Assignment 3

How to Run the Code: By running all the cells of 1905105.ipynb we can get the output.

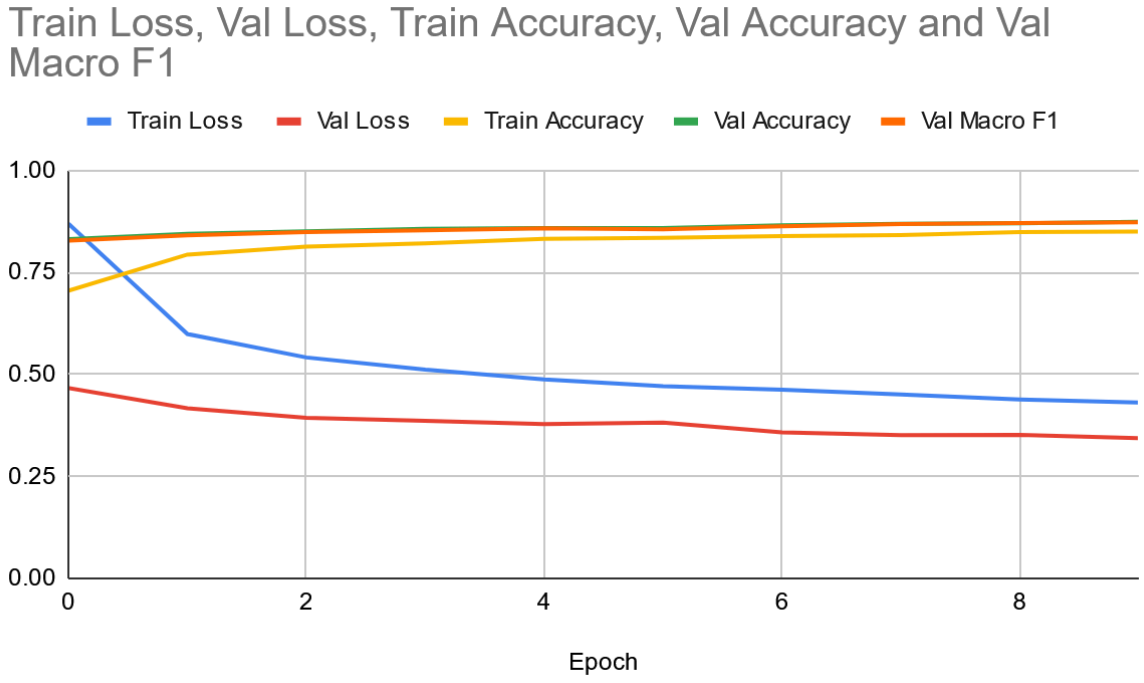
Models:

1.

```
layers = [
    DenseLayer(input_size, 128),
    BatchNormalization(128),
    ReLU(),
    Dropout(0.5),
    DenseLayer(128, 64),
    BatchNormalization(64),
    ReLU(),
    Dropout(0.5),
    DenseLayer(64, num_classes)
]
```

Learning Rate: 0.001

Epoch	Train Loss	Val Loss	Train Accuracy	Val Accuracy	Val Macro F1
0	0.869477	0.466470	0.705063	0.831217	0.828013
1	0.598904	0.417039	0.793646	0.844418	0.840624
2	0.541512	0.393376	0.813000	0.850936	0.848939
3	0.511400	0.386441	0.821333	0.856534	0.853373
4	0.487886	0.378241	0.832229	0.858874	0.857919
5	0.470874	0.381653	0.834750	0.858874	0.855871
6	0.462343	0.358065	0.839063	0.865224	0.862896
7	0.450566	0.351085	0.841521	0.869235	0.868572
8	0.438674	0.351771	0.848917	0.870655	0.871131
9	0.430723	0.343340	0.850250	0.874164	0.873058

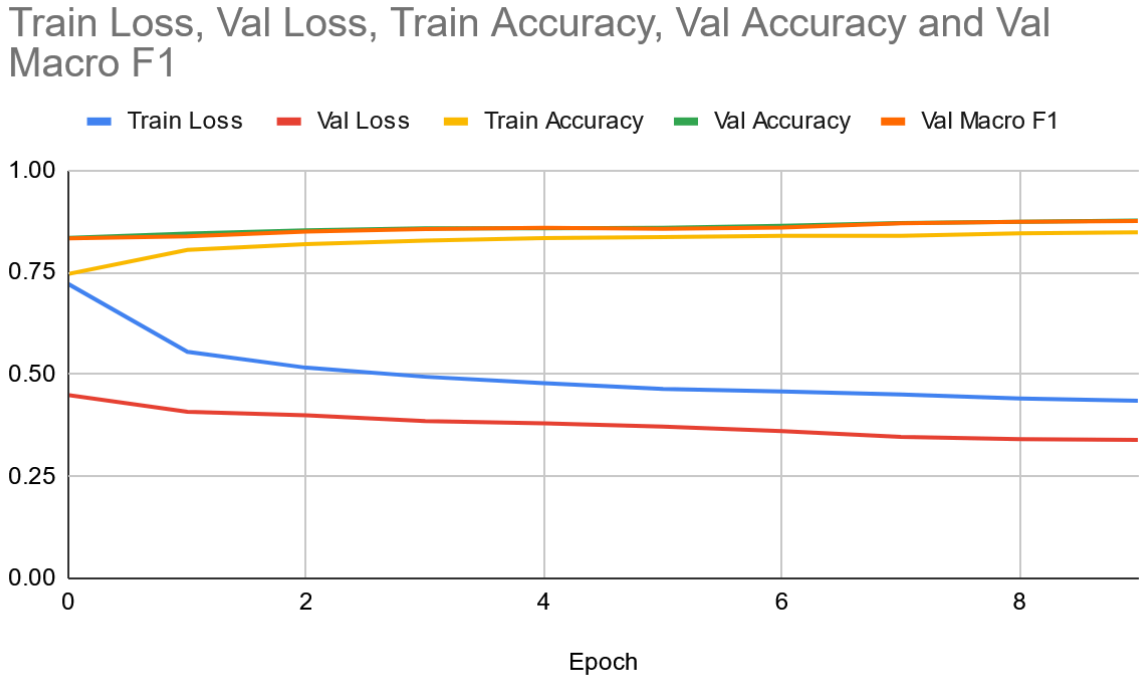

Train Loss, Val Loss, Train Accuracy, Val Accuracy and Val Macro F1
Train Loss
Val Loss
Train Accuracy
Val Accuracy
Val Macro F1
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
Epoch

2.

```
layers = [
    DenseLayer(input_size, 128),
    BatchNormalization(128),
    ReLU(),
    Dropout(0.5),
    DenseLayer(128, 64),
    BatchNormalization(64),
    ReLU(),
    Dropout(0.5),
    DenseLayer(64, num_classes)
]
```

Learning Rate: 0.005

Epoch	Train Loss	Val Loss	Train Accuracy	Val Accuracy	Val Macro F1
0	0.721180	0.449120	0.746354	0.834809	0.833203
1	0.555435	0.408080	0.805250	0.844920	0.838315
2	0.516300	0.399454	0.819063	0.853025	0.849956
3	0.494124	0.385470	0.827750	0.858205	0.856172
4	0.478194	0.379808	0.833833	0.858205	0.859047
5	0.464282	0.371861	0.836562	0.859291	0.856672
6	0.457944	0.360727	0.839625	0.864138	0.860031
7	0.450698	0.346692	0.839458	0.871240	0.870589
8	0.440605	0.341425	0.845750	0.874164	0.873999
9	0.435051	0.339127	0.848063	0.877172	0.876152

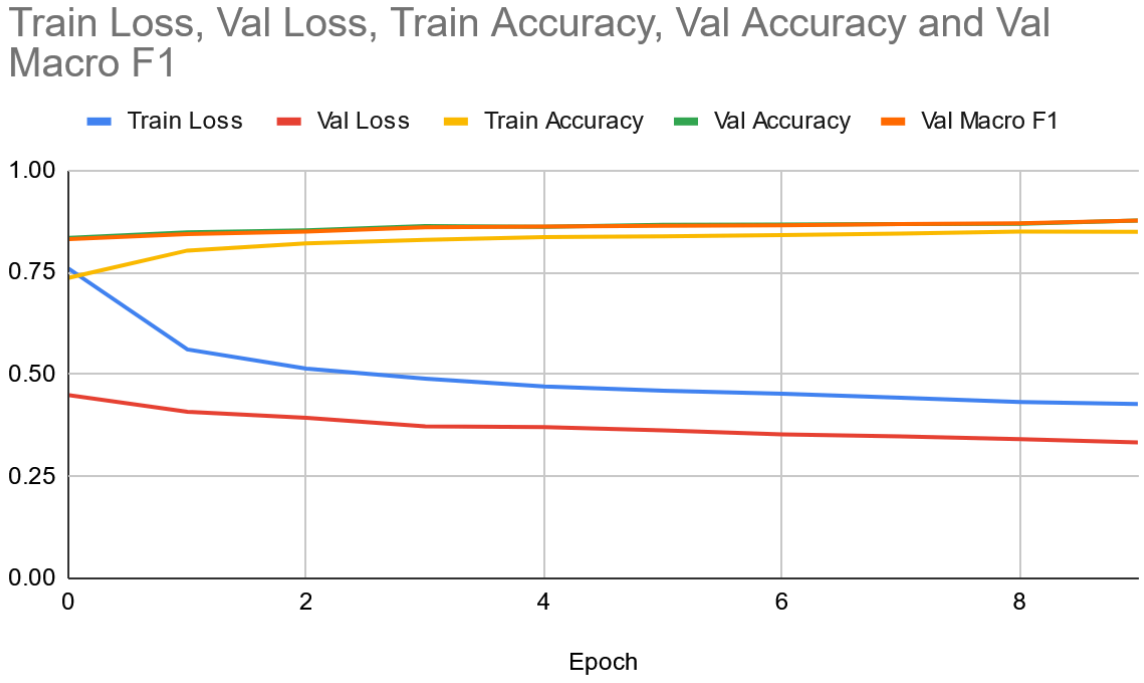

Train Loss, Val Loss, Train Accuracy, Val Accuracy and Val Macro F1
Train Loss
Val Loss
Train Accuracy
Val Accuracy
Val Macro F1
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
Epoch

3.

```
layers = [
    DenseLayer(input_size, 128),
    BatchNormalization(128),
    ReLU(),
    Dropout(0.5),
    DenseLayer(128, 64),
    BatchNormalization(64),
    ReLU(),
    Dropout(0.5),
    DenseLayer(64, num_classes)
]
```

Learning Rate: 0.0025

Epoch	Train Loss	Val Loss	Train Accuracy	Val Accuracy	Val Macro F1
0	0.759425	0.449343	0.736521	0.834308	0.831341
1	0.561154	0.408559	0.803479	0.848429	0.843660
2	0.514086	0.393461	0.820958	0.853108	0.850067
3	0.489448	0.372833	0.829625	0.863302	0.860393
4	0.470661	0.370507	0.836292	0.861965	0.862337
5	0.459828	0.362702	0.838250	0.866811	0.863913
6	0.452615	0.352925	0.841667	0.867062	0.865226
7	0.442866	0.347750	0.844875	0.868483	0.868740
8	0.432259	0.341426	0.850125	0.869402	0.870296
9	0.427466	0.333093	0.849708	0.877423	0.877241

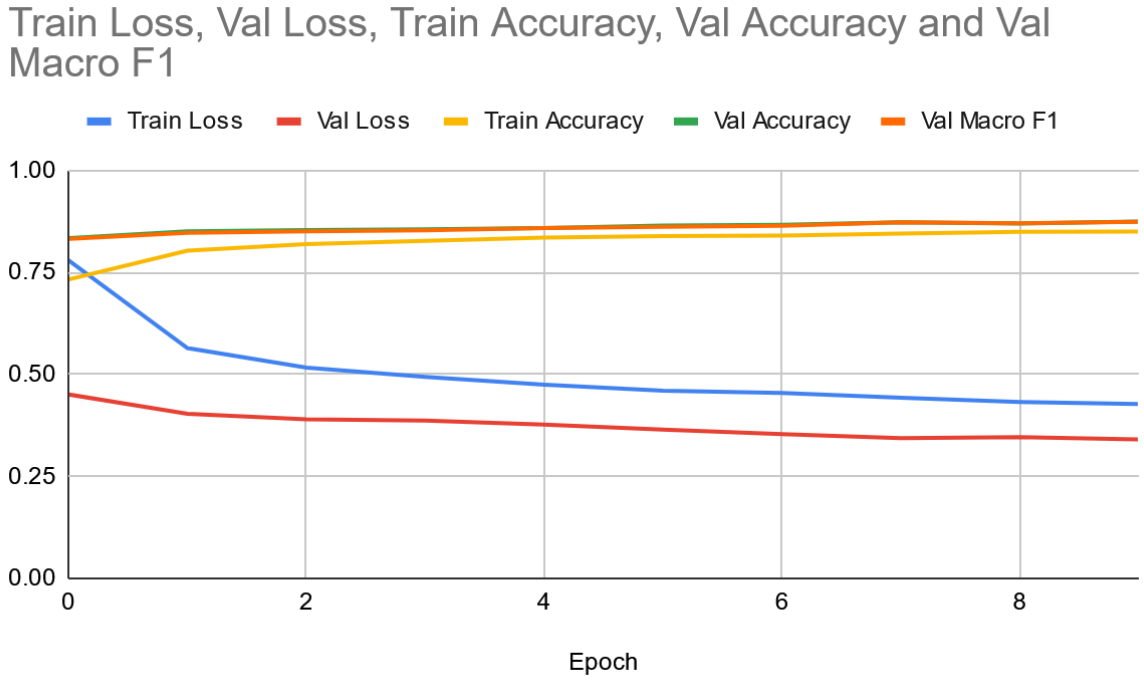

Train Loss, Val Loss, Train Accuracy, Val Accuracy and Val Macro F1
Train Loss
Val Loss
Train Accuracy
Val Accuracy
Val Macro F1
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
Epoch

4.

```
layers = [
    DenseLayer(input_size, 128),
    BatchNormalization(128),
    ReLU(),
    Dropout(0.5),
    DenseLayer(128, 64),
    BatchNormalization(64),
    ReLU(),
    Dropout(0.5),
    DenseLayer(64, num_classes)
]
```

Learning Rate: 0.002

Epoch	Train Loss	Val Loss	Train Accuracy	Val Accuracy	Val Macro F1
0	0.779764	0.450770	0.732479	0.833640	0.831946
1	0.564593	0.403225	0.803167	0.850852	0.847080
2	0.516615	0.389994	0.819479	0.853610	0.850473
3	0.493611	0.386971	0.827208	0.855866	0.852914
4	0.475022	0.376946	0.835313	0.858623	0.858663
5	0.460101	0.364784	0.838812	0.864555	0.861680
6	0.454618	0.353745	0.840354	0.866728	0.864414
7	0.442470	0.343406	0.844979	0.872660	0.872661
8	0.432121	0.346244	0.849500	0.869485	0.870388
9	0.427492	0.340773	0.849833	0.874833	0.873970

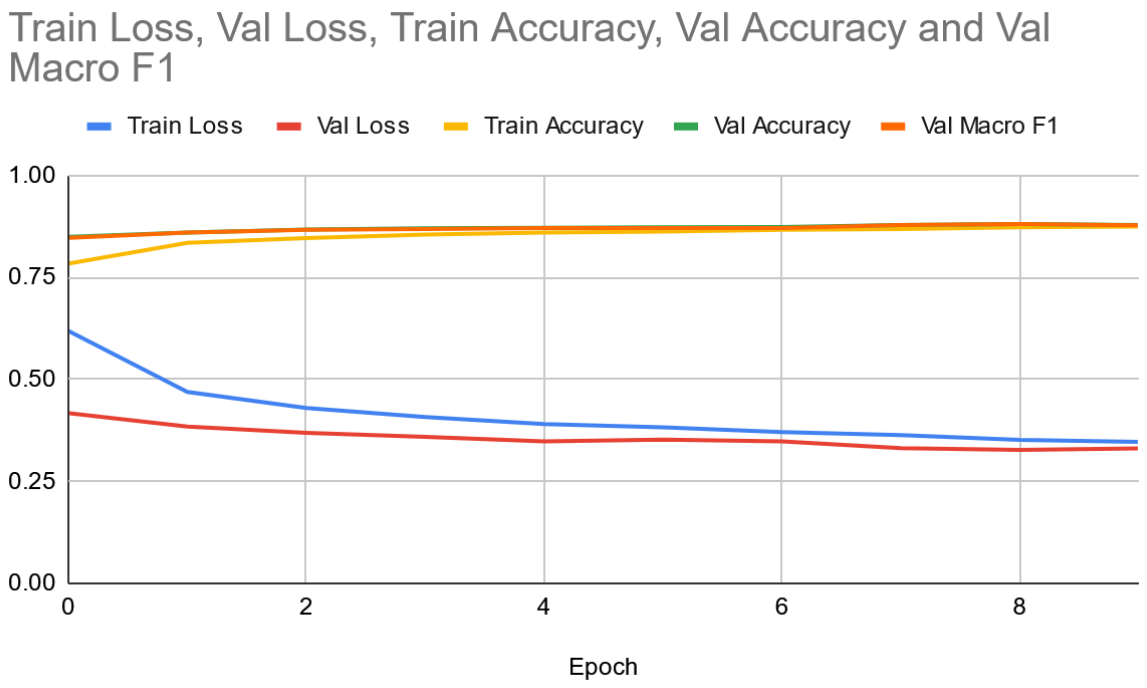

Train Loss, Val Loss, Train Accuracy, Val Accuracy and Val Macro F1
Train Loss
Val Loss
Train Accuracy
Val Accuracy
Val Macro F1
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
Epoch

5.

```
layers = [
    DenseLayer(input_size, 128),
    BatchNormalization(128),
    ReLU(),
    Dropout(0.5),
    DenseLayer(64, num_classes)
]
```

Learning Rate: 0.001

Epoch	Train Loss	Val Loss	Train Accuracy	Val Accuracy	Val Macro F1
0	0.618645	0.417051	0.783771	0.849348	0.846740
1	0.469250	0.384150	0.834583	0.859793	0.860008
2	0.430079	0.368685	0.846167	0.867396	0.866839
3	0.407786	0.359002	0.855021	0.870488	0.868791
4	0.390559	0.347832	0.859604	0.871324	0.870949
5	0.382606	0.352317	0.862125	0.872660	0.871117
6	0.370639	0.347708	0.866396	0.873329	0.870801
7	0.363593	0.331675	0.868708	0.878342	0.878572
8	0.351506	0.327246	0.872729	0.880515	0.880535
9	0.346436	0.330914	0.874521	0.878259	0.877936

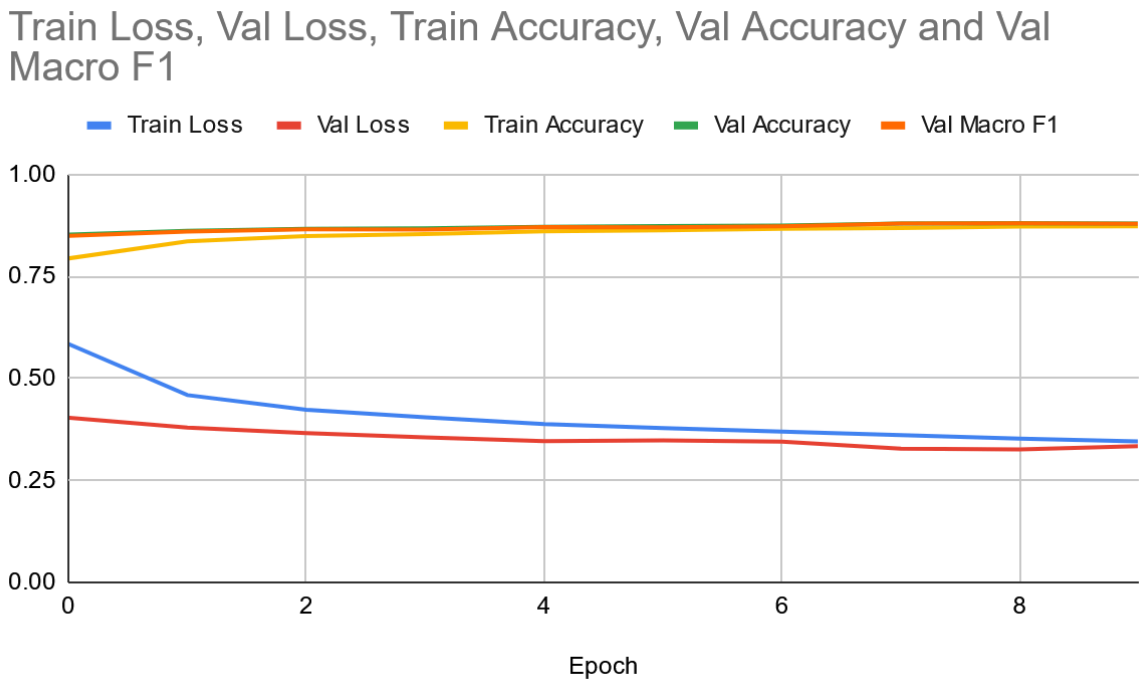


6.

```
layers = [
    DenseLayer(input_size, 128),
    BatchNormalization(128),
    ReLU(),
    Dropout(0.5),
    DenseLayer(64, num_classes)
]
```

Learning Rate: 0.002

Epoch	Train Loss	Val Loss	Train Accuracy	Val Accuracy	Val Macro F1
0	0.584217	0.403677	0.794063	0.852440	0.849475
1	0.458930	0.379230	0.836000	0.861715	0.859785
2	0.423048	0.365907	0.848729	0.866811	0.865253
3	0.404832	0.355130	0.853479	0.868148	0.864748
4	0.387915	0.346407	0.860292	0.871491	0.870923
5	0.377941	0.347919	0.862917	0.873580	0.870639
6	0.369571	0.345128	0.866896	0.874916	0.872571
7	0.360631	0.327838	0.868583	0.879345	0.879719
8	0.352080	0.326079	0.872375	0.880097	0.880077
9	0.345576	0.333854	0.872812	0.879846	0.878856

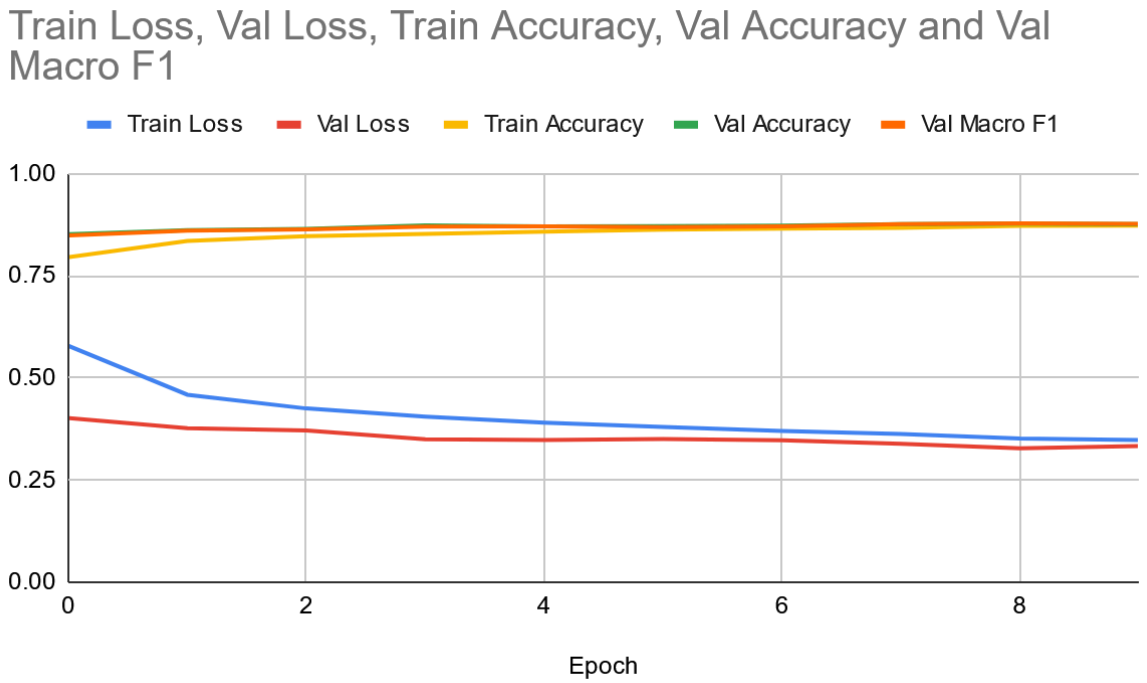


7.

```
layers = [
    DenseLayer(input_size, 128),
    BatchNormalization(128),
    ReLU(),
    Dropout(0.5),
    DenseLayer(64, num_classes)
]
```

Learning Rate: 0.0025

Epoch	Train Loss	Val Loss	Train Accuracy	Val Accuracy	Val Macro F1
0	0.578433	0.401508	0.795354	0.851939	0.848904
1	0.458935	0.377125	0.835063	0.861715	0.860687
2	0.425231	0.371589	0.846792	0.865057	0.863396
3	0.405403	0.349766	0.852542	0.873329	0.870176
4	0.390383	0.347914	0.858313	0.870989	0.870984
5	0.379760	0.350648	0.863104	0.871741	0.868955
6	0.370181	0.347561	0.865625	0.872828	0.870678
7	0.362706	0.338927	0.867396	0.876337	0.876392
8	0.351917	0.327460	0.872333	0.877924	0.878131
9	0.347758	0.333567	0.872646	0.876588	0.876936

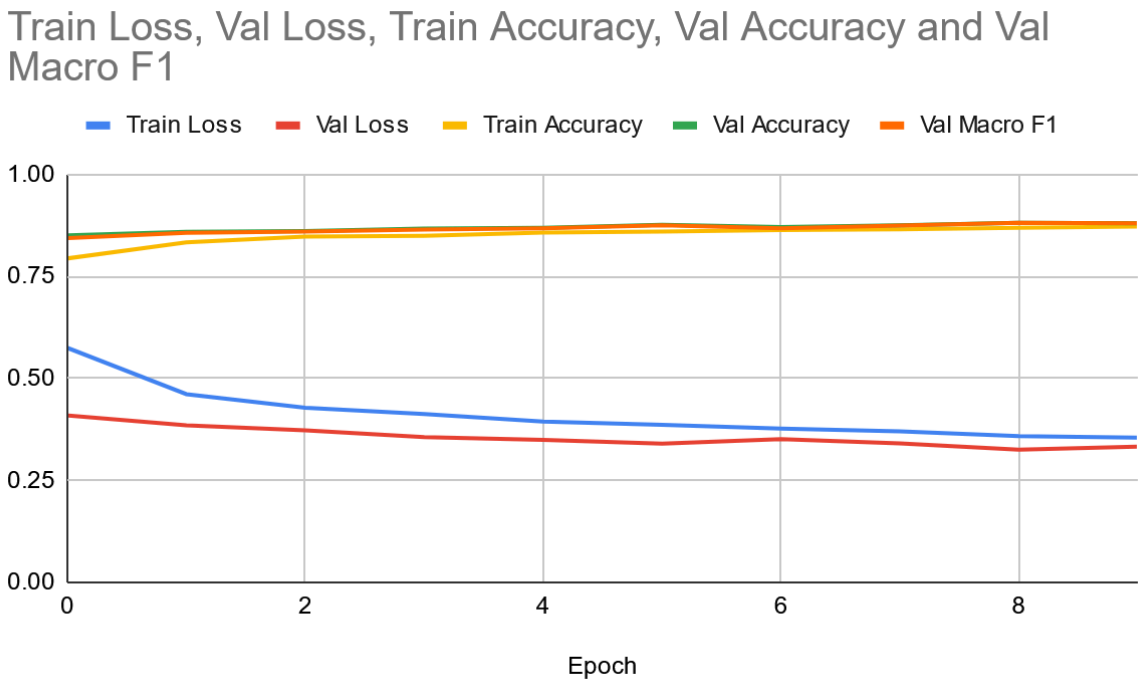


8.

```
layers = [
    DenseLayer(input_size, 128),
    BatchNormalization(128),
    ReLU(),
    Dropout(0.5),
    DenseLayer(64, num_classes)
]
```

Learning Rate: 0.005 (Best)

Epoch	Train Loss	Val Loss	Train Accuracy	Val Accuracy	Val Macro F1
0	0.574693	0.409061	0.794083	0.850518	0.844064
1	0.461391	0.384785	0.833313	0.859208	0.857004
2	0.427779	0.372836	0.847833	0.861380	0.860000
3	0.412311	0.356236	0.849583	0.867313	0.864962
4	0.394283	0.349244	0.857500	0.869235	0.867840
5	0.385842	0.340150	0.860167	0.876420	0.875119
6	0.376892	0.351366	0.863646	0.870989	0.868032
7	0.369884	0.340639	0.865521	0.875167	0.874712
8	0.358403	0.325087	0.869167	0.881350	0.881459
9	0.354898	0.332827	0.872188	0.880431	0.880274



We got best performance for learning rate = 0.005 and architecture:

```
layers = [
    DenseLayer(input_size, 128),
    BatchNormalization(128),
    ReLU(),
    Dropout(0.5),
    DenseLayer(64, num_classes)
]
```

Testing Accuracy: 0.8729
Test Macro-F1 Score: 0.8725546236397171