - rslim087a/road-video: Video required for finding lane lines](https://github.com/rslim087a/road-video)
3. [CULane dataset](https://xingangpan.github.io/projects/CULane.html):

https://drive.google.com/drive/folders/1mSLgwVTiaUMAb4AVOWwlCD5JcWdrwpvu

1. TUsimple dataset: https://github.com/TuSimple/tusimple-benchmark/issues/3
2. [Real-time detection of road lane-lines for autonomous driving](https://www.researchgate.net/publication/331478663_Real-Time_Detection_of_Road_Lane-Lines_for_Autonomous_Driving)
3. [Multi-Lane Detection and Tracking Using Vision for Traffic Situation Awareness](https://ieeexplore.ieee.org/document/9253415)
4. [Real-Time Tracking and Lane Line Detection Technique for an Autonomous Ground Vehicle System](https://link.springer.com/chapter/10.1007/978-981-15-0633-8_156#:~:text=Vehicle%20tracking%20has%20been%20performed,using%20adaptive%20background%20subtraction%20technique.)
5. https://tryolabs.com/resources/introductory-guide-computer-vision