Fraud Detection In financial service using cnn-au421221243023

[ ]:
```
#!pip install tensorflow-gpu==2.0.0-rc0
# Keep your eyes for update: https://www.tensorflow.org/
```

[27]:
```
import tensorflow as tf from
tensorflow import keras from
tensorflow.keras import
Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout,
BatchNormalization from tensorflow.keras.layers import Conv1D,
MaxPool1D from tensorflow.keras.optimizers import Adam
print(tf.__version__)
```

```
2.0.0-rc0
```

[2]:
```
import pandas as pd
import numpy as np
import  seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

[3]:
```
data = pd.read_csv('creditcard.csv')
data.head()
```

[3]:
```
   Time        V1        V2        V3        V4        V5        V6        V7 \
 0     0.0 -1.359807 -0.072781 2.536347 1.378155 -0.338321 0.462388
   0.239599 1  0.0 1.191857 0.266151 0.166480 0.448154 0.060018 -
 0.082361 -0.078803 2   1.0 -1.358354 -1.340163 1.773209 0.379780 -
0.503198 1.800499 0.791461 3      1.0 -0.966272 -0.185226 1.792993 -
              0.863291 -0.010309 1.247203 0.237609
42.0 -1.158233 0.877737 1.548718 0.403034 -0.407193 0.095921 0.592941

        V8        V9 …       V21       V22       V23       V24       V25 \
0  0.098698 0.363787 … -0.018307 0.277838 -0.110474 0.066928 0.128539
1  0.085102 -0.255425 … -0.225775 -0.638672 0.101288 -0.339846
   0.167170
2  0.247676 -1.514654 … 0.247998 0.771679 0.909412 -0.689281 -0.327642
3  0.377436 -1.387024 … -0.108300 0.005274 -0.190321 -1.175575
   0.647376
4  -0.270533 0.817739 … -0.009431 0.798278 -0.137458 0.141267 -
   0.206010

       V26       V27       V28  Amount  Class
0 -0.189115 0.133558 -0.021053 149.62  0
```

```
1 0.125895 -0.008983 0.014724 2.69     0
2 -0.139097 -0.055353 -0.059752 378.66 0
3 -0.221929 0.062723 0.061458 123.50   0
4 0.502292 0.219422 0.215153 69.99     0
[5 rows x 31 columns]
```

[4]:
```
data.shape
```

[4]: (284807, 31)

[5]:
```
data.isnull().sum()
```

```
[5]: Time     0
V1       0
V2       0
V3       0
V4       0
V5       0
V6       0
V7       0
V8       0
V9       0
V10      0
V11      0
V12      0
V13      0
V14      0
V15      0
V16      0
V17      0
V18      0
V19      0
V20      0
V21      0
V22      0
V23      0
V24      0
V25      0
V26      0
V27      0
V28      0
Amount 0 Class
0
dtype: int64
```

[6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to
```

```
284806 Data columns (total 31
columns):
Time     284807      non-null
         float64
V1       284807      non-null
         float64
V2       284807      non-null
         float64
V3       284807      non-null
         float64
V4       284807      non-null
         float64
V5       284807      non-null
         float64
V6       284807      non-null
         float64
V7       284807      non-null
         float64
V8       284807      non-null
         float64
V9       284807      non-null
         float64
V10      284807      non-null
         float64
V11      284807      non-null
         float64
V12      284807      non-null
         float64
V13      284807      non-null
         float64
V14      284807      non-null
         float64
V15      284807      non-null
         float64
V16      284807      non-null
         float64
V17      284807      non-null
         float64
V18      284807      non-null
         float64
V19      284807      non-null
         float64
V20      284807      non-null
         float64
V21      284807      non-null
         float64
V22      284807      non-null
         float64
```

```
V23       284807      non-null
          float64
V24       284807      non-null
          float64
V25       284807      non-null
          float64
V26       284807      non-null
          float64
V27       284807      non-null
          float64
V28       284807      non-null
          float64
Amount    284807      non-null
          float64
Class     284807 non-null
          int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

[7]:
```
data['Class'].value_counts()
```

[7]:
```
0    284315
1       492
Name: Class, dtype: int64
```

0.1.1 Balance Dataset

[8]:
```
non_fraud = data[data['Class']==0]
fraud = data[data['Class']==1]
```

[9]:
```
non_fraud.shape, fraud.shape
```

[9]:
```
((284315, 31), (492, 31))
```

[10]:
```
non_fraud = non_fraud.sample(fraud.shape[0])
non_fraud.shape
```

[10]:
```
(492, 31)
```

[11]:
```
data = fraud.append(non_fraud, ignore_index=True)
data
```

[11]:
```
         Time        V1        V2         V3        V4        V5        V6 \
0       406.0 -2.312227 1.951992 -1.609851 3.997906 -0.522188 -
        1.426545
1       472.0 -3.043541 -3.157307  1.088463 2.288644 1.359805 -
        1.064823 24462.0 -2.303350 1.759247 -0.359745 2.330243 -
        0.821628 -0.075788
```

```
3      6986.0 -4.397974 1.358367 -2.592844 2.679787 -1.128131 -1.706536
4      7519.0 1.234235 3.019740 -4.304597 4.732795 3.624201 -1.357746
5      7526.0 0.008430 4.137837 -6.240697 6.675732 0.768307 -3.353060
6      7535.0 0.026779 4.132464 -6.560600 6.348557 1.329666 -2.513479
7      7543.0 0.329594 3.712889 -5.775935 6.078266 1.667359 -2.420168
8      7551.0 0.316459 3.809076 -5.615159 6.047445 1.554026 -2.651353
9      7610.0 0.725646 2.300894 -5.329976 4.007683 -1.730411 -1.732193
10     7672.0 0.702710 2.426433 -5.234513 4.416661 -2.170806 -2.667554
11     7740.0 1.023874 2.001485 -4.769752 3.819195 -1.271754 -1.734662
12     7891.0 -1.585505 3.261585 -4.137422 2.357096 -1.405043 -1.879437
13     8090.0 -1.783229 3.402794 -3.822742 2.625368 -1.976415 -2.731689
14     8169.0 0.857321 4.093912 -7.423894 7.380245 0.973366 -2.730762
15     8408.0 -1.813280 4.917851 -5.926130 5.701500 1.204393 -3.035138
16     8415.0 -0.251471 4.313523 -6.891438 6.796797 0.616297 -2.966327
17     8451.0 0.314597 2.660670 -5.920037 4.522500 -2.315027 -2.278352
18     8528.0 0.447396 2.481954 -5.660814 4.455923 -2.443780 -2.185040
19     8614.0 -2.169929 3.639654 -4.508498 2.730668 -2.122693 -2.341017
20     8757.0 -1.863756 3.442644 -4.468260 2.805336 -2.118412 -2.332285
21     8808.0 -4.617217 1.695694 -3.114372 4.328199 -1.873257 -0.989908
22     8878.0 -2.661802 5.856393 -7.653616 6.379742 -0.060712 -3.131550
23     8886.0 -2.535852 5.793644 -7.618463 6.395830 -0.065210 -3.136372
24     9064.0 -3.499108 0.258555 -4.489558 4.853894 -6.974522 3.628382
25     11080.0 -2.125490 5.973556 -11.034727 9.007147 -1.689451 -2.854415
26     11092.0 0.378275 3.914797 -5.726872 6.094141 1.698875 -2.807314
27     11131.0 -1.426623 4.141986 -9.804103 6.666273 -4.749527 -2.073129
28     11629.0 -3.891192 7.098916 -11.426467 8.607557 -2.065706 -2.985288
29     11635.0 0.919137 4.199633 -7.535607 7.426940 1.118215 -2.886722
..        …        …        …        …        …        …        …
954    51632.0 -0.910542 -0.207061 -0.238652 -1.620610 1.068202 4.117210
955    40914.0 -0.915835 1.317547 0.993125 -0.236196 0.197397 -0.666025
956    157688.0 -0.497431 0.678159 -1.520938 -1.000571 -0.741602 -0.888050
```

```
957   139800.0 -0.927206 -0.725931     0.818936 -0.552991 0.558724 -
      0.535803
958   135483.0 -5.770397 -5.696525 -1.220788 0.991614 2.852955 -
      2.299697
959   49684.0 -1.065596 0.842987  0.172065 -0.436670 2.653078 3.806699
960   70867.0 -0.682535 1.058084  0.664150 -0.030638 -0.183924 -
      0.772025
961   125113.0 -0.550393 0.606198 1.732814 -0.477593 0.063260 -
      0.080612
962   57642.0 1.296055 0.307048 -0.340150 0.931280 0.572522 0.236900
963   67905.0 1.124166 -0.245039 -1.243372 0.323470 2.115415 3.632770
964   12443.0 1.105762 -0.616627  0.816607 0.339242 -0.900344 0.138514
965   45034.0 1.195644 -1.696748  0.772249 -1.386388 -2.002641 -
      0.146342
966   65436.0 -3.877934 2.831185 -0.682614 1.295636 -2.063089
      1.283378
967   146776.0 1.970169 -0.596364 -1.733929 -0.680361 1.818093
      3.778353
968   74968.0 0.969108 -1.810261 -0.070629 -1.054164 -1.634434 -
      0.743319
969   152001.0 -1.414994 2.236620 1.378835 4.262823 -0.478623 0.823817
970   56322.0 0.179097 1.945647 -3.804657 0.395820 3.220904 2.333843
971   157194.0 0.014710 1.232299 -0.961770 -0.669596 1.595960 -
      1.128452
972   38007.0 -0.596652 0.606731  2.035640 -1.216988 -0.220961 -
      0.898262
973   44791.0 -2.014166 2.057500  0.800515 -0.046729 0.237468 1.924049
974   79383.0 0.268050 0.012069   1.282745 -1.201270 -0.622377 -
      0.405405
975   63925.0 1.139142 -0.574897  0.176115 -0.812239 -0.809421 -
      0.568300
976   8449.0 1.192691 1.243546 -1.373662 1.799776 0.713990 -1.618146
977   162993.0 -0.562449 1.665333 -0.789924 -0.246244 1.866147 -
      0.912833
978   36002.0 1.318495 -0.229179  0.307091 0.254830 -0.444094 -
      0.059009
979   153538.0 0.134416 0.743800 -1.984022 -1.295774 3.151207
      3.155450
980   79156.0 0.886793 -0.890167  0.956626 0.388763 -1.369543 -
      0.334280
981   48642.0 -1.713619 1.357466 -0.138878 0.260421 0.880219 0.228354
982   89988.0 1.819294 -0.098211 -1.190861 2.760301 2.164190 4.772675
983   97710.0 1.559744 0.590840 -1.936930 1.116909 0.397621 -0.747927

          V7        V8        V9 …       V21       V22       V23 \
```

```
0     -2.537387 1.391657 -2.770089 … 0.517232 -0.035049 -
0.465211 1  0.325574 -0.067794 -0.270953 … 0.661696
0.435477 1.375966
2 0.562320 -0.399147 -0.238253 … -0.294166 -0.932391
0.172726 3 -3.496197 -0.248778 -0.247768 … 0.573574
0.176968 -0.436207
4    1.713445 -0.496358 -1.282858 … -0.379068 -0.704181 -0.656805
5    -1.631735 0.154612 -2.795892 … 0.364514 -0.608057 -0.539528
6    -1.689102 0.303253 -3.139409 … 0.370509 -0.576752 -0.669605
7    -0.812891 0.133080 -2.214311 … 0.156617 -0.652450 -0.551572
8    -0.746579 0.055586 -2.678679 … 0.208828 -0.511747 -0.583813
9    -3.968593 1.063728 -0.486097 … 0.589669 0.109541 0.601045
10   -3.878088 0.911337 -0.166199 … 0.551180 -0.009802 0.721698
11   -3.059245 0.889805 0.415382 … 0.343283 -0.054196 0.709654
12   -3.513687 1.515607 -1.207166 … 0.501543 -0.546869 -0.076584
13   -3.430559 1.413204 -0.776941 … 0.454032 -0.577526 0.045967
14   -1.496497 0.543015 -2.351190 … 0.375026 0.145400 0.240603
15   -1.713402 0.561257 -3.796354 … 0.615642 -0.406427 -0.737018
16   -2.436653 0.489328 -3.371639 … 0.536892 -0.546126 -0.605240
17   -4.684054 1.202270 -0.694696 … 0.743314 0.064038 0.677842
18   -4.716143 1.249803 -0.718326 … 0.756053 0.140168 0.665411
19   -4.235253 1.703538 -1.305279 … 0.645103 -0.503529 -0.000523
20   -4.261237 1.701682 -1.439396 … 0.667927 -0.516242 -0.012218
21   -4.577265 0.472216 0.472017 … 0.481830 0.146023 0.117039
22   -3.103570 1.778492 -3.831154 … 0.734775 -0.435901 -0.384766
23   -3.104557 1.823233 -3.878658 … 0.716720 -0.448060 -0.402407
24   5.431271 -1.946734 -0.775680 … -1.052368 0.204817 -2.119007
25   -7.810441 2.030870 -5.902828 … 1.646518 -0.278485 -0.664841
26   -0.591118 -0.123496 -2.530713 … 0.149896 -0.601967 -0.613724
27   -10.089931 2.791345 -3.249516 … 1.865679 0.407809 0.605809
28   -8.138589 2.973928 -6.272790 … 1.757085 -0.189709 -0.508629
29   -1.341036 0.363933 -2.203224 … 0.316094 0.055179 0.210692
..        …        …        … …        …        …        …
954  -0.141094 1.477371 0.350114 … 0.020076 -0.350732 0.552200
955  0.757835 -0.116348 0.023059 … -0.001425 0.379368 -0.239480
956  0.653594 0.429176 -2.047518 … -0.004812 0.276263 0.064681
957  0.086743 0.137233 0.323443 … -0.228540 -0.427330 1.049918
958  0.548609 0.190603 0.285034 … -0.567495 0.116072 5.485748
959  0.304196 0.812782 -0.368669 … -0.108110 -0.390159 -0.408153
960  0.625303 0.359726 -0.987842 … 0.176234 0.250496 -0.000062
961  0.392867 -0.053336 0.414040 … 0.329778 1.366743 -0.392367
962  0.201531 -0.020137 -0.161151 … -0.069141 -0.045908 -0.325746
963  -0.343524 0.830462 0.163974 … -0.139787 -0.518830 -0.190092
964  -0.719458 0.063580 2.698915 … -0.399295 -0.884769 -0.075285
965  -1.411495 0.114976 -1.577998 … -0.040671 -0.113978 -0.075130
966  -2.270468 3.243956 0.390027 … -0.289132 -0.746505 0.111810
```

```
967  -1.102810 1.027472 1.120832 … -0.181881 -0.489404 0.411581
968  -0.432448 -0.247401 -2.120424 … 0.013725 -0.070490 -0.302346
969  -0.247187 0.504294 -0.490021 … 0.104418 0.861802 -0.109472
970  0.195374 0.950539 -0.113631 … -0.269630 -0.504177 0.088089
971  1.593136 -0.342923 -0.832564 … 0.044372 0.104575 -0.330069
972  0.699627 -0.109285 0.237855 … -0.081429 -0.028667 -0.001075
973  -1.105882 -3.077766 -0.190878 … -0.813233 -0.423911 -0.088260
974  -0.237844 0.162505 1.565277 … 0.113336 0.589706 0.079297
975  -0.259045 0.085214 1.617206 … -0.212520 -0.587116 -0.081918
976  0.466555 -0.234287 0.828311 … -0.316004 -0.683985 -0.162753
977  1.595264 -0.138339 -1.304623 … -0.015247 -0.024051 -0.683411
978  -0.384156 -0.014430 -0.971022 … -0.595372 -1.235881 0.141314
979  0.410317 0.842173 -0.140472 … 0.267528 0.876781 -0.120359
980  -0.428430 0.008812 1.185911 … -0.131964 -0.499977 -0.067951981 -
     0.138277 1.160096 -1.090570 … 0.180030 0.336668 -0.134927
982  -1.163118 1.219110 -0.411377 … 0.236633 0.655310 0.175033
983  0.543659 -0.423814 1.219402 … 0.073917 0.519319 0.071452

        V24      V25      V26      V27      V28 Amount Class
 0  0.320198 0.044519 0.177840 0.261145 -0.143276   0.00     1
 1-0.293803 0.279798 -0.145362 -0.252773 0.035764529.00     1
 2    -0.087330 -0.156114 -0.542628 0.039566 -    239.93     1
 0.153029
 3-0.053502 0.252405 -0.657488 -0.827136 0.849573 59.00     1
 4 -1.632653 1.488901 0.566797 -0.010016 0.146793   1.00     1
 5  0.128940 1.488481 0.507963 0.735822 0.513574   1.00     1
 6 -0.759908 1.605056 0.540675 0.737040 0.496699   1.00     1
 7 -0.716522 1.415717 0.555265 0.530507 0.404474   1.00     1
 8 -0.219845 1.474753 0.491192 0.518868 0.402528   1.00     1
 9 -0.364700 -1.843078 0.351909 0.594550 0.099372   1.00     1
 10 0.473246 -1.959304 0.319476 0.600485 0.129305   1.00     1
 11 -0.372216 -2.032068 0.366778 0.395171          1.00     1
 0.020206
 12 -0.425550 0.123644 0.321985 0.264028 0.132817   1.00     1
 13 0.461700 0.044146 0.305704 0.530981 0.243746   1.00     1
 14 -0.234649 -1.004881 0.435832 0.618324          1.00     1
 0.148469
 15 -0.279642 1.106766 0.323885 0.894767 0.569519   1.00     1
 16 -0.263743 1.539916 0.523574 0.891025 0.572741   1.00     1
 17 0.083008 -1.911034 0.322188 0.620867 0.185030   1.00     1
 18 0.131464 -1.908217 0.334808 0.748534 0.175414   1.00     1
 19 0.071696 0.092007 0.308498 0.552591 0.298954   1.00     1
 20 0.070614 0.058504 0.304883 0.418012 0.208858   1.00     1
 21 -0.217565 -0.138776 -0.424453 -1.002041        1.10     1
 0.890780
 22 -0.286016 1.007934 0.413196 0.280284 0.303937   1.00     1
 23 -0.288835 1.011752 0.425965 0.413140 0.308205   1.00     1
```

```
240.170279 -0.393844 0.296367 1.985913 -0.900452 1809.68   1
25 -1.164555 1.701796 0.690806 2.119749 1.108933  1.00     1
26 -0.403114 1.568445 0.521884 0.527938 0.411910  1.00     1
27 -0.769348 -1.746337 0.502040 1.977258 0.711607 1.00     1
28 -1.189308 1.188536 0.605242 1.881529 0.875260  1.00     1
29 -0.417918 -0.911188 0.466524 0.627393 0.157851 1.00     1
..       …       …       …       …       …       …    …
954 1.034758 -0.700922 0.648581 -0.057502 0.127988202.31   0
955 -0.081333 -0.074885 0.330273 -0.149884 -0.2534222.29   0
956 0.051060 -0.429060 0.750027 -0.282959 -0.101003126.00  0
957 -0.458899 -0.679002 0.191879 0.037741 -0.07671028.98   0
958 0.470629 0.552619 0.352217 1.019398 -0.462908298.33    0
959 1.012433 0.773454 -0.328314 -0.307860 0.06235627.42    0
960 0.334310 -0.346491 0.240723 -0.052977 0.07627342.81    0
961 0.066650 -0.089823 -0.114947 0.135096 -0.0195037.50    0
962 -0.968079 0.998955 -0.196661 0.006097 -0.0096091.00    0
963 1.013573 0.866757 -0.334311 0.011730 0.02499980.43     0
964 -0.466664 0.157056 0.950496 -0.084727 0.01212894.85    0
965 -0.031352 0.100761 -0.228967 0.027680       149.92     0
0.042448
966 -0.859272 0.411489 -0.300106 0.226980         8.52     0
0.054999
967 0.657254 -0.390779 -0.601500 0.054462 -       7.49     0
0.037135
968 0.571201 0.525534 -0.122198 -0.036533       275.00     0
0.046692
969 -0.025219 -0.504606 0.437847 0.504577        10.00     0
0.209921
970 0.646476 -0.201730 -0.402726 0.180500 -       1.79     0
0.156480
971 0.457462 0.330857 0.584320 -0.057324          8.67     0
0.033154
972 0.613370 -0.545931 0.690521 0.104853 -        5.30     0
0.048594
973 -1.321815 0.279711 0.549207 0.263836         25.00     0
0.262827
974 0.086910 -0.469945 -0.799564 0.093064 -       1.00     0
0.051164
975 -0.034412 0.545794 -0.839016 0.040610        59.90     0
0.018029
976 0.101930 0.718241 -0.357059 -0.001664         0.76     0
0.078659
977 0.522385 1.049123 0.872576 -0.046526          0.76     0
0.081082
978 -0.545857 0.253408 -0.567055 0.064468        12.32     0
0.027141
979 0.631436 -0.284440 -0.118500 0.401493        20.80     0
0.185501
```

```
980 0.468261 0.074170 0.924785 -0.065359        176.42     0
0.043538
981 -0.735988 -0.324900 0.418468 -0.177558 -      0.76     0
0.104505
982 0.689492 -0.154773 0.106330 0.032736 -       22.66     0
0.035525
983 -0.503598 -0.355496 0.497140 -0.499752 -     27.31     0
0.388871
[984 rows x 31 columns]
```

[12]:
```
data['Class'].value_counts()
```

[12]:
```
1
492 0
492
Name: Class, dtype: int64
```

[13]:
```
X = data.drop('Class', axis = 1)
y = data['Class']
```

[14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
↪random_state = 0, stratify = y)
```

[16]:
```
X_train.shape, X_test.shape
```

[16]:
```
((787, 30), (197, 30))
```

[18]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[19]:
```
y_train = y_train.to_numpy()
y_test = y_test.to_numpy()
```

[20]:
```
X_train.shape
```

[20]:
```
(787, 30)
```

[22]:
```
X_train = X_train.reshape(X_train.shape[0],
X_train.shape[1], 1) X_test =
X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

[23]:
```
X_train.shape, X_test.shape
```

[23]:
```
((787, 30, 1), (197, 30, 1))
```

0.1.2 Build CNN

[24]:
```
epochs = 20
model = Sequential()
model.add(Conv1D(32, 2, activation='relu', input_shape = X_train[0].shape))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Conv1D(64, 2, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(1, activation='sigmoid'))
```

[25]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=============================================================
==== conv1d (Conv1D)   (None, 29, 32)   96
_________________________________________________________________
batch_normalization (BatchNo (None, 29, 32)  128
______________________________________________________________
___ dropout (Dropout)  (None, 29, 32)   0
_______________________________________________________________
__ conv1d_1 (Conv1D)   (None, 28, 64)   4160
_________________________________________________________________
batch_normalization_1 (Batch (None, 28, 64)  256
_________________________________________________________________
dropout_1 (Dropout)          (None, 28, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 1792)              0
_________________________________________________________________
dense (Dense)                (None, 64)                114752
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 119,457
Trainable params: 119,265
Non-trainable params: 192
```

---

[28]:
```
model.compile(optimizer=Adam(lr=0.0001), loss =
 'binary_crossentropy',␣ ↪metrics=['accuracy'])
```

[29]:
```
history = model.fit(X_train, y_train, epochs=epochs,
 validation_data=(X_test,␣ ↪y_test), verbose=1)
```

```
WARNING: Logging before flag parsing goes to stderr.
W0904 18:49:33.834567 8812 deprecation.py:323] From
C:\ProgramData\Anaconda3\lib\site-
packages\tensorflow_core\python\ops\nn_impl.py:183: where (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in
a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

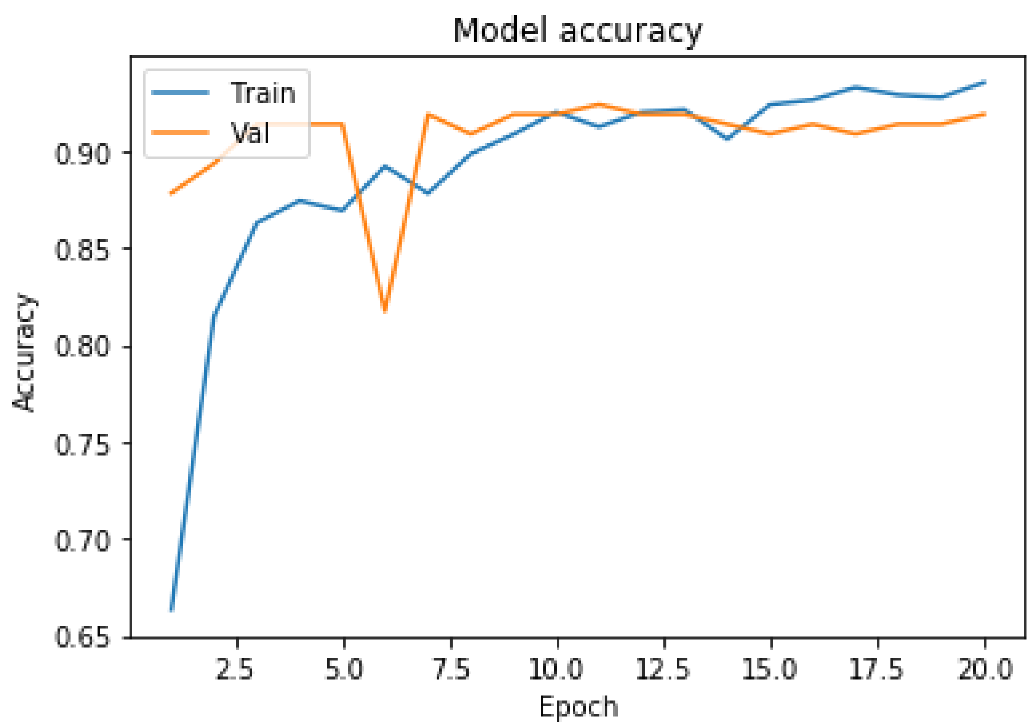
```
Train on 787 samples, validate on 197 samples
Epoch 1/20
787/787 [==============================] - 2s 2ms/sample - loss:
0.7682 -
accuracy: 0.6633 - val_loss: 0.6181 - val_accuracy: 0.8782
Epoch 2/20
787/787 [==============================] - 0s 243us/sample - loss:
0.4962 -
accuracy: 0.8145 - val_loss: 0.5716 - val_accuracy: 0.8934
Epoch 3/20
787/787 [==============================] - 0s 250us/sample - loss:
0.4026 accuracy: 0.8628 - val_loss: 0.5365 - val_accuracy: 0.9137
Epoch 4/20
787/787 [==============================] - 0s 244us/sample - loss:
0.3464 -
accuracy: 0.8742 - val_loss: 0.5008 - val_accuracy: 0.9137
Epoch 5/20
787/787 [==============================] - 0s 239us/sample - loss:
0.3336 accuracy: 0.8691 - val_loss: 0.4857 - val_accuracy: 0.9137
Epoch 6/20
787/787 [==============================] - 0s 227us/sample - loss:
0.2994 accuracy: 0.8920 - val_loss: 0.4685 - val_accuracy: 0.8173
Epoch 7/20
787/787 [==============================] - 0s 250us/sample - loss:
0.3255 accuracy: 0.8780 - val_loss: 0.4145 - val_accuracy: 0.9188
Epoch 8/20
787/787 [==============================] - 0s 230us/sample - loss:
0.2744 -
accuracy: 0.8983 - val_loss: 0.3900 - val_accuracy: 0.9086
```

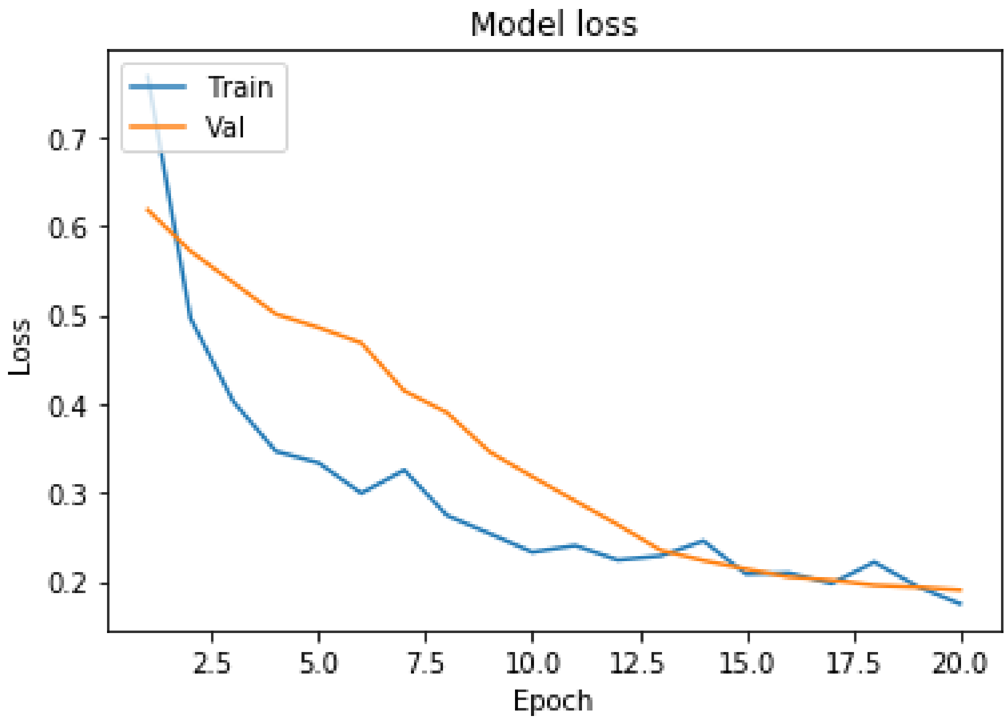
```
Epoch 9/20
787/787 [==============================] - 0s 267us/sample - loss:
0.2540 -
accuracy: 0.9085 - val_loss: 0.3458 - val_accuracy: 0.9188
Epoch 10/20
787/787 [==============================] - 0s 264us/sample - loss:
0.2330 -
accuracy: 0.9199 - val_loss: 0.3180 - val_accuracy: 0.9188
Epoch 11/20
787/787 [==============================] - 0s 226us/sample - loss:
0.2405 -
accuracy: 0.9123 - val_loss: 0.2908 - val_accuracy: 0.9239
Epoch 12/20
787/787 [==============================] - 0s 231us/sample - loss:
0.2241 -
accuracy: 0.9199 - val_loss: 0.2639 - val_accuracy: 0.9188
Epoch 13/20
787/787 [==============================] - 0s 229us/sample - loss:
0.2288 -
accuracy: 0.9212 - val_loss: 0.2350 - val_accuracy: 0.9188
Epoch 14/20
787/787 [==============================] - 0s 249us/sample - loss:
0.2457 -
accuracy: 0.9060 - val_loss: 0.2236 - val_accuracy: 0.9137
Epoch 15/20
787/787 [==============================] - 0s 273us/sample - loss:
0.2084 -
accuracy: 0.9238 - val_loss: 0.2141 - val_accuracy: 0.9086
Epoch 16/20
787/787 [==============================] - 0s 268us/sample - loss:
0.2096 -
accuracy: 0.9263 - val_loss: 0.2050 - val_accuracy: 0.9137
Epoch 17/20
787/787 [==============================] - 0s 257us/sample - loss:
0.1976 -
accuracy: 0.9327 - val_loss: 0.2014 - val_accuracy: 0.9086
Epoch 18/20
787/787 [==============================] - 0s 262us/sample - loss:
0.2219 accuracy: 0.9288 - val_loss: 0.1957 - val_accuracy: 0.9137
Epoch 19/20
787/787 [==============================] - 0s 257us/sample - loss:
0.1945 accuracy: 0.9276 - val_loss: 0.1937 - val_accuracy: 0.9137
Epoch 20/20
787/787 [==============================] - 0s 259us/sample - loss:
0.1750 accuracy: 0.9352 - val_loss: 0.1904 - val_accuracy: 0.9188
```

[30]:
```
def plot_learningCurve(history, epoch):
  # Plot training & validation accuracy
  values epoch_range = range(1, epoch+1)
  plt.plot(epoch_range,
  history.history['accuracy'])
  plt.plot(epoch_range,
  history.history['val_accuracy'])
  plt.title('Model accuracy')
  plt.ylabel('Accuracy')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Val'], loc='upper left')
  plt.show()

  # Plot training & validation loss values
  plt.plot(epoch_range, history.history['loss'])
  plt.plot(epoch_range, history.history['val_loss'])
  plt.title('Model loss')
  plt.ylabel('Loss')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Val'], loc='upper left')
  plt.show()
```

[31]:
```
plot_learningCurve(history, epochs)
```





[ ]:

```

```

0.1.3 Adding MaxPool

[33]:

```
epochs = 50
model = Sequential()
model.add(Conv1D(32, 2, activation='relu', input_shape = X_train[0].shape))
model.add(BatchNormalization())
model.add(MaxPool1D(2))
model.add(Dropout(0.2))

model.add(Conv1D(64, 2, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool1D(2))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(1, activation='sigmoid'))
```

```
model.compile(optimizer=Adam(lr=0.0001), loss =
'binary_crossentropy',␣
 ↪metrics=['accuracy']) history = model.fit(X_train, y_train,
epochs=epochs, validation_data=(X_test,␣
 ↪y_test), verbose=1)
plot_learningCurve(history, epochs)
```

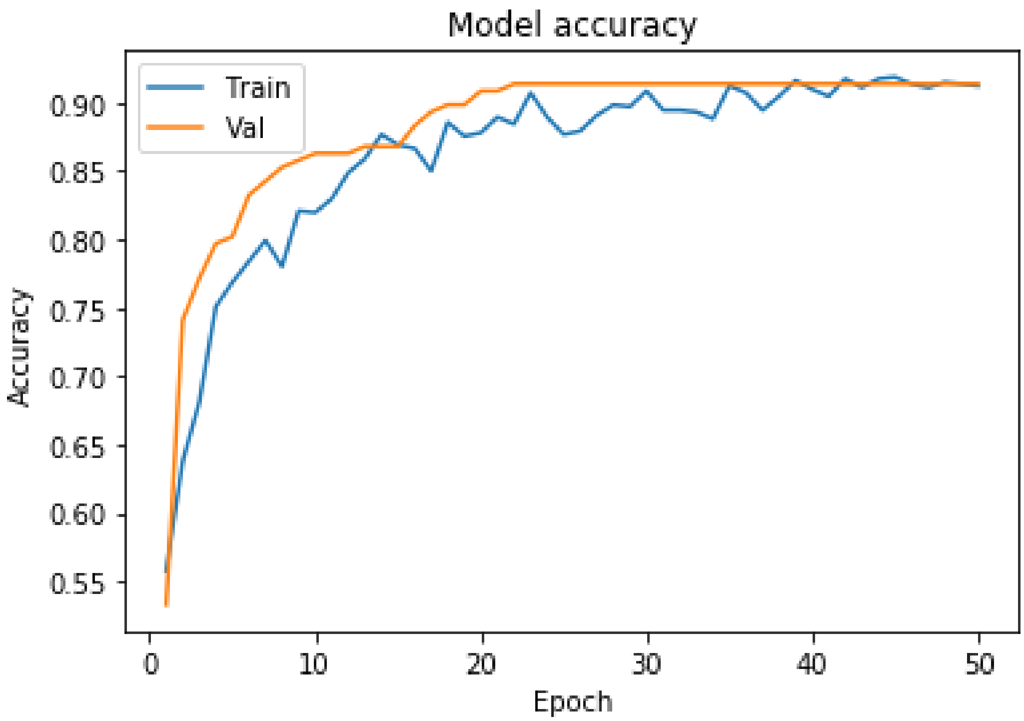
```
Train on 787 samples, validate on 197 samples
Epoch 1/50
787/787 [==============================] - 2s 2ms/sample - loss:
1.1020 -
accuracy: 0.5578 - val_loss: 0.6766 - val_accuracy: 0.5330
Epoch 2/50
787/787 [==============================] - 0s 203us/sample - loss:
0.8175 -
accuracy: 0.6379 - val_loss: 0.6338 - val_accuracy: 0.7411
Epoch 3/50
787/787 [==============================] - 0s 196us/sample - loss:
0.7102 accuracy: 0.6811 - val_loss: 0.5959 - val_accuracy: 0.7716
Epoch 4/50
787/787 [==============================] - 0s 196us/sample - loss:
0.6062 -
accuracy: 0.7510 - val_loss: 0.5602 - val_accuracy: 0.7970
Epoch 5/50
787/787 [==============================] - 0s 197us/sample - loss:
0.5300 -
accuracy: 0.7687 - val_loss: 0.5268 - val_accuracy: 0.8020
Epoch 6/50
787/787 [==============================] - 0s 204us/sample - loss:
0.5243 -
accuracy: 0.7840 - val_loss: 0.4918 - val_accuracy: 0.8325
Epoch 7/50
787/787 [==============================] - 0s 217us/sample - loss:
0.5553 -
accuracy: 0.7992 - val_loss: 0.4584 - val_accuracy: 0.8426
Epoch 8/50
787/787 [==============================] - 0s 216us/sample - loss:
0.4727 -
accuracy: 0.7802 - val_loss: 0.4261 - val_accuracy: 0.8528
Epoch 9/50
787/787 [==============================] - 0s 222us/sample - loss:
0.4551 accuracy: 0.8208 - val_loss: 0.3972 - val_accuracy: 0.8579
Epoch 10/50
787/787 [==============================] - 0s 216us/sample - loss:
0.4387 accuracy: 0.8196 - val_loss: 0.3710 - val_accuracy: 0.8629
Epoch 11/50
```

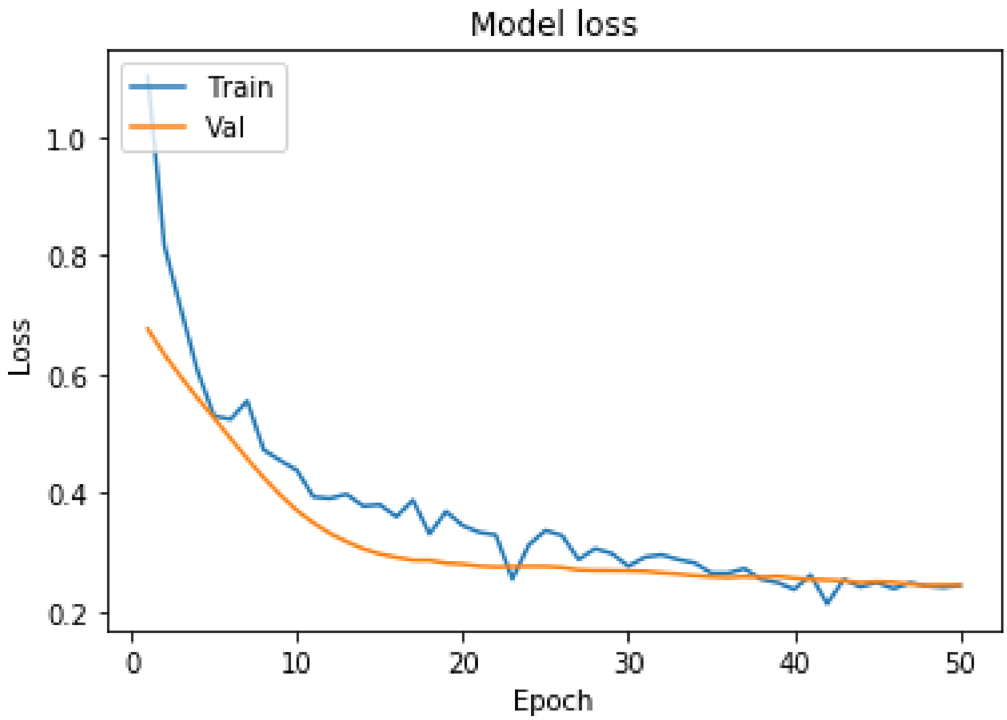
```
787/787 [==============================] - 0s 194us/sample - loss:
0.3938 accuracy: 0.8297 - val_loss: 0.3500 - val_accuracy: 0.8629
Epoch 12/50
787/787 [==============================] - 0s 194us/sample - loss:
0.3911 accuracy: 0.8488 - val_loss: 0.3322 - val_accuracy: 0.8629
Epoch 13/50
787/787 [==============================] - 0s 201us/sample - loss:
0.3984 -
accuracy: 0.8590 - val_loss: 0.3183 - val_accuracy:
0.8680 Epoch 14/50
787/787 [==============================] - 0s 216us/sample - loss:
0.3784 -
accuracy: 0.8767 - val_loss: 0.3063 - val_accuracy: 0.8680
Epoch 15/50
787/787 [==============================] - 0s 203us/sample - loss:
0.3806 -
accuracy: 0.8691 - val_loss: 0.2976 - val_accuracy: 0.8680
Epoch 16/50
787/787 [==============================] - 0s 223us/sample - loss:
0.3605 -
accuracy: 0.8666 - val_loss: 0.2918 - val_accuracy: 0.8832
Epoch 17/50
787/787 [==============================] - 0s 210us/sample - loss:
0.3881 -
accuracy: 0.8501 - val_loss: 0.2870 - val_accuracy: 0.8934
Epoch 18/50
787/787 [==============================] - 0s 196us/sample - loss:
0.3314 -
accuracy: 0.8856 - val_loss: 0.2868 - val_accuracy: 0.8985
Epoch 19/50
787/787 [==============================] - 0s 190us/sample - loss:
0.3690 accuracy: 0.8755 - val_loss: 0.2820 - val_accuracy: 0.8985
Epoch 20/50
787/787 [==============================] - 0s 194us/sample - loss:
0.3454 -
accuracy: 0.8780 - val_loss: 0.2805 - val_accuracy: 0.9086
Epoch 21/50
787/787 [==============================] - 0s 221us/sample - loss:
0.3340 -
accuracy: 0.8895 - val_loss: 0.2770 - val_accuracy: 0.9086
Epoch 22/50
787/787 [==============================] - 0s 221us/sample - loss:
0.3296 -
accuracy: 0.8844 - val_loss: 0.2755 - val_accuracy: 0.9137
Epoch 23/50
```

```
787/787 [==============================] - 0s 199us/sample - loss:
0.2552 -
accuracy: 0.9072 - val_loss: 0.2762 - val_accuracy: 0.9137
Epoch 24/50
787/787 [==============================] - 0s 204us/sample - loss:
0.3131 -
accuracy: 0.8895 - val_loss: 0.2765 - val_accuracy: 0.9137
Epoch 25/50
787/787 [==============================] - 0s 213us/sample - loss:
0.3371 accuracy: 0.8767 - val_loss: 0.2763 - val_accuracy: 0.9137
Epoch 26/50
787/787 [==============================] - 0s 202us/sample - loss:
0.3286 accuracy: 0.8793 - val_loss: 0.2748 - val_accuracy: 0.9137
Epoch 27/50
787/787 [==============================] - 0s 204us/sample - loss:
0.2882 accuracy: 0.8907 - val_loss: 0.2708 - val_accuracy: 0.9137
Epoch 28/50
787/787 [==============================] - 0s 221us/sample - loss:
0.3065 accuracy: 0.8983 - val_loss: 0.2697 - val_accuracy: 0.9137
Epoch 29/50
787/787 [==============================] - 0s 201us/sample - loss:
0.2984 -
accuracy: 0.8971 - val_loss: 0.2696 - val_accuracy:
0.9137 Epoch 30/50
787/787 [==============================] - 0s 231us/sample - loss:
0.2769 -
accuracy: 0.9085 - val_loss: 0.2691 - val_accuracy: 0.9137
Epoch 31/50
787/787 [==============================] - 0s 258us/sample - loss:
0.2923 -
accuracy: 0.8945 - val_loss: 0.2683 - val_accuracy: 0.9137
Epoch 32/50
787/787 [==============================] - 0s 241us/sample - loss:
0.2961 -
accuracy: 0.8945 - val_loss: 0.2658 - val_accuracy: 0.9137
Epoch 33/50
787/787 [==============================] - 0s 227us/sample - loss:
0.2881 -
accuracy: 0.8933 - val_loss: 0.2636 - val_accuracy: 0.9137
Epoch 34/50
787/787 [==============================] - 0s 222us/sample - loss:
0.2823 -
accuracy: 0.8882 - val_loss: 0.2611 - val_accuracy: 0.9137
Epoch 35/50
787/787 [==============================] - 0s 244us/sample - loss:
0.2652 accuracy: 0.9123 - val_loss: 0.2592 - val_accuracy: 0.9137
```

```
Epoch 36/50
787/787 [==============================] - 0s 249us/sample - loss:
0.2650 -
accuracy: 0.9072 - val_loss: 0.2581 - val_accuracy: 0.9137
Epoch 37/50
787/787 [==============================] - 0s 211us/sample - loss:
0.2729 -
accuracy: 0.8945 - val_loss: 0.2594 - val_accuracy: 0.9137
Epoch 38/50
787/787 [==============================] - 0s 198us/sample - loss:
0.2550 -
accuracy: 0.9047 - val_loss: 0.2584 - val_accuracy: 0.9137
Epoch 39/50
787/787 [==============================] - 0s 190us/sample - loss:
0.2493 -
accuracy: 0.9161 - val_loss: 0.2599 - val_accuracy: 0.9137
Epoch 40/50
787/787 [==============================] - 0s 188us/sample - loss:
0.2378 -
accuracy: 0.9098 - val_loss: 0.2568 - val_accuracy: 0.9137
Epoch 41/50
787/787 [==============================] - 0s 207us/sample - loss:
0.2616 accuracy: 0.9047 - val_loss: 0.2544 - val_accuracy: 0.9137
Epoch 42/50
787/787 [==============================] - 0s 213us/sample - loss:
0.2136 accuracy: 0.9174 - val_loss: 0.2537 - val_accuracy: 0.9137
Epoch 43/50
787/787 [==============================] - 0s 194us/sample - loss:
0.2545 accuracy: 0.9111 - val_loss: 0.2522 - val_accuracy: 0.9137
Epoch 44/50
787/787 [==============================] - 0s 223us/sample - loss:
0.2420 accuracy: 0.9174 - val_loss: 0.2494 - val_accuracy: 0.9137
Epoch 45/50
787/787 [==============================] - 0s 211us/sample - loss:
0.2494 -
accuracy: 0.9187 - val_loss: 0.2509 - val_accuracy:
0.9137 Epoch 46/50
787/787 [==============================] - 0s 212us/sample - loss:
0.2390 -
accuracy: 0.9136 - val_loss: 0.2498 - val_accuracy: 0.9137
Epoch 47/50
787/787 [==============================] - 0s 225us/sample - loss:
0.2490 -
accuracy: 0.9111 - val_loss: 0.2466 - val_accuracy: 0.9137
Epoch 48/50
```

```
787/787 [==============================] - 0s 210us/sample - loss:
0.2435 -
accuracy: 0.9149 - val_loss: 0.2443 - val_accuracy: 0.9137
Epoch 49/50
787/787 [==============================] - 0s 192us/sample - loss:
0.2413 -
accuracy: 0.9136 - val_loss: 0.2453 - val_accuracy: 0.9137
Epoch 50/50
787/787 [==============================] - 0s 194us/sample - loss:
0.2445 accuracy: 0.9123 - val_loss: 0.2449 - val_accuracy: 0.9137
```




Model loss
Train
Val
1.0
0.8
0.6
0.4
0.2
Loss
0
10
20
30
40
50
Epoch