Reading assistive technology for the visually impaired: ASL to Braille conversion

Rohan Srinivasan
2020A7PS0081P
f20200081@pilani.bits-pilani.ac.in

Samriddha Sinha
2020A7PS0021P
f20200021@pilani.bits-pilani.ac.in

December 10, 2022

1 Problem Statement

While there have been strides to bridge the communication gap between the visually impaired and the sighted, and that between the hearing/speech impaired and others, there has not been a concerted effort to help visually and hearing impaired people communicate with each other. In this project, we aim to address this gap.

2 Proposed Solution

We have analyzed the model Sign Pose-based Transformer (SPOTER) proposed by [1]. We have performed the hyperparameter tuning beyond what is described in the paper. We have performed a learning rate range test and a grid search to find the optimal values of learning rate, weight decay and momentum. We have also generated the confusion matrix demonstrating the performance of the model. The code for the original model along with the changes we did for finding the right hyperparameters are included in the GitHub repository.

3 Introduction about SPOTER and WLASL dataset

We have used the WLASL[2] dataset to compare different models and techniques for converting ASL videos to text. American Sign Language is a standardized sign language. It consists of letter symbols and word symbols, a bit like Japanese hiragana (the phonetic alphabet) and kanji (whole words represented by a single symbol). Translation to English is context-based as a lot of the articles and prepositions used in spoken language are meant to be inferred in ASL.

WLASL stands for Word Level American Sign Language. It is a video dataset containing more than 2000 words performed by over 100 signers. Using a video dataset is important as most image datasets are collected by intentionally signing the words or letters; video datasets allow you the opportunity to use real-world data.

SPOTER proposes a sign-pose based transformer model. They converted the raw video data of the WLASL dataset into sign-pose encodings and fed this data into the transformer-based model. We have attempted to perform hyperparameter tuning on this model to find optimal hyperparameters for faster convergence and better generalisation. We have tried to find optimal hyperparameter combinations that would be optimal at low compute training. To be precise, we have tried to maximize the testing accuracy when trained for 15 epochs.

4 Choice of Hyperparameters

4.1 Learning Rate

The learning rate is one of the most important hyperparameters. It plays a crucial role in determining the rate of convergence. Keeping it low would result in more precise gradient descend steps, however, the speed of training would be abysmally low and would be expensive in terms of time and compute. Keeping it high results in wider descent steps. but the steps are less precise. The authors have used a learning rate of 1e-3. We attempt to find the maximum possible value of learning rate for which the model still converges. For this, we used the learning rate range test[3] as described in the next section on hyperparameter search experiments.

4.2 Momentum

Momentum is a technique used to accelerate the training process. It works on the principle of inertia. While making every step, the optimizer considers both the current gradient and the history of gradients. The proportion of weightage that is given to the current step and the history is determined by the values of momentum. A momentum of 0 implies that the history is not considered at all. The authors have used the value of momentum as 0, meaning they haven't used momentum. We include momentum and attempt to achieve faster convergence.

4.3 Weight Decay

Weight decay is a technique used to prevent overfitting. We introduce regularization by adding a penalty term to our loss function. Thus, it prevents the model from becoming influenced by the outliers in the data. The authors have not used weight decay (ie weight decay = 0). We attempt to investigate if including a decay term would lead to a better performance of our model.

5 Hyperparameter Search Experiments

For this section, we have largely employed the techniques described by Leslie Smith in [3].

5.1 Learning Rate Range Test

The heart of the learning rate range test is to start at a small learning rate and increase it gradually during the course of a run. We performed two such tests.

For the first range test, we started with a learning rate of 4e-10 and increased it by a factor of 4 every mini-run. As illustrated by the graph in Figure 1, we obtained a loss minima at the 12th iteration. This translates to a value close to 1.677e-3.

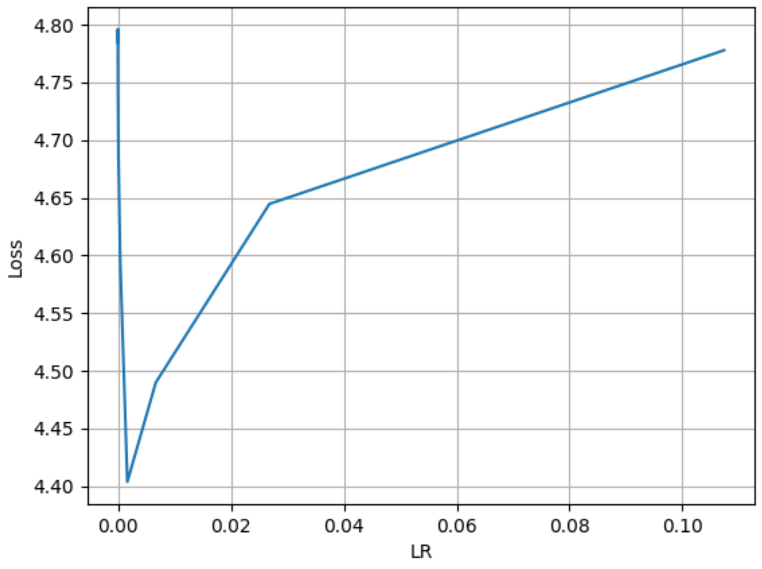


Figure 1: First learning rate range test

To find the most suitable learning rate within this order of magnitude, we performed another learning rate range test. We started at 0.5e-3 and added 0.5e-3 to the learning rate on each mini-run. We found the greatest reduction in loss at **2e-3** as seen in Figure 2. Hence we decided to carry forward this value of learning rate in the processes further.

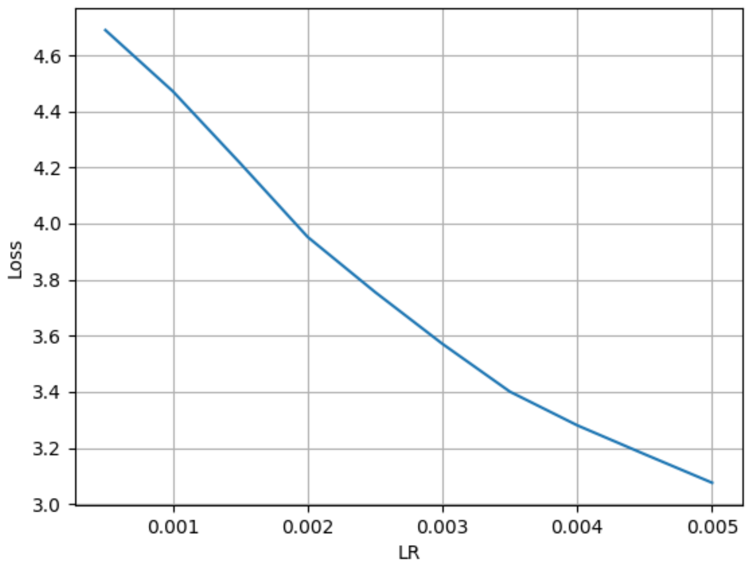


Figure 2: Second learning rate range test

5.2 Grid Search

For momentum and weight decay values, we performed a grid search over the search space. We search for momentum values 0.99, 0.97, 0.95, 0.9, 0.8, 0.7, 0.5, 0.2, 0.1, 0.01 and weight values 1e-3, 1e-4, 1e-5 and 0. We ran each combination for 15 epochs and compared the results in Table 1.

		Weight Decay			
		0.001	0.0001	0.00001	0
	0.99	0.015504	0.015504	0.015504	0.019380
	0.97	0.023256	0.015504	0.015504	0.019380
Momentum	0.95	0.023256	0.015504	0.015504	0.015504
	0.9	0.085271	0.015504	0.038760	0.015504
	0.7	0.224806	0.201550	0.186047	0.232558
	0.5	0.29845	0.290698	0.263566	0.271318
	0.2	0.352713	0.337209	0.313953	0.317829
	0.1	0.348837	**0.356589**	0.348837	**0.356589**
	0.01	0.348837	0.348837	0.341085	0.337209
	0	0.348837	0.344961	0.348837	0.348837

Table 1: Grid Search

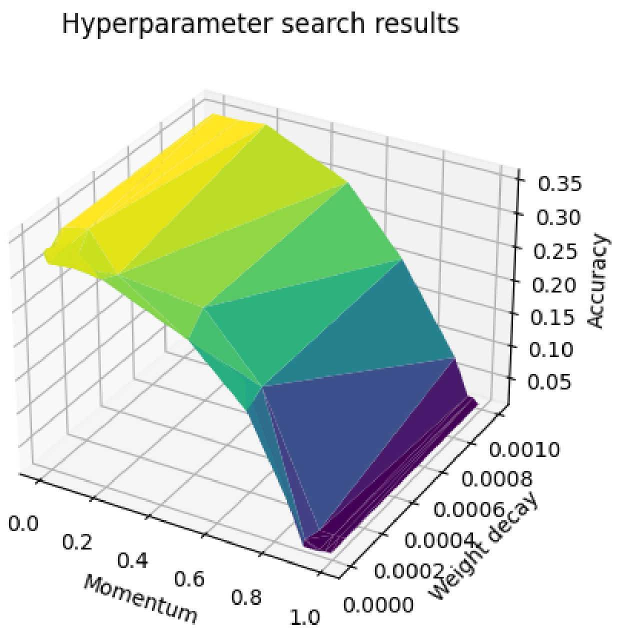


Figure 3: Hyperparameter Search Visualisation

We found two combinations of momentum and weight decay to give the same highest test accuracy of **0.3566** at 15 epochs.

5.3 Confusion Matrix

A confusion matrix is a table used to evaluate a classification algorithm's performance. Given below is the confusion matrix for one of the best combinations of hyperparameters. Here, the matrix has been plotted for the values of learning rate 2e-3, momentum 0.1 and weight decay 1e-4, when trained for 15 epochs. For a higher resolution image, click here.

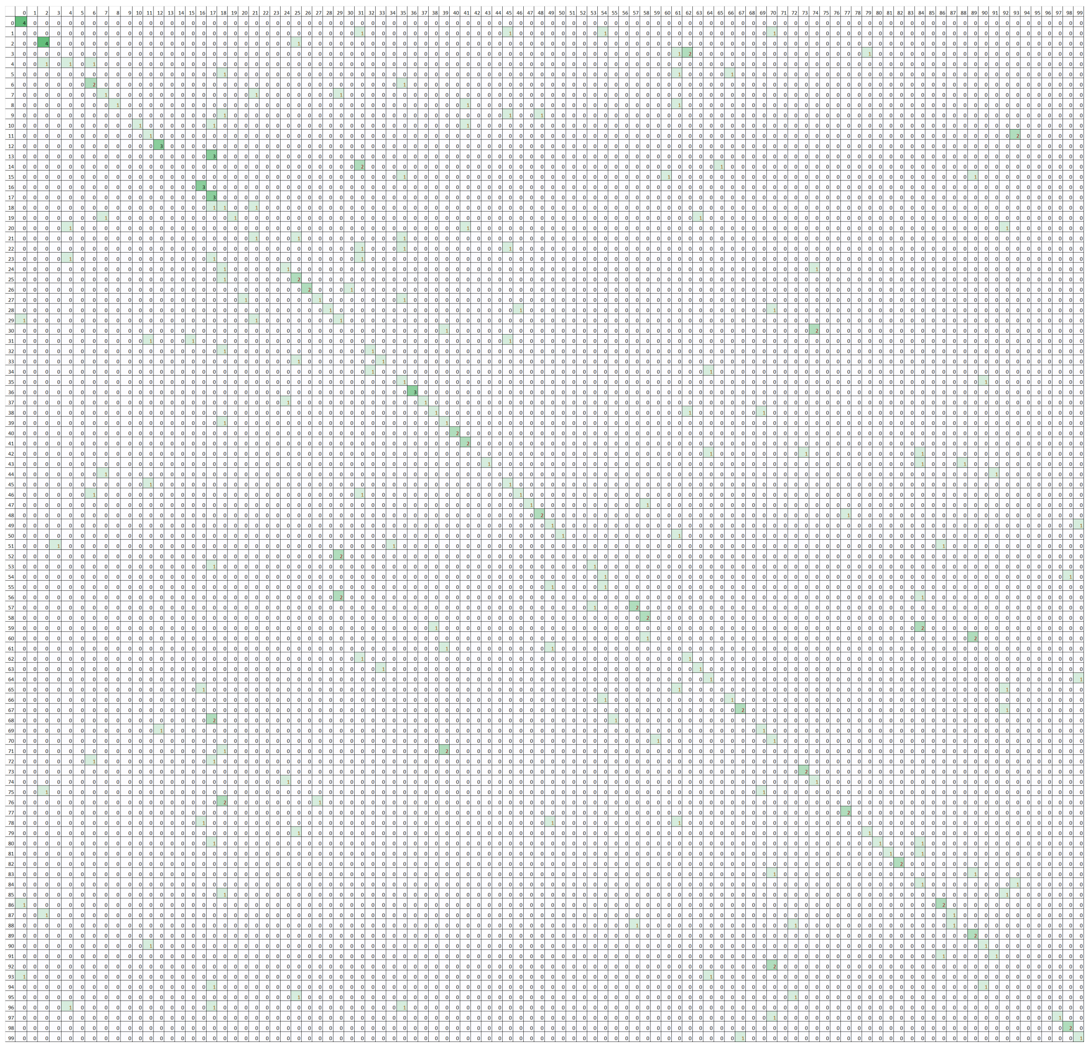


Figure 4: Confusion Matrix

As evidenced here, the bulk of the colour in the image is along the principal diagonal which conveys that the model made correct predictions for those values. There are a few systemic mistakes such as never predicting the output as label 13, but predicting the output as label 17 for all the instances of images of label 13 in the dataset. Thus the model does seem to have learned that images of this label belong together, but still ends up confusing it for label 17. This might be because of the small size of our training set and our training of the model for a much lower number of epochs as compared to the original paper. We can also look at the class-wise precision, recall and F-1 values as calculated in the Appendix in Figure 7 for the same run to draw further insights.

6 Results

We found the best combination of hyperparameters to be

- Learning Rate: 2e-3
- Momentum: 0.1

- Weight Decay: 1e-4 or 0

These combinations worked better on the WLASL100 dataset than the combination used by the paper (learning rate 1e-3, momentum and weight decay 0) after 15 epochs as seen in Table 2.

Experiment Name	Epochs	Learning Rate	Momentum	Weight Decay	Testing Accuracy
baseline-model	15	$1.00E-03$	0	0	0.341085
hp-search-0-0	15	$2.00E-03$	0	0	0.348837
hp-search-0.1-0.0001	15	$2.00E-03$	0.1	$1.00E-04$	**0.356589**
hp-search-0.1-0	15	$2.00E-03$	0.1	0	**0.356589**

Table 2: Comparision of our best models with the baseline

7 Conclusion

Through this report, we have demonstrated the usefulness of SPOTER technology and the importance on hyperparameter tuning. We have achieved a better result than the original paper when run in a low compute environment, which might be the case with low-powered hardware converting ASL to Braille in the real world.

If we look at Figure 5, we can see that even at the end of 15 epochs, the loss is still decreasing and hasn't converged yet. So we postulate that we would be able to reach comparable results to the original paper in fewer epochs than the 350 used in the original paper.

References

[1] M. Boháček and M. Hrúz, "Sign pose-based transformer for word-level sign language recognition," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2022, pp. 182–191. DOI: 10.1109/WACVW54805.2022.00024.

[2] D. Li, C. Rodriguez, X. Yu, and H. Li, "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 1459–1469.

[3] L. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay," Mar. 2018.

Appendix

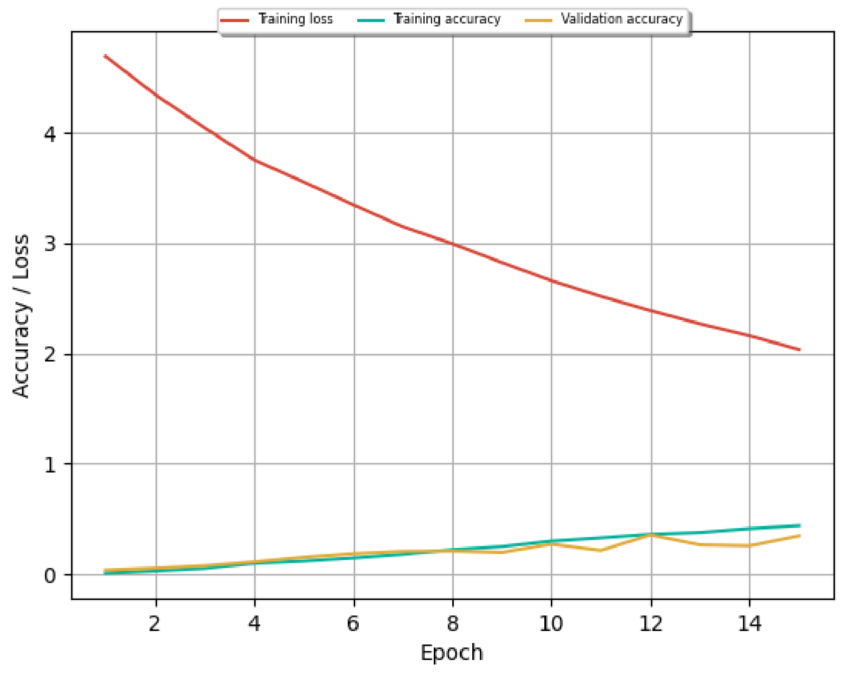


Figure 5: Loss reduction at every epoch on the best run

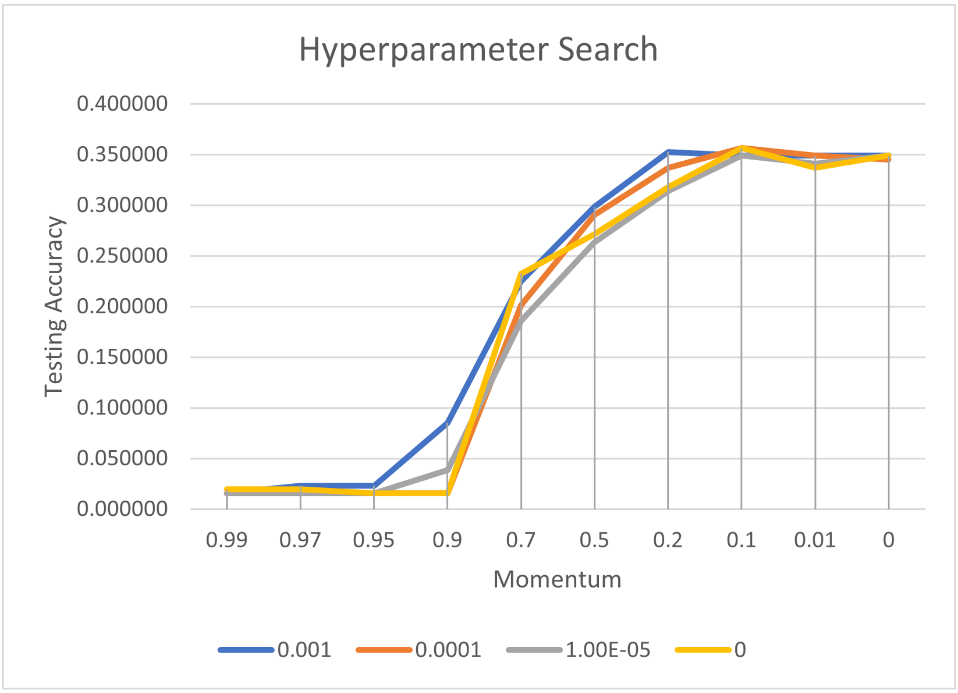
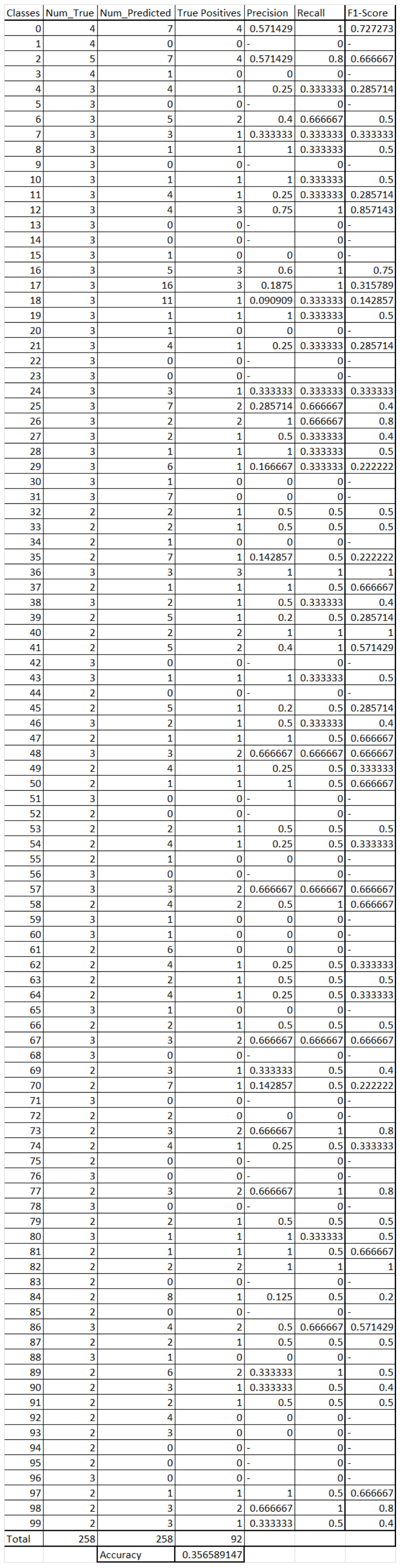


Figure 6: Momentum and Weight Decay vs Testing Accuracy



Classes	Num_True	Num_Predicted	True Positives	Precision	Recall	F1-Score
0	4	7	4	0.571429	1	0.727273
1	4	0	0	-	0	-
2	5	7	4	0.571429	0.8	0.666667
3	4	1	0	0	0	-
4	3	4	1	0.25	0.333333	0.285714
5	3	0	0	-	0	-
6	3	5	2	0.4	0.666667	0.5
7	3	3	1	0.333333	0.333333	0.333333
8	3	1	1	1	0.333333	0.5
9	3	0	0	-	0	-
10	3	1	1	1	0.333333	0.5
11	3	4	1	0.25	0.333333	0.285714
12	3	4	3	0.75	1	0.857143
13	3	0	0	-	0	-
14	3	0	0	-	0	-
15	3	1	0	0	0	-
16	3	5	3	0.6	1	0.75
17	3	16	3	0.1875	1	0.315789
18	3	11	1	0.090909	0.333333	0.142857
19	3	1	1	1	0.333333	0.5
20	3	1	0	0	0	-
21	3	4	1	0.25	0.333333	0.285714
22	3	0	0	-	0	-
23	3	0	0	-	0	-
24	3	3	1	0.333333	0.333333	0.333333
25	3	7	2	0.285714	0.666667	0.4
26	3	2	2	1	0.666667	0.8
27	3	2	1	0.5	0.333333	0.4
28	3	1	1	1	0.333333	0.5
29	3	6	1	0.166667	0.333333	0.222222
30	3	1	0	0	0	-
31	3	7	0	0	0	-
32	2	2	1	0.5	0.5	0.5
33	2	2	1	0.5	0.5	0.5
34	2	1	0	0	0	-
35	2	7	1	0.142857	0.5	0.222222
36	3	3	3	1	1	1
37	2	1	1	1	0.5	0.666667
38	3	2	1	0.5	0.333333	0.4
39	2	5	1	0.2	0.5	0.285714
40	2	2	2	1	1	1
41	2	5	2	0.4	1	0.571429
42	3	0	0	-	0	-
43	3	1	1	1	0.333333	0.5
44	2	0	0	-	0	-
45	2	5	1	0.2	0.5	0.285714
46	3	2	1	0.5	0.333333	0.4
47	2	1	1	1	0.5	0.666667
48	3	3	2	0.666667	0.666667	0.666667
49	2	4	1	0.25	0.5	0.333333
50	2	1	1	1	0.5	0.666667
51	3	0	0	-	0	-
52	2	0	0	-	0	-
53	2	2	1	0.5	0.5	0.5
54	2	4	1	0.25	0.5	0.333333
55	2	1	0	0	0	-
56	3	0	0	-	0	-
57	3	3	2	0.666667	0.666667	0.666667
58	2	4	2	0.5	1	0.666667
59	3	1	0	0	0	-
60	3	1	0	0	0	-
61	2	6	0	0	0	-
62	2	4	1	0.25	0.5	0.333333
63	2	2	1	0.5	0.5	0.5
64	2	4	1	0.25	0.5	0.333333
65	3	1	0	0	0	-
66	2	2	1	0.5	0.5	0.5
67	3	3	2	0.666667	0.666667	0.666667
68	3	0	0	-	0	-
69	2	3	1	0.333333	0.5	0.4
70	2	7	1	0.142857	0.5	0.222222
71	3	0	0	-	0	-
72	2	2	0	0	0	-
73	2	3	2	0.666667	1	0.8
74	2	4	1	0.25	0.5	0.333333
75	2	0	0	-	0	-
76	3	0	0	-	0	-
77	2	3	2	0.666667	1	0.8
78	3	0	0	-	0	-
79	2	2	1	0.5	0.5	0.5
80	3	1	1	1	0.333333	0.5
81	2	1	1	1	0.5	0.666667
82	2	2	2	1	1	1
83	2	0	0	-	0	-
84	2	8	1	0.125	0.5	0.2
85	2	0	0	-	0	-
86	3	4	2	0.5	0.666667	0.571429
87	2	2	1	0.5	0.5	0.5
88	3	1	0	0	0	-
89	2	6	2	0.333333	1	0.5
90	2	3	1	0.333333	0.5	0.4
91	2	2	1	0.5	0.5	0.5
92	2	4	0	0	0	-
93	2	3	0	0	0	-
94	2	0	0	-	0	-
95	2	0	0	-	0	-
96	3	0	0	-	0	-
97	2	1	1	1	0.5	0.666667
98	2	3	2	0.666667	1	0.8
99	2	3	1	0.333333	0.5	0.4
Total	258	258	92			
		Accuracy	0.356589147			

Figure 7: Precision, Recall and F1 values, full resolution picture here