Table of Contents

1- reading the data in the file...

```
clc
clear all;
close all;
f = fopen("Test_Text_File.txt");
data = fileread("Test_Text_File.txt");
fclose(f);
fprintf('%s\n','The message is :')
fprintf('%s',data)
keyset= unique(data); %% find all characters in the file ...
```

```
The message is :
This work proposes integrating traditional computer vision techniques
 and deep learning methods to develop a reliable benchmarking
 framework for lane detection tasks in complex and dynamic road
 scenes. Firstly, an automatic segmentation algorithm based on
 a sequence of traditional computer vision techniques has been
 experimented. This algorithm precisely segments the semantic region
 of the host lane in the complex urban images of nuScenes dataset used
 in this framework; hence corresponding weak labels are generated.
 After that, the developed data is qualitatively evaluated to be
 used in training and benchmarking five state-of-the-art FCN-based
 architectures: SegNet, Modified SegNet, U-Net, ResUNet, and ResUNet+
+. The performance evaluation of the trained models is done visually
 and quantitatively by considering lane detection a binary semantic
 segmentation task. The output results show robust performance,
 especially ResUNet++, which outperforms all the other models while
 testing them in different complex road scenes with dynamic scenarios
 and various lighting conditions.
```

2- calculate the entropy....

```
P=[];
entrop=0;
```

```
for i = 1:length(keyset)
P=[P count(data,keyset(i))/length(data)];
e=P(i)*log2(1/P(i)); % calculating entropy of this character...
entrop = entrop+e;% add entropy of this character to the previous
 characters...
end
disp('The entropy is')
disp(entrop);
disp('bits/symbol')
```

```
The entropy is
    4.3831

bits/symbol
```

3- Number of bits for construction of a fixed length code

```
len=ceil(log2(length(keyset)));
disp(len)
disp('bits/symbol')
disp('Efficiency for this construction:')
disp((entrop/len)*100)
```

```
     6

bits/symbol
Efficiency for this construction:
   73.0508
```

4- Huffman Encoder...

```
[codeword,indices,P] = Huffencoder(P,keyset);
```

```
Table for each character and its codeword
Keyset     codeword
           001
+          01001011
,          1000101
-          00001010
.          00000001
:          110101011
;          0100101000
A          0100101001
C          0100101010
F          110101010
M          0100101011
N          1101011
R          000000000
S          000000001
```

```
T          10001000
U          10001001
a          0101
b          0000001
c          10010
d          01000
e          101
f          110100
g          000011
h          10011
i          0110
k          1000110
l          10000
m          11011
n          0111
o          1110
p          010011
q          00001011
r          1111
s          1100
t          0001
u          000001
v          0000100
w          1000111
x          11010100
y          0100100
```

5- calculating L:(average codeword length)'

```
L_codeword = 0;
for i = 1:length(P)
  L=P(i)*length(cell2mat(codeword(indices(i))));
  L_codeword= L_codeword + L;
end
disp('L average =')
disp(L_codeword)
disp('bits/symbol')
```

```
L average =
    4.3983

bits/symbol
```

6- Efficiency

```
efficiency = (entrop/L_codeword)*100;
disp('Efficiency =')
disp(efficiency)
```

```
Efficiency =
   99.6523
```

7- Encode the whole text in the file

```
seq=[];
for i =1:length(data)
    k=find(keyset==data(i)); %% find the index in keyset where it and
 the data(i)((first letter in text for example)) are equal....
    seq=[seq cell2mat(codeword(k))]; %% get the codeword for this
 letter which its index is k...
    %this sequance will be decoded...
end
fprintf('%d',seq);
fileID = fopen('encodedseq.txt','w');
fprintf(fileID,'%-10s\r\n','The Encoded sequence');
fprintf(fileID,'%d',seq);
fclose(fileID);
```

```
1000100010011011011000011000111111011111000110001010011111111100100111110110010111
```

8- Decoding with AWGN, No channel coding

```
disp('Decoding without channel coding and No AWGN:')
out = Huffdecoder(codeword,seq,keyset,'decoded_message.txt');


disp('Decoding without channel coding AWGN SNR = 2:')
seq1=AddingNoise(seq,2);
out1 = Huffdecoder(codeword,seq1,keyset,'Noisy_decoded_message1.txt');
disp('Number of errors is:')
[number,ratio] = biterr(seq1,seq);
disp(number)

disp('Decoding  without channel coding AWGN SNR = 7:')
seq2=AddingNoise(seq,7);
out2 = Huffdecoder(codeword,seq2,keyset,'Noisy_decoded_message2.txt');
disp('Number of errors is:')
[number,ratio] = biterr(seq2,seq);
disp(number)

disp('Decoding without channel coding AWGN SNR = 15:')
seq3=AddingNoise(seq,15);
out3 = Huffdecoder(codeword,seq3,keyset,'Noisy_decoded_message3.txt');
disp('Number of errors is:')
[number,ratio] = biterr(seq3,seq);
disp(number)
```

```
Decoding without channel coding and No AWGN:
The decoded seq is
This work proposes integrating traditional computer vision techniques
 and deep learning methods to develop a reliable benchmarking
 framework for lane detection tasks in complex and dynamic road
 scenes. Firstly, an automatic segmentation algorithm based on
 a sequence of traditional computer vision techniques has been
```

experimented. This algorithm precisely segments the semantic region of the host lane in the complex urban images of nuScenes dataset used in this framework; hence corresponding weak labels are generated. After that, the developed data is qualitatively evaluated to be used in training and benchmarking five state-of-the-art FCN-based architectures: SegNet, Modified SegNet, U-Net, ResUNet, and ResUNet++. The performance evaluation of the trained models is done visually and quantitatively by considering lane detection a binary semantic segmentation task. The output results show robust performance, especially ResUNet++, which outperforms all the other models while testing them in different complex road scenes with dynamic scenarios and various lighting conditions.

Decoding without channel coding AWGN SNR = 2:
The decoded seq is

```
T, ldrsrt, Fan eo: rs  rhtonkpraus nNesucote sieeccdessoadaslrm
 kpn-end tidkuariieainn  hlslps dec  phiapNdses lllnnenholog
 xnhphoontoredunrssttssdcdchm oiao teln p nns s  oeqosill Neecgr
+dmsotheugeumrtdn  thnaqv mpqa-ag:r atpheeottoi lmisl-heurn a se
+tiroUodthooulethkuceifsd hdpbtsmcelteerfecdl  sot.oesesronireeNkett
 cet eTeileorrdhact ostr nschtbioesrucipcofn cstre omhfee-ooh
 oovstnsmhssnrrrnnfsonayti taecoe yssnmdtoattoeaooaC ayteuyd in e,rr
 lrameworeiMtnq fsesnrntroidin io sei Ttxseag neege NeaoeRse a d er
 tmat, -fetdmdn-ed inedtrsitivngsseaTmoyeeval  FfthsngyeoCthlti
 d sece og eenlheNrc hointitche eue N R g r aget afhahsed-otdeioh
 m nlctiAed ltotro ttoa srim ueauSscNeesr U-Neh e Reeiiiret-a
 oettun tmtsc ermbtohteed,omeeCrea  nlft-irmierdux todsc tdoehnte
 nmatsraoiim,igt teFgle nasaleaab igbbuv soeiFopu l+s ifoTaesoa
 fepmieoteamartsctcureasadeh sbi tolh vser- ptsdrT ilatsgiwt obgo
 eoi ronnortoaiveidomtii u Rdevgkodaeio ptonseprde,emosniir neiUthsnw
 msans,gwteee gctnuinTwniinkrdiao no sss,eeileodnh fnsconaRis
 tclrpamim s.rnees lehivi amigsests adetot coadotio og
```

Number of errors is:
859

Decoding without channel coding AWGN SNR = 7:
The decoded seq is

```
,imiSrerahprrposes integrating toadntirnal comssctnslwsiorbechniblio
 eafaaaeisf arninu methods tsedeveloisedrebcl sa be se iNrth
 fsohNemtemo wpeoi-r+iksl esodenbelehh nis lardnaf evwtmih
 roadicdeoo.hentne  ete tCrayweeedee tsepsco atisnielgorithm
 ba+cnhcoi i igaoaprl tradit neNbcomput r visio c  ee emuigng
 envgaesmedtmomnasiedu This algorithm precimivliagme wvinpcoF,e
 tregion eilk e hol  lane inethe complexelrl an images of  lRiaaro
 e- aoet gse.in thoste oermhrnsiylwaiaisoerrnt n f hlssn ak labels
 prxoeireuesdtve rtincehntlvinyuvelopedileta is qbelitttivcvlf
 cs.a-,ts beSsvtin training aid benchmarcthlsokve state-of;inda
 ixoeyemaleR dfttnctehm vtres:R ogendsa Coa eetef lu+Netlsspdeendsa
 ResUNet, destutn t Neccfr.eThevorepereeaoxf ulledeesiaee-soew:eoe
  Nfae leeletsry teltally anitvfeasnlr,et ivll yt  ho e,eehlk-
r;etection aiuonmoctsemantic oegsNhatoon aasktt Ttfrbtistt
 ehctlts d rpsrr.ust performNfe, esM  cs u Rua t Net++w whnch
 nutdmrerereoaalleteohs her mouitwte ee a qlaestc hem in different
 ceifot oe orad oceneNwivsevwpifw  anaried an.vniirps lighting
 Nmoveiono.
```

Number of errors is:
272

Decoding without channel coding AWGN SNR = 15:
The decoded seq is
This work proposes integrating traditional computer vision techniques and deep learning methods to develop a reliable benchmarking framework for lane detection tasks in complex and dynamic road scenes. Firstly, an automatic segmentation algorithm based on a sequence of traditional computer vision techniques has been experimented. This algorithm precisely segments the semantic region of the host lane in the complex urban images of nuScenes dataset used in this framework; hence corresponding weak labels are generated. After that, the developed data is qualitatively evaluated to be used in training and benchmarking five state-of-the-art FCN-based architectures: SegNet, Modified SegNet, U-Net, ResUNet, and ResUNet++. The performance evaluation of the trained models is done visually and quantitatively by considering lane detection a binary semantic segmentation task. The output results show robust performance, especially ResUNet++, which outperforms all the other models while testing them in different complex road scenes with dynamic scenarios and various lighting conditions.
Number of errors is:
0

Using Convolutional Encoder

```
disp('Decoding using channel coding')
stream=Convolutional_Encoder(seq);

disp('Decoding with AWGN SNR = 2:')
stream1=AddingNoise(stream,2);
stream1=Viterbi_Decoder(stream1);
out1 = Huffdecoder(codeword,stream1,keyset,'Noisy_decoded_Conv1.txt');
disp('Number of errors is:')
[number,ratio] = biterr(stream1,seq);
disp(number)

disp('Decoding with AWGN SNR = 7:')
stream2=AddingNoise(stream,7);
stream2=Viterbi_Decoder(stream2);
out2 = Huffdecoder(codeword,stream2,keyset,'Noisy_decoded_Conv2.txt');
disp('Number of errors is:')
[number,ratio] = biterr(stream2,seq);
disp(number)


disp('Decoding with AWGN SNR = 15:')
stream3=AddingNoise(stream,15);
stream3=Viterbi_Decoder(stream3);
out3 = Huffdecoder(codeword,stream3,keyset,'Noisy_decoded_Conv3.txt');
disp('Number of errors is:')
```

```
[number,ratio] = biterr(stream3,seq);
disp(number)
```

```
Decoding using channel coding
Decoding with AWGN SNR = 2:
The decoded seq is
ld is wwsi penetises ifilceddf ccnoubesoel computerl tekovtecl
 lques anisdeepem hoopk methmx tetud+uciaps ucl le benchmark  s he
 oemhoeno etpeoiaene udm arn s,n tee in comod ntbaftdynamvwrceum
 anes. Fird ldneCoad  etinkotsegtsrtation algo  lks baslueesedetq
 encsneeviFtceesoel compfter Sfion tecp iques has blerTu srimen-lb
 vpimcsbonekrm itoshialy segrants u in nienhe.o gir hmRiipuelnstaocse
 dhp nisiunaqls-efteednngmdasRtcenes prdr ooln auin this a a
 mewessi; hence otnnslhonding weyiReicNleioctforedxS After that,
 the detsthcnaudata it+iSlitatively evaluate.to beigcettkurai bs
 cend benchferking rtlAstd .amdtdlole ile-caam TSsed arcr lstvtres:
 u egNetebModsfiedRutaafi pt tm adudRd t Net-nrftResUNet+ vutat
 fe mre mrih -il,lntdeh srn.  s trained msamgqvetsnCteltally tnd
 qenro evd l+u vgree n- l nnas cleie deteciion-dtdoery seman ee
 tfuooesadevl,nnaS ,ge outputeresults showrrsuustvoroe e dgsei
 pnbacialea v.ar  Netadtntpv  e gtsva mre mrnle vvsnps her modecdi
 oit-dmlmtuct mecsoigih+rasp nifd nagrseieitseootllviatatNeee tslen
 reemu nd Snmnbligussahlssesols aans.
Number of errors is:
   511

Decoding with AWGN SNR = 7:
The decoded seq is
This work proposes integrating tradiuesoel computer vision techniques
 and deep learning methods to develop a reliable benchmarking
 framework for lane detection tasks in complex and dynamic road
 scenes. Firstly, an automatic segmentation algorithm based on
 a sequence of traditional computer vision techniques has been
 experimented. This algorithm pr cisely segments the semantic region
 of the host lane in the complex urban images of nuScenes dataset used
 in this framework; hence tnnrmUr f hlssneak labels are generated.
 After that, the developed data is qualitatively evaluated to be
 used in training and benchmarlthlsokve state-of-the-art F-eeitd-oad
 architectures: SegNet, Modified SegNea, U-Net, ResUNet, and ResUNet
++. The percnno:sepddeluation of the trained models is done visually
 and quantitatively by considering lane detection a binary semantic
 segFo ation task. The output results show robust performance,
 especially ResUco, ico ptoife hsva mre mrnleitbthe other models while
 testing them in different cose s nagrsed scenes wite;ynamic scenarios
 and various lighting conditions.
Number of errors is:
    13

Decoding with AWGN SNR = 15:
The decoded seq is
This work proposes integrating traditional computer vision techniques
 and deep learning methods to develop a reliable benchmarking
 framework for lane detection tasks in complex and dynamic road
 scenes. Firstly, an automatic segmentation algorithm based on
```

```
 a sequence of traditional computer vision techniques has been
 experimented. This algorithm precisely segments the semantic region
 of the host lane in the complex urban images of nuScenes dataset used
 in this framework; hence corresponding weak labels are generated.
 After that, the developed data is qualitatively evaluated to be
 used in training and benchmarking five state-of-the-art FCN-based
 architectures: SegNet, Modified SegNet, U-Net, ResUNet, and ResUNet+
+. The performance evaluation of the trained models is done visually
 and quantitatively by considering lane detection a binary semantic
 segmentation task. The output results show robust performance,
 especially ResUNet++, which outperforms all the other models while
 testing them in different complex road scenes with dynamic scenarios
 and various lighting conditions.
Number of errors is:
     0
```

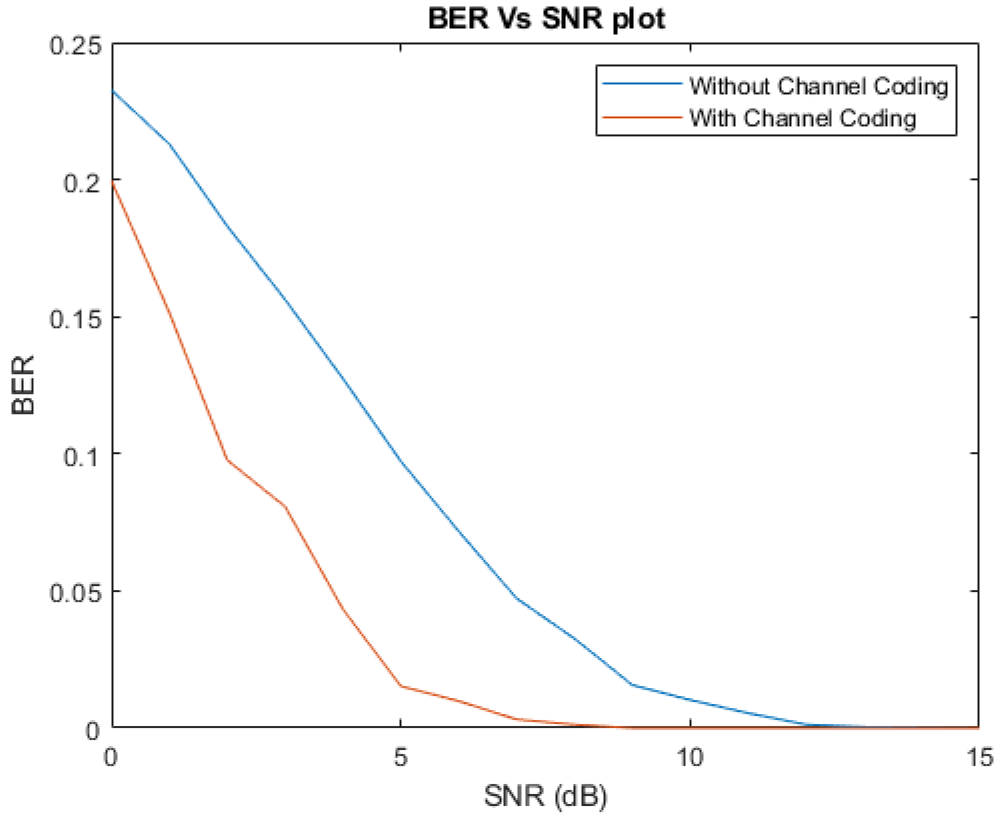
Plotting BER VS SNR

```
BER1=[];
BER2=[];
snr=0:1:15;
for i = 1:length(snr)

    seq1=AddingNoise(seq,snr(i));
    [number,ratio] = biterr(seq1,seq);
    BER1=[BER1 ratio];

    stream1=AddingNoise(stream,snr(i));
    stream1=Viterbi_Decoder(stream1);
    [number,ratio] = biterr(stream1,seq);
    BER2=[BER2 ratio];
end

figure
plot(snr,BER1)
hold on
plot(snr,BER2)
xlabel('SNR (dB)');
ylabel('BER');
legend('Without Channel Coding','With Channel Coding')
title('BER Vs SNR plot');
```



9- Checking if the decoded message is the same as original message (phase 1)

```
if(isequal(data,out))
    disp('They are exactly the same')
else
    disp('They are not the same')
end
```

```
They are exactly the same
```

Published with MATLAB® R2021a