Assignment 2

Introduction

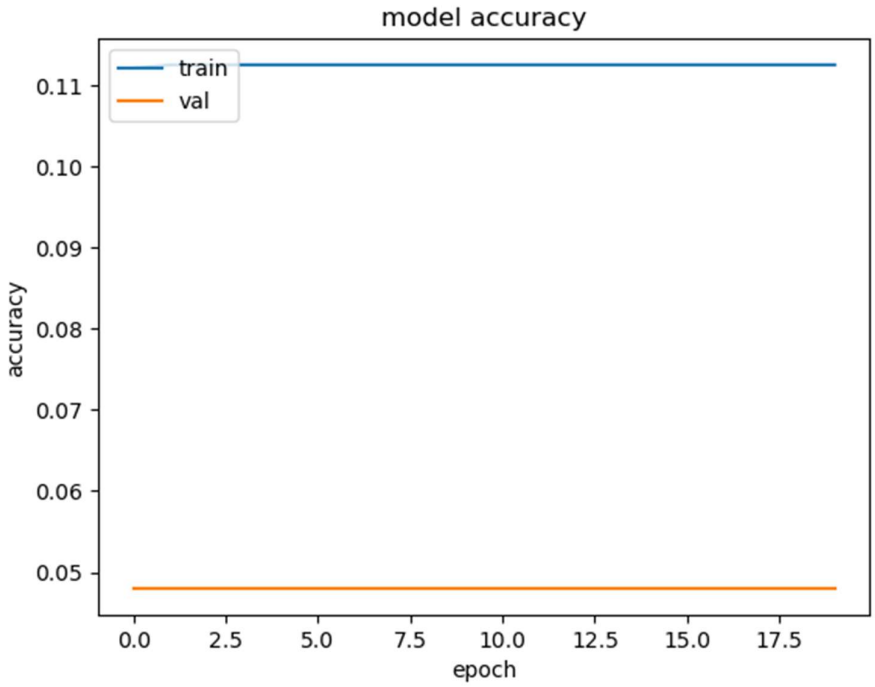
The aim of the assignment is to develop a model for multiple audio event detection. The dataset given to us involves 10000 samples of audio event spectrograms along with their labels. Also, a validation dataset containing 2000 samples was provided containing audio event spectrograms and their labels as well.

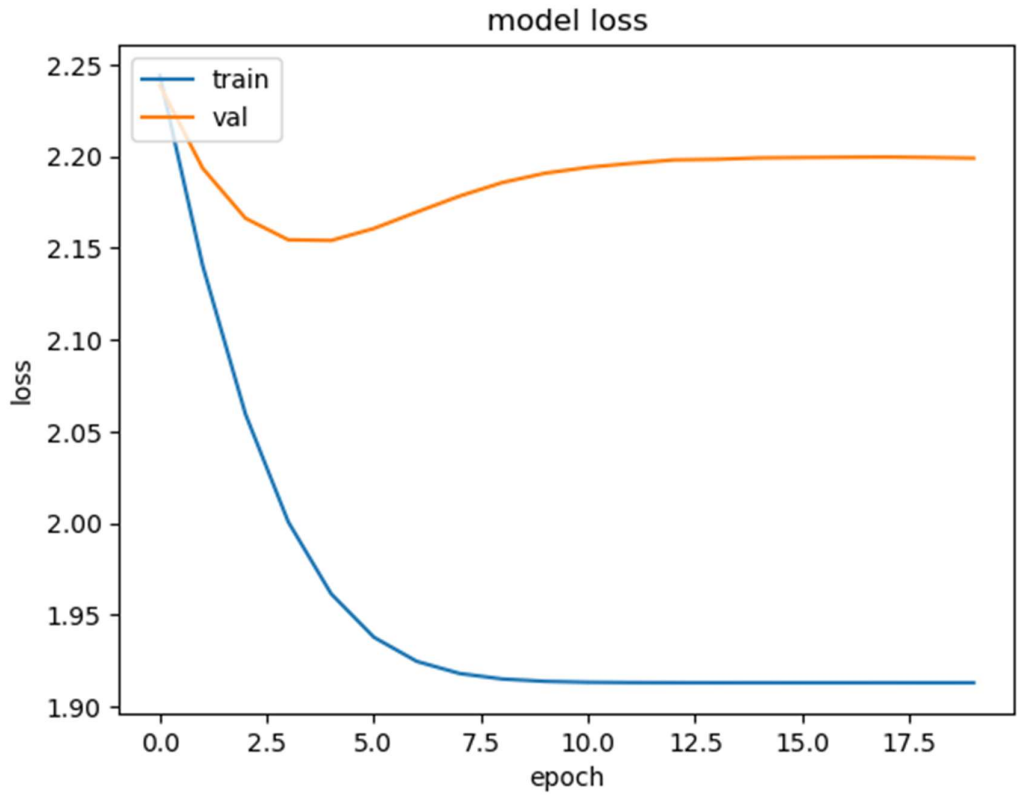
My method

I first did the pre-processing of the given data. I converted all the spectrograms provided from power units to decibel units. Also, conversion of the event-roll from (1000,11) form was done using the function provided to us, resulting in a multihot vector. I did this to all the data provided – training data, validation data as well as test data.

An additional change I did to the multihot vectors was to divide each vector by the sum of its values. For example, if a vector was [1, 0, 0, 0, 0, 1, 0, 0, 0, 0], I made it [0.5, 0, 0, 0, 0, 0.5, 0, 0, 0, 0] I did this so that while training, the model tries to get values in multihot vectors instead of 1. This created a much more stable training environment for the data. When I did not apply this the loss exponentially increased and did not train properly.

After the pre-processing, I created a model for training. Initially, I used a simple **Deep/Dense Neural Network (DNN).** Using this model, I could only achieve a training accuracy of 11.25% and validation accuracy of 4.8% while training. After applying a threshold, validation accuracy of 33% was achieved but on **test data an accuracy of 54.97% as well as f1 score of 57.93% was achieved.**





The classification report is as given –

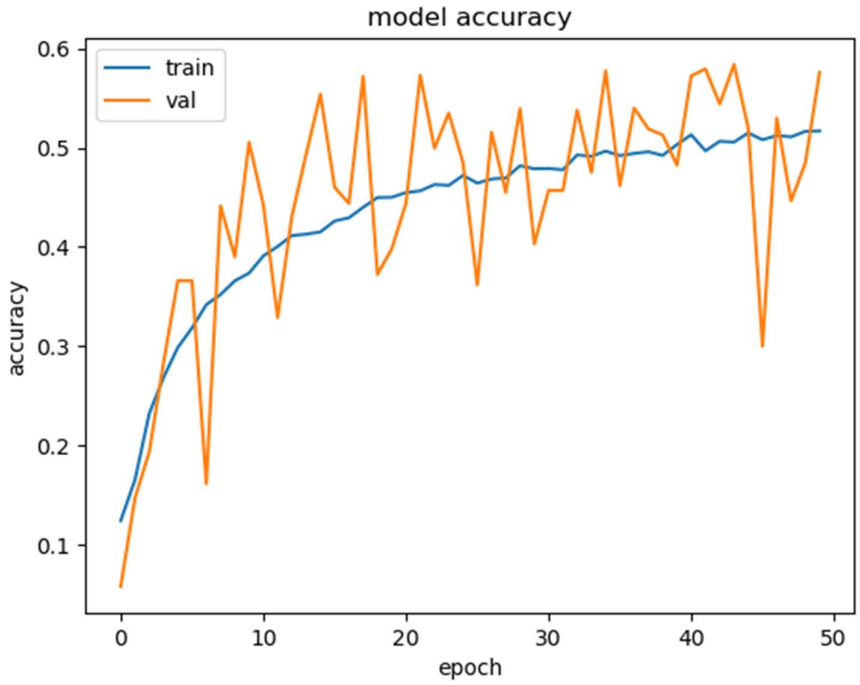
	precision	recall	f1-score	support
0	0.00	0.00	0.00	400
1	0.00	0.00	0.00	266
2	0.00	0.00	0.00	284
3	0.28	1.00	0.43	689
4	0.00	0.00	0.00	341
5	0.00	0.00	0.00	283
6	0.00	0.00	0.00	377
7	0.00	0.00	0.00	306
8	0.95	1.00	0.97	2373
9	0.00	0.00	0.00	251
micro avg	0.61	0.55	0.58	5570
macro avg	0.12	0.20	0.14	5570
weighted avg	0.44	0.55	0.47	5570
samples avg	0.61	0.54	0.57	5570

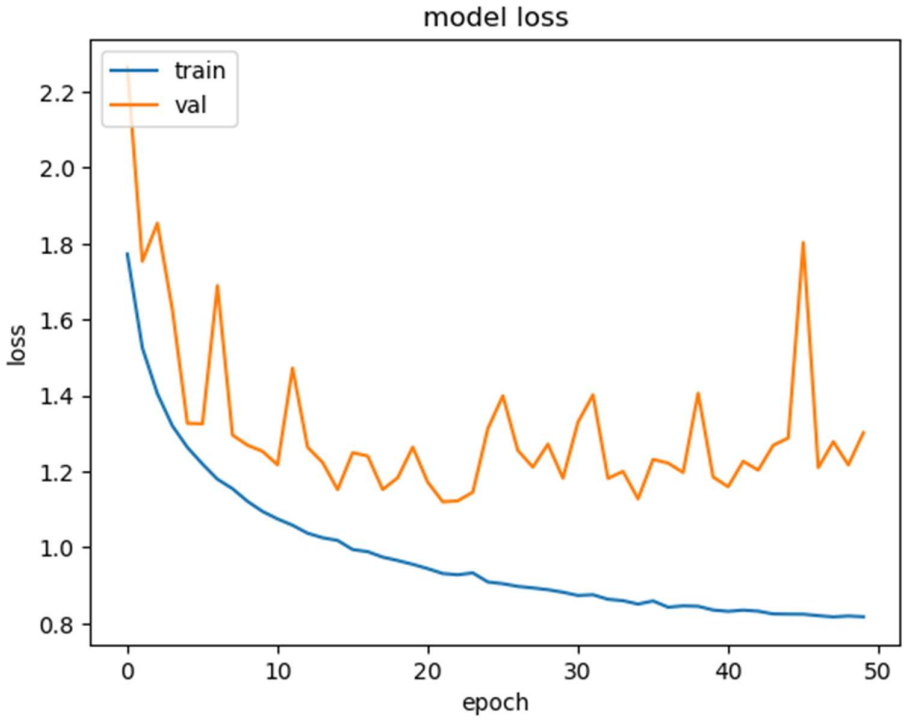
The confusion matrices are –

```
array([[[2100,    0],
        [ 400,    0]],

       [[2234,    0],
        [ 266,    0]],

       [[2216,    0],
        [ 284,    0]],

       [[   0, 1811],
        [   0,  689]],

       [[2159,    0],
        [ 341,    0]],

       [[2217,    0],
        [ 283,    0]],

       [[2123,    0],
        [ 377,    0]],

       [[2194,    0],
        [ 306,    0]],

       [[   0,  127],
        [   0, 2373]],

       [[2249,    0],
        [ 251,    0]]], dtype=int64)
```

```
accuracy =  0.5497307001795332
TP =  3062
FP =  1938
TN =  17492
FN =  2508
Precision =  0.6123998775200244
Recall =  0.5497306014846317
F1 score =  0.579375481669729
```

Due to the low training accuracy, I moved on to using a **Convoluted Neural Network (CNN)** model. This model performed much better while training, achieving a training accuracy of 51.7% and validation accuracy of **57.6%**. The validation accuracy was fluctuating a lot but by saving the best model I was able to achieve a good model for predicting audio events.





	precision	recall	f1-score	support
0	0.79	0.27	0.40	400
1	0.74	0.44	0.55	266
2	0.68	0.81	0.74	284
3	0.74	0.58	0.65	689
4	0.66	0.58	0.62	341
5	0.51	0.78	0.62	283
6	0.76	0.82	0.79	377
7	0.58	0.51	0.54	306
8	0.95	1.00	0.97	2373
9	0.59	0.77	0.67	251
micro avg	0.80	0.77	0.78	5570
macro avg	0.70	0.66	0.66	5570
weighted avg	0.80	0.77	0.77	5570
samples avg	0.81	0.79	0.78	5570

```
array([[[2072,   28],
        [ 292,  108]],

       [[2193,   41],
        [ 149,  117]],

       [[2106,  110],
        [  55,  229]],

       [[1667,  144],
        [ 286,  403]],

       [[2058,  101],
        [ 144,  197]],

       [[2004,  213],
        [  61,  222]],

       [[2028,   95],
        [  69,  308]],

       [[2078,  116],
        [ 149,  157]],

       [[   5,  122],
        [   6, 2367]],

       [[2114,  135],
        [  57,  194]]], dtype=int64)
```

```
accuracy =  0.7723518850987433
TP =  4302
FP =  1105
TN =  18325
FN =  1268
Precision =  0.7956352875901609
Recall =  0.7723518850987433
F1 score =  0.7838207160426347
```

In the end, an **f1 score of 78.38** was achieved.

Observations and Discussions

- One thing that stands out very strongly is that the DNN model performed much worse than the CNN model. This was strongly evident in the training period and carried through to the validation accuracy and test accuracy.
- CNN model was very turbulent but upon saving the better models, a much higher accuracy was achieved. In the end, my best model achieved 60% better f1 score compared to my best DNN model.
- CNN model performed better because of patterns in the dataset that the model was able to identify and classify the test data on the basis of these patterns. DNN model does not have this feature and hence performed poorly.
- DNN model performed poorly on multiple classes as observed in the confusion matrix (multiple classes have 0 TP predictions). CNN did not perform this way and accurately performed for multiple classes.