DAFTAR PUSTAKA

Adiwibowo, J., Gunadi, K., & Setyati, E. (2020). Deteksi Alat Pelindung Diri Menggunakan Metode YOLO dan Faster R-CNN. *Jurnal Infra*, *8*(2).

Anam, S., Nashihin, H., Taufik, A., Mubarok, Sitompul, H. S., Manik, Y. M., Suparto, Arsid, I., Jumini, S., Nurhab, M. I., Solehudin, W, N. E., & Luturmas, Y. (2023). *Metode Penelitian (Kualitatif, Kuantitatif, Eksperimen, dan R&D)*. Global Eksekutif Teknologi.

Bayram, M. T. (2023). *PPE Detection* [Dataset]. https://www.kaggle.com/datasets/mustafatayyipbayram/ppe-detection.

Bhake, D. (2023). *PPE Dataset 3* [Dataset]. https://www.kaggle.com/datasets/devashishbhake01/ppe-dataset-3.

Djaohar, M., Sunarwan, A., & Dannys. (2022). Rancang Bangun Pengecekan Alat Pelindung Diri Menggunakan Algoritma You Only Look Once (YOLO). *Journal of Electrical Vocational Education and Technology*, *7*(1).

Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022). A Review of Yolo Algorithm Developments [Paper presentation]. *The 8th International Conference on Information Technology and Quantitative Management*.

Laily, M. E., Fajri, F. N., & Pratamasunu, G. Q. O. (2022). Deteksi Penggunaan Alat Pelindung Diri (APD) Untuk Keselamatan dan Kesehatan Kerja Menggunakan Metode Mask Region Convolutional Neural Network (Mask R-CNN). *Jurnal Politeknik Caltex Riau*, *8*(2), 279-288.

Mafra, R., Riduan, & Zulfikri. (2021). Analisis Kebutuhan Penggunaan Alat Pelindung Diri (APD) Pada Peserta Pelatihan Keterampilan Tukang dan Pekerja Konstruksi. *Jurnal Arsir Muhammadiyah Palembang*, *5*, 48-63.

Mailoa, R. M., & Santoso, L. W. (2020). Deteksi Rompi dan Helm Keselamatan Menggunakan Metode YOLO dan CNN. *Jurnal Infra*.

Nirvana, M. N., Rachmadi, R. F., & Purnama, K. E. (2023). *Sistem Pendeteksi Alat Pelindung Diri (APD) Pada Pekerja Konstruksi Berbasis Convolutional Neural Network* [Undergraduate thesis, Institut Teknologi Sepuluh November]. ITS Repository. https://repository.its.ac.id/101058.

Nurfirmansyah, A., & Dijaya, R. (2022). Deteksi Kelalaian Alat Pelindung Diri (APD) pada Pekerja Konstruksi Bangunan [Paper presentation]. *Seminar Nasional Inovasi Teknologi*.

Paredes, B. R., & Torr, P. H. S. (2016). Recurrent Instance Segmentation [Paper presentation]. *14th European Conference on Computer Vision*, 312-329.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection [Paper presentation]. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Roboflow. (2023). *Helomet.v2 Computer Vision Project* [Dataset]. https://universe.roboflow.com/x-camp/helomet.v2.

Roboflow (2023). *Personal Protective Equipment* [Dataset]. https://universe.roboflow.com/roboflow-universe-projects/personal-protective-equipment-combined-model.

Roboflow. (2022). *PPE detection Computer Vision Project* [Dataset]. https://universe.roboflow.com/augmented-startups/ppe-detection-qhryg.

Schröer, C., Kruse, F., & Gómez, J. M. (2020). A Systematic Literature Review on Applying CRISP-DM Process Model [Paper presentation]. *International Conference on Enterprise Information Systems.*

Stevens, E., Antiga, L., & Viehmann, T. (2020). *Deep Learning with PyTorch: Build, Train, and Tune Neural Networks Using Python Tools*. Manning.

Terven, J. R., & Cordova-Esparza, D. M. (2023). A Comprehensive Review of YOLO: from YOLOV1 and Beyond. *Journal of Machine Learning and Knowledge Extraction*.

Unair News. (2019). Perusahaan Perlu Tekankan Pentingnya Gunakan Pelindung Diri untuk Keselamatan Pekerja. *UNAIR News*. https://news.unair.ac.id/2019/09/18/perusahaan-perlu-tekankan-pentingnya-gunakan-pelindung-diri-untuk-keselamatan-pekerja.

Ultralytics. (2023). Ultralytics YOLOv8 Docs. https://docs.ultralytics.com/.

Wiranda, N., Purba, H. S., & Sukmawati, R. A. (2020). Survei Penggunaan Tensorflow pada Machine Learning untuk Identifikasi Ikan Kawasan Lahan Basah. *Indonesian Journal of Electronics and Instrumentations Systems*, *10*(2).

Zhao, Z.-Q., Zheng, P., Xu, S.-T., & Wu, X. (2019). Object Detection with Deep Learning: A Review [Paper presentation]. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 1-21

LAMPIRAN

Lampiran 1. *Training log* model YOLOv8 *object detection*

Epoch	*Train*			*Val*		
	box_loss	***cls_loss***	***dfl_loss***	***box_loss***	***cls_loss***	***dfl_loss***
1	1,5805	3,0674	1,5018	1,4672	2,1978	1,5503
2	1,5532	2,1567	1,4887	1,3416	2,0054	1,547
3	1,5407	2,1003	1,4779	1,3633	1,8316	1,5472
4	1,5234	2,0574	1,4654	1,3114	1,7216	1,4459
5	1,5056	1,9023	1,4637	1,3045	1,5655	1,3942
6	1,4924	1,7864	1,4614	1,3119	1,5008	1,3997
7	1,4997	1,6966	1,4397	1,2535	1,3612	1,3446
8	1,4795	1,6216	1,4216	1,2447	1,2863	1,3437
9	1,4599	1,5656	1,4086	1,2162	1,1943	1,3173
10	1,4514	1,5137	1,4000	1,2148	1,1696	1,3306
11	1,4355	1,4792	1,3892	1,2270	1,1918	1,3133
12	1,4304	1,4501	1,3825	1,2001	1,1350	1,2904
13	1,4196	1,4155	1,3748	1,2100	1,1372	1,2986
14	1,4029	1,3907	1,3635	1,1700	1,0857	1,264
15	1,3950	1,3683	1,3598	1,1623	1,0450	1,2715
16	1,3883	1,3382	1,3486	1,1781	1,0773	1,2839
17	1,3818	1,3350	1,3502	1,1553	0,9885	1,2548
18	1,3728	1,2988	1,3406	1,1495	0,9775	1,2536
19	1,3705	1,2785	1,3360	1,1411	0,9585	1,2339
20	1,3600	1,2557	1,3299	1,1430	0,9211	1,2407
21	1,3521	1,2389	1,3211	1,1306	0,9356	1,2281
22	1,3508	1,2352	1,3193	1,1261	0,9126	1,2327
23	1,3455	1,2185	1,3185	1,1282	0,9230	1,2288
24	1,3431	1,2090	1,3111	1,1202	0,8572	1,2159
25	1,3308	1,1885	1,3092	1,1338	0,8746	1,2218
26	1,3294	1,1802	1,3026	1,1129	0,8751	1,2211
27	1,3264	1,1682	1,2992	1,1129	0,8730	1,2192
28	1,3172	1,1607	1,2985	1,1055	0,8404	1,2037
29	1,3121	1,1440	1,2920	1,0912	0,8202	1,1962
30	1,3078	1,1305	1,2872	1,0877	0,8225	1,1923
31	1,3012	1,1216	1,2806	1,0953	0,8178	1,2006
32	1,3039	1,1117	1,2821	1,0876	0,8056	1,1918
33	1,2960	1,1081	1,2775	1,0814	0,8202	1,192
34	1,2960	1,1009	1,2761	1,0754	0,8130	1,1777
35	1,2923	1,0885	1,2753	1,0832	0,7928	1,1811
36	1,2789	1,0809	1,2671	1,0830	0,7766	1,1871

Lanjutan Lampiran 1

Epoch	Train			Val		
	box_loss	cls_loss	dfl_loss	box_loss	cls_loss	dfl_loss
37	1,2802	1,0730	1,2666	1,0814	0,7910	1,1812
38	1,2719	1,0607	1,2581	1,0720	0,7824	1,1787
39	1,2714	1,0611	1,2640	1,0698	0,7707	1,1814
40	1,2701	1,0487	1,2611	1,0531	0,7677	1,1683
41	1,2701	1,0448	1,2597	1,0763	0,7573	1,1792
42	1,2556	1,0204	1,2504	1,0503	0,7456	1,149
43	1,2599	1,0209	1,2477	1,0436	0,7478	1,1464
44	1,2497	1,0144	1,2507	1,0609	0,7348	1,1626
45	1,2491	1,0080	1,2445	1,0531	0,7344	1,1567
46	1,2486	1,0059	1,2438	1,0492	0,7317	1,1556
47	1,2409	1,0007	1,2439	1,0449	0,7202	1,1441
48	1,2390	0,9921	1,2387	1,0437	0,7173	1,1446
49	1,2359	0,9827	1,2350	1,0375	0,7138	1,1441
50	1,2287	0,9785	1,2283	1,0380	0,7207	1,1395
51	1,2329	0,9743	1,2299	1,0367	0,7182	1,1376
52	1,2303	0,9715	1,2306	1,0400	0,7044	1,1442
53	1,2293	0,9702	1,2305	1,0333	0,7027	1,1383
54	1,2243	0,9643	1,2278	1,0272	0,6960	1,1354
55	1,2141	0,9563	1,2214	1,0369	0,6973	1,1408
56	1,2144	0,9467	1,2220	1,0343	0,6920	1,1391
57	1,2103	0,9411	1,2148	1,0345	0,6854	1,1391
58	1,2109	0,9407	1,2217	1,0323	0,6862	1,1396
59	1,2094	0,9352	1,2176	1,0318	0,6824	1,1401
60	1,2126	0,9366	1,2193	1,0297	0,6822	1,1336
61	1,1909	0,9037	1,2078	1,0388	0,7055	1,1504
62	1,2006	0,9142	1,2061	1,0361	0,7026	1,1485
63	1,1916	0,9121	1,2096	1,0429	0,7016	1,1554
64	1,1922	0,9072	1,2051	1,0359	0,7032	1,1491
65	1,1834	0,9030	1,2041	1,0439	0,7029	1,1524
66	1,1823	0,8965	1,2008	1,0375	0,6984	1,1514
67	1,1821	0,8903	1,1987	1,0299	0,6880	1,1446
68	1,1759	0,8875	1,1938	1,0321	0,6892	1,1441
69	1,1770	0,8828	1,1942	1,0384	0,6848	1,1469
70	1,1761	0,8785	1,1939	1,0418	0,6772	1,1521
71	1,1734	0,8718	1,1946	1,0442	0,6778	1,1563
72	1,1709	0,8742	1,1902	1,0371	0,6777	1,1525
73	1,1632	0,8713	1,1876	1,0386	0,6812	1,1535
74	1,1643	0,8631	1,1861	1,0389	0,6811	1,1544

Lanjutan Lampiran 1

Epoch	*Train*			*Val*		
	box_loss	***cls_loss***	***dfl_loss***	***box_loss***	***cls_loss***	***dfl_loss***
75	1,1591	0,8599	1,1834	1,0418	0,6817	1,1583
76	1,1596	0,8505	1,1828	1,0411	0,6795	1,1589
77	1,1524	0,8517	1,1811	1,0366	0,6758	1,1548
78	1,1421	0,8411	1,1758	1,0350	0,6760	1,1524
79	1,1452	0,8472	1,1709	1,0372	0,6774	1,1528
80	1,1497	0,8350	1,1693	1,0361	0,6761	1,1521
81	1,1422	0,8257	1,1687	1,0339	0,6747	1,1507
82	1,1349	0,8264	1,1624	1,0342	0,6747	1,1505
83	1,1368	0,8271	1,1601	1,0348	0,6763	1,1506
84	1,1388	0,8265	1,1612	1,0342	0,6782	1,1503
85	1,1922	0,9072	1,2051	1,0359	0,7032	1,1491
86	1,1309	0,8235	1,1602	1,0355	0,6797	1,151
87	1,1318	0,8180	1,1688	1,0360	0,6795	1,1514
88	1,1273	0,8127	1,1646	1,0362	0,6788	1,1514
89	1,1239	0,8109	1,1618	1,0358	0,6794	1,1509
90	1,1276	0,8077	1,1632	1,0358	0,6778	1,1509
91	1,1251	0,8052	1,1620	1,0361	0,6762	1,1511
92	1,1243	0,8069	1,1624	1,0365	0,6743	1,152
93	1,1236	0,8060	1,1615	1,0367	0,6752	1,1524
94	1,1172	0,8019	1,1603	1,0372	0,6743	1,153
95	1,1186	0,7972	1,1513	1,0376	0,6742	1,1534
96	1,1165	0,7984	1,1507	1,0368	0,6741	1,1526
97	1,1086	0,7800	1,1502	1,0369	0,6726	1,1531
98	1,1156	0,7704	1,1434	1,0363	0,6725	1,1527
99	1,1164	0,7719	1,1409	1,0361	0,6718	1,1526
100	1,1131	0,7626	1,1407	1,0362	0,6723	1,1526
101	1,1100	0,7429	1,1373	1,0375	0,6727	1,1534
102	1,0953	0,7345	1,1335	1,0368	0,6732	1,1531
103	1,0832	0,7496	1,1325	1,0373	0,6726	1,1534
104	1,0776	0,7537	1,1307	1,0373	0,6717	1,1538
105	1,0817	0,7551	1,1381	1,0369	0,6713	1,1532
106	1,0844	0,7568	1,1369	1,0366	0,6707	1,1529
107	1,0787	0,7557	1,1336	1,0365	0,6706	1,1526
108	1,0822	0,7522	1,1357	1,0362	0,6716	1,1527
109	1,0805	0,7501	1,1348	1,0362	0,6710	1,1524
110	1,0880	0,7511	1,1337	1,0362	0,6710	1,1529
111	1,0829	0,7500	1,1326	1,0364	0,6708	1,1531
112	1,0818	0,7486	1,1313	1,0360	0,6706	1,153

Lanjutan Lampiran 1

Epoch	*Train*			*Val*		
	box_loss	***cls_loss***	***dfl_loss***	***box_loss***	***cls_loss***	***dfl_loss***
113	1,0832	0,7403	1,1309	1,0361	0,6702	1,1531
114	1,0837	0,7424	1,1296	1,0361	0,6698	1,1524
115	1,0794	0,7403	1,1283	1,0360	0,6694	1,1516
116	1,0793	0,7484	1,1271	1,0364	0,6692	1,1505
117	1,0711	0,7477	1,1260	1,0366	0,6684	1,1477
118	1,0622	0,7392	1,1243	1,0372	0,6683	1,1441
119	1,0602	0,7346	1,1219	1,0376	0,6683	1,1432
120	1,0526	0,7206	1,1208	1,0378	0,6676	1,1405
121	1,0457	0,7162	1,1140	1,0278	0,6502	1,1364
122	1,0428	0,7021	1,1174	1,0279	0,6501	1,1366
123	1,0489	0,7052	1,1161	1,0280	0,6503	1,1368
125	1,0471	0,7122	1,1141	1,0287	0,6512	1,1374
126	1,0432	0,7110	1,1113	1,0289	0,6517	1,1378
127	1,0464	0,7089	1,1103	1,0293	0,6519	1,1379
128	1,0448	0,7068	1,1189	1,0293	0,6524	1,138
129	1,0433	0,7076	1,1181	1,0297	0,6524	1,1382
130	1,0471	0,7076	1,1199	1,0298	0,6522	1,1384
131	1,0486	0,7060	1,1111	1,0203	0,6520	1,1389
132	1,0482	0,7091	1,1106	1,0204	0,6522	1,1397
133	1,0472	0,7113	1,1112	1,0206	0,6526	1,1394
134	1,0476	0,7087	1,1106	1,0231	0,6522	1,1394
135	1,0473	0,7078	1,1102	1,0232	0,6525	1,1399
136	1,0402	0,7077	1,1100	1,0232	0,6516	1,1383
137	1,0400	0,6911	1,1091	1,0203	0,6518	1,1377
138	1,0392	0,6875	1,1063	1,0203	0,6521	1,1361
139	1,0371	0,6799	1,1029	1,0200	0,6521	1,1361
140	1,0327	0,6508	1,1081	1,0190	0,6528	1,1361
141	1,0296	0,6264	1,1059	1,0135	0,6524	1,1331
142	1,0233	0,6110	1,1037	1,0134	0,6522	1,1331
143	1,0241	0,6070	1,1022	1,0134	0,6521	1,1321
144	1,0193	0,6015	1,1005	1,0134	0,6522	1,1321
145	1,0125	0,5961	1,1001	1,0134	0,6520	1,1316
146	1,0104	0,5930	1,0997	1,0133	0,6518	1,1316
147	1,0117	0,5881	1,0970	1,0133	0,6513	1,1309
148	1,0080	0,5865	1,0960	1,0133	0,6501	1,1303
149	1,0033	0,5835	1,0948	1,0133	0,6501	1,1303
150	1,0010	0,5794	1,0934	1,0132	0,6500	1,1301

Lampiran 2. *Training log* model YOLOv8 *instance segmentation*

Epoch	*Train*			*Val*		
	seg_loss	***cls_loss***	***dfl_loss***	***seg_loss***	***cls_loss***	***dfl_loss***
1	2,6141	3,2133	1,2923	2,6394	2,2389	1,4263
2	2,6038	2,2174	1,2815	2,5770	2,1941	1,4029
3	2,5463	2,1257	1,2621	2,4517	2,0075	1,3528
4	2,5163	2,0916	1,2615	2,4303	1,8723	1,3441
5	2,4938	1,9591	1,2306	2,3424	1,6811	1,3407
6	2,4345	1,8428	1,2271	2,2989	1,5570	1,3411
7	2,4266	1,7715	1,2250	2,2905	1,4928	1,2857
8	2,4177	1,6994	1,2240	2,2552	1,4757	1,2855
9	2,3654	1,6422	1,2231	2,2382	1,4520	1,2627
10	2,3387	1,6064	1,2215	2,1533	1,3456	1,2612
11	2,3197	1,5686	1,2163	2,1525	1,2555	1,2539
12	2,2883	1,5404	1,2049	2,1527	1,2315	1,2573
13	2,2518	1,4900	1,1981	2,1340	1,2129	1,2590
14	2,2515	1,4642	1,1925	2,0881	1,1834	1,2515
15	2,2279	1,4534	1,1866	2,0814	1,1433	1,2469
16	2,2113	1,4117	1,1800	1,9944	1,1217	1,2195
17	2,1843	1,3955	1,1734	1,9861	1,0800	1,2194
18	2,1584	1,3854	1,1709	1,9796	1,1057	1,2018
19	2,1524	1,3612	1,1698	1,9523	1,0860	1,1890
20	2,1436	1,3456	1,1647	1,9506	1,0602	1,1842
21	2,1226	1,3221	1,1588	1,9436	1,0383	1,1783
22	2,1098	1,3182	1,1553	1,9410	1,0124	1,1772
23	2,1137	1,2980	1,1555	1,9389	1,0478	1,1781
24	2,0752	1,2782	1,1492	1,9392	1,0095	1,1765
25	2,0666	1,2692	1,1457	1,9391	1,0146	1,1757
26	2,0644	1,2557	1,1449	1,9217	0,9967	1,1614
27	2,0470	1,2547	1,1381	1,9120	0,9778	1,1569
28	2,0195	1,2252	1,1334	1,9185	1,0043	1,1564
29	2,0218	1,2164	1,1317	1,9161	0,9587	1,1562
30	2,0068	1,2227	1,1284	1,9077	0,9312	1,1495
31	1,9928	1,1923	1,1238	1,9081	0,9245	1,1455
32	1,9794	1,1865	1,1212	1,9007	0,9168	1,1498
33	1,9603	1,1822	1,1201	1,8983	0,9296	1,1480
34	1,9625	1,1716	1,1212	1,9009	0,9296	1,1453
35	1,9654	1,1682	1,1186	1,8960	0,9391	1,1387
36	1,9509	1,1548	1,1155	1,9084	0,9071	1,1407
37	1,9407	1,1541	1,1122	1,9048	0,9069	1,1478
38	1,9410	1,1398	1,1157	1,9090	0,8920	1,1407

Lanjutan Lampiran 2

Epoch	*Train*			*Val*		
	seg_loss	***cls_loss***	***dfl_loss***	***seg_loss***	***cls_loss***	***dfl_loss***
39	1,9123	1,1300	1,1103	1,9109	0,8919	1,1373
40	1,9089	1,1269	1,1071	1,9035	0,8906	1,1342
41	1,9078	1,1233	1,1081	1,9024	0,8819	1,1325
42	1,8880	1,1149	1,1017	1,9011	0,8825	1,1342
43	1,8720	1,0960	1,0991	1,9027	0,8906	1,1344
44	1,8736	1,0922	1,0980	1,9090	0,8635	1,1325
45	1,8696	1,0879	1,0954	1,9041	0,8705	1,1343
46	1,8512	1,0712	1,0911	1,8967	0,8637	1,1343
47	1,8414	1,0701	1,0910	1,8914	0,8527	1,1305
48	1,8348	1,0605	1,0872	1,8891	0,8594	1,1263
49	1,8251	1,0592	1,0876	1,8710	0,8464	1,1253
50	1,8141	1,0594	1,0865	1,8665	0,8458	1,1254
51	1,8198	1,0556	1,0875	1,8590	0,8242	1,1255
52	1,8242	1,0581	1,0867	1,8606	0,8365	1,1261
53	1,8135	1,0413	1,0860	1,8501	0,8199	1,1246
54	1,8038	1,0454	1,0830	1,8350	0,8234	1,1254
55	1,8064	1,0344	1,0846	1,8220	0,8175	1,1253
56	1,7794	1,0239	1,0796	1,8230	0,8213	1,1252
57	1,7809	1,0234	1,0774	1,8160	0,8198	1,1258
58	1,7799	1,0178	1,0787	1,8056	0,8204	1,1259
59	1,7620	1,0129	1,0733	1,8060	0,8074	1,1271
60	1,7490	0,9983	1,0725	1,7956	0,8283	1,1260
61	1,7177	0,9895	1,0627	1,7873	0,8283	1,1168
62	1,7164	0,9744	1,0618	1,7809	0,8163	1,1160
63	1,7073	0,9747	1,0618	1,7898	0,8051	1,1160
64	1,7104	0,9744	1,0640	1,7621	0,8051	1,1150
65	1,7212	0,9710	1,0635	1,7626	0,8051	1,1155
66	1,7141	0,9634	1,0630	1,7592	0,7975	1,1142
67	1,7041	0,9689	1,0604	1,7682	0,7949	1,1146
68	1,7084	0,9567	1,0614	1,7482	0,7945	1,1156
69	1,6770	0,9444	1,0573	1,7209	0,7955	1,1157
70	1,6856	0,9473	1,0563	1,7137	0,8007	1,1165
71	1,6850	0,9344	1,0575	1,7136	0,8028	1,1106
72	1,6768	0,9346	1,0543	1,7146	0,8030	1,1103
73	1,6582	0,9348	1,0577	1,7134	0,8087	1,1101
74	1,6305	0,9386	1,0564	1,7121	0,8058	1,1015
75	1,6205	0,9251	1,0462	1,6909	0,8020	1,1014
76	1,6141	0,9189	1,0438	1,6857	0,8019	1,1013

Lanjutan Lampiran 2

Epoch	*Train*			*Val*		
	seg_loss	***cls_loss***	***dfl_loss***	***seg_loss***	***cls_loss***	***dfl_loss***
77	1,6180	0,9090	1,0414	1,6654	0,7980	1,1003
78	1,6163	0,9012	1,0416	1,6721	0,7936	1,0980
79	1,6192	0,9003	1,0428	1,6801	0,7919	1,0904
80	1,6140	0,9008	1,0434	1,6756	0,7913	1,0906
81	1,6154	0,8929	1,0424	1,6686	0,7901	1,0958
82	1,6160	0,8969	1,0402	1,6699	0,7885	1,0934
83	1,6130	0,8917	1,0413	1,6629	0,7840	1,0927
84	1,5924	0,8808	1,0404	1,6609	0,7789	1,0920
85	1,6042	0,8839	1,0398	1,6566	0,7819	1,0919
86	1,6047	0,8844	1,0380	1,6435	0,7858	1,0917
87	1,6041	0,8792	1,0371	1,6334	0,7900	1,0899
88	1,6099	0,8605	1,0324	1,6111	0,7909	1,0887
89	1,6006	0,8491	1,0248	1,6220	0,7913	1,0885
90	1,6043	0,8392	1,0215	1,6216	0,7905	1,0858
91	1,5959	0,8311	1,0215	1,6288	0,7903	1,0887
92	1,5836	0,8327	1,0220	1,6141	0,7891	1,0898
93	1,5883	0,8302	1,0200	1,6149	0,7882	1,0810
94	1,5740	0,8353	1,0124	1,6026	0,7869	1,0810
95	1,5503	0,8342	1,0124	1,6027	0,7865	1,0754
96	1,5502	0,8330	1,0125	1,6027	0,7854	1,0730
97	1,5425	0,8341	1,0124	1,6017	0,7830	1,0711
98	1,5327	0,8365	1,0126	1,6081	0,7811	1,0710
99	1,5275	0,8294	1,0123	1,6003	0,7805	1,0701
100	1,5283	0,8138	1,0123	1,6029	0,7835	1,0659
101	1,5155	0,8041	1,0127	1,6003	0,7840	1,0676
102	1,5107	0,7984	1,0126	1,5949	0,7848	1,0685
103	1,5100	0,7878	1,0032	1,5914	0,7860	1,0612
104	1,4940	0,7853	1,0100	1,5972	0,7782	1,0609
105	1,4709	0,7719	1,0030	1,5832	0,7787	1,0594
106	1,4622	0,7795	1,0053	1,5829	0,7788	1,0545
107	1,4625	0,7871	1,0098	1,5818	0,7791	1,0559
108	1,4627	0,7962	1,0012	1,5759	0,7803	1,0563
109	1,4500	0,7935	1,0013	1,5723	0,7805	1,0520
110	1,4449	0,8021	1,0014	1,5628	0,7794	1,0507
111	1,4410	0,8014	1,0016	1,5617	0,7792	1,0597
112	1,4380	0,7955	1,0014	1,5468	0,7793	1,0598
113	1,4264	0,8066	1,0015	1,5369	0,7799	1,0498
114	1,4123	0,8098	1,0018	1,5270	0,7800	1,0481

Lanjutan Lampiran 2

Epoch	*Train*			*Val*		
	seg_loss	***cls_loss***	***dfl_loss***	***seg_loss***	***cls_loss***	***dfl_loss***
115	1,4238	0,8016	1,0017	1,5267	0,7790	1,0479
116	1,4054	0,7982	1,0024	1,5206	0,7796	1,0468
117	1,4001	0,7988	1,0029	1,5126	0,7792	1,0441
118	1,3964	0,7876	1,0031	1,5013	0,7787	1,0410
119	1,3961	0,7875	1,0036	1,4960	0,7788	1,0397
120	1,3869	0,7864	1,0035	1,4926	0,7787	1,0372
121	1,3759	0,7863	1,0038	1,4900	0,7755	1,0350
122	1,3639	0,7852	1,0100	1,4858	0,7780	1,0350
123	1,3669	0,7785	1,0099	1,4826	0,7784	1,0330
124	1,3576	0,7784	1,0099	1,4662	0,7789	1,0320
125	1,3542	0,7740	1,0100	1,4695	0,7784	1,0301
126	1,3568	0,7747	1,0100	1,4471	0,7793	1,0290
127	1,3443	0,7745	1,0003	1,4407	0,7798	1,0271
128	1,3437	0,7675	1,0004	1,4327	0,7795	1,0251
129	1,3435	0,7675	1,0021	1,4278	0,7799	1,0231
130	1,3433	0,7675	1,0011	1,4260	0,7794	1,0212
131	1,3449	0,7676	1,0012	1,4211	0,7798	1,0210
132	1,3457	0,7676	1,0055	1,4203	0,7795	1,0200
133	1,3347	0,7568	1,0040	1,4128	0,7799	1,0199
134	1,3349	0,7568	1,0001	1,4028	0,7794	1,0193
135	1,3249	0,7568	1,0002	1,3956	0,7794	1,0190
136	1,3349	0,7569	0,9979	1,3914	0,7715	1,0187
137	1,3322	0,7567	0,9982	1,3864	0,7736	1,0184
138	1,3310	0,7457	0,9985	1,3829	0,7712	1,0182
139	1,3301	0,7457	0,9989	1,3803	0,7713	1,0175
140	1,3300	0,7457	0,9989	1,3830	0,7723	1,0171
141	1,3268	0,7457	0,9903	1,3800	0,7719	1,0161
142	1,3258	0,7356	0,9810	1,3790	0,7719	1,0156
143	1,3249	0,7357	0,9810	1,3780	0,7712	1,0146
144	1,3249	0,7346	0,9810	1,3771	0,7706	1,0141
145	1,3248	0,7346	0,9813	1,3737	0,7708	1,0139
146	1,3247	0,7246	0,9801	1,3702	0,7708	1,0137
147	1,3246	0,7245	0,9810	1,3702	0,7707	1,0133
148	1,3247	0,7235	0,9781	1,3700	0,7705	1,0124
149	1,3245	0,7235	0,9710	1,3700	0,7703	1,0121
150	1,3244	0,7235	0,9701	1,3699	0,7701	1,0101

Lampiran 3. *Source code* model YOLOv8 *object detection*

```
# Install libraries
pip install ultralytics
pip install roboflow

# Import libraries
from roboflow import Roboflow
from ultralytics import YOLO

# Load dataset
rf = Roboflow(api_key="API_KEY")
project = rf.workspace("WORKSPACE_NAME").project("PROJECT_NAME")
version = project.version(1)
dataset = version.download("yolov8")

# Load model
 = YOLO('yolov8n.pt')

# Train model
results = model.train(data ='dataset.yaml',
                      epochs = 100,
                      lrf = 0.01,
                      batch = 16,
                      optimizer = 'auto',
                      imgsz = 640)
# Resume model
results = model.train(resume=True)

# Validate model
metrics = model.val()
```

Lampiran 4. *Source code* model YOLOv8 *instance segmentation*

```
# Install libraries
pip install ultralytics
pip install roboflow

# Import libraries
from roboflow import Roboflow
from ultralytics import YOLO

# Load dataset
rf = Roboflow(api_key="API_KEY")
project = rf.workspace("WORKSPACE_NAME").project("PROJECT_NAME")
version = project.version(1)
dataset = version.download("yolov8")

# Load model
model = YOLO('yolov8n-seg.pt')

# Train model
results = model.train(data ='dataset.yaml',
                      epochs = 100,
                      lrf = 0.01,
                      batch = 16,
                      optimizer = 'auto',
                      imgsz = 640)
# Resume model
results = model.train(resume=True)

# Validate model
metrics = model.val()
```

Lampiran 5. *Source code dashboard*

```
# FILE requirements.txt
altair==5.1.1
attrs==23.1.0
backports.zoneinfo;python_version<"3.9"
blinker==1.6.2
cachetools==5.3.1
certifi==2023.7.22
charset-normalizer==3.2.0
click==8.1.7
cmake==3.27.4.1
contourpy==1.1.0
cycler==0.11.0
Cython==3.0.2
filelock==3.12.3
fonttools==4.42.1
gitdb==4.0.10
GitPython==3.1.35
idna==3.4
importlib-metadata==6.8.0
importlib-resources==6.0.1
Jinja2==3.1.2
jsonschema==4.19.0
jsonschema-specifications==2023.7.1
kiwisolver==1.4.5
lapx==0.5.4
lit==16.0.6
markdown-it-py==3.0.0
MarkupSafe==2.1.3
matplotlib==3.7.2
mdurl==0.1.2
mpmath==1.3.0
networkx==3.1
numpy==1.24.4
nvidia-cublas-cu11==11.10.3.66
nvidia-cuda-cupti-cu11==11.7.101
nvidia-cuda-nvrtc-cu11==11.7.99
nvidia-cuda-runtime-cu11==11.7.99
nvidia-cudnn-cu11==8.5.0.96
nvidia-cufft-cu11==10.9.0.58
nvidia-curand-cu11==10.2.10.91
nvidia-cusolver-cu11==11.4.0.1
```

```
nvidia-cusparse-cu11==11.7.4.91
nvidia-nccl-cu11==2.14.3
nvidia-nvtx-cu11==11.7.91
opencv-python-headless==4.8.0.76
packaging==23.1
pandas==2.0.3
Pillow==9.5.0
pkgutil-resolve-name==1.3.10
protobuf==4.24.3
psutil==5.9.5
py-cpuinfo==9.0.0
pyarrow==13.0.0
pydeck==0.8.0
Pygments==2.16.1
Pympler==1.0.1
pyparsing==3.0.9
python-dateutil==2.8.2
pytube==15.0.0
pytz==2023.3.post1
pytz-deprecation-shim==0.1.0.post0
PyYAML==6.0.1
referencing==0.30.2
requests==2.31.0
rich==13.5.2
rpds-py==0.10.2
scipy==1.10.1
seaborn==0.12.2
six==1.16.0
smmap==5.0.0
streamlit==1.26.0
sympy==1.12
tenacity==8.2.3
toml==0.10.2
toolz==0.12.0
torch==2.0.1
torchvision==0.15.2
tornado==6.3.3
tqdm==4.66.1
triton==2.0.0
typing-extensions==4.7.1
tzdata==2023.3
tzlocal==4.3.1
ultralytics==8.0.173
```

```
urllib3==2.0.4
validators==0.22.0
watchdog==3.0.0
zipp==3.16.2
```

```
# FILE packages.txt
freeglut3-dev
libgtk2.0-dev
libgl1-mesa-glx
```

```
# FILE app.py
# Python In-built packages
from pathlib import Path
import PIL

# External packages
import streamlit as st

# Local Modules
import settings
import helper

# Setting page layout
st.set_page_config(
    page_title="PPE Detection Dashboard",
    page_icon="👷",
    layout="wide",
    initial_sidebar_state="expanded"
)

# Main page heading
st.title("PPE Detection Dashboard")

# Sidebar
st.sidebar.header("Model Configurations")

# Model Options
model_type = st.sidebar.radio(
    "Select Model", ['YOLOv8 Object Detection', 'YOLOv8 Instance
Segmentation'])

confidence = float(st.sidebar.slider(
```

```
    "Select Confidence", 25, 100, 40)) / 100

# Selecting Detection Or Segmentation
if model_type == 'YOLOv8 Object Detection':
    model_path = Path(settings.DETECTION_MODEL)
elif model_type == 'YOLOv8 Instance Segmentation':
    model_path = Path(settings.SEGMENTATION_MODEL)

# Load Pre-trained Model
try:
    model = helper.load_model(model_path)
except Exception as ex:
    st.error(f"Unable to load model. Check the specified path:
{model_path}")
    st.error(ex)

st.sidebar.header("Source Configuration")
source_radio = st.sidebar.radio(
    "Select Source", settings.SOURCES_LIST)

# Source Options
source_img = None
source_vid = None

if source_radio == settings.IMAGE:
    helper.detect_image(confidence, model)

elif source_radio == settings.VIDEO:
    helper.detect_video(confidence, model)

elif source_radio == settings.WEBCAM:
    helper.detect_webcam(confidence, model)

elif source_radio == settings.RTSP:
    helper.detect_rtsp_stream(confidence, model)

elif source_radio == settings.YOUTUBE:
    helper.detect_youtube_video(confidence, model)

else:
    st.error("Please select a valid source type!")
```

```
# FILE helper.py
import time
import tempfile
import streamlit as st
import cv2
import PIL
from ultralytics import YOLO
from pytube import YouTube
import settings

def load_model(model_path):
    model = YOLO(model_path)
    return model

def display_tracker_options():
    display_tracker = st.radio("Display Tracker", ('Yes', 'No'))
    is_display_tracker = True if display_tracker == 'Yes' else
False
    if is_display_tracker:
        tracker_type = st.radio("Tracker", ("bytetrack.yaml",
"botsort.yaml"))
        return is_display_tracker, tracker_type
    return is_display_tracker, None

def _display_detected_frames(conf, model, st_frame, image,
is_display_tracking=None, tracker=None):
    image = cv2.resize(image, (720, int(720*(9/16))))
    if is_display_tracking:
        res = model.track(image, conf=conf, persist=True,
tracker=tracker)
    else:
        res = model.predict(image, conf=conf)
    res_plotted = res[0].plot()
    st_frame.image(res_plotted,
                   caption='Detected Video',
                   channels="BGR",
                   use_column_width=True
                   )

def detect_image(conf, model):
    source_img = st.sidebar.file_uploader(
        "Upload an image...", type=("jpg", "jpeg", "png", 'bmp',
'webp'))
```

```
    col1, col2 = st.columns(2)
    with col1:
        try:
            if source_img is None:
                default_image_path = str(settings.DEFAULT_IMAGE)
                default_image =
PIL.Image.open(default_image_path)
                st.image(default_image_path, caption="Default
Image",
                         use_column_width=True)
            else:
                uploaded_image = PIL.Image.open(source_img)
                st.image(source_img, caption="Uploaded Image",
                         use_column_width=True)
        except Exception as ex:
            st.error("Error occurred while opening the image.")
            st.error(ex)

    with col2:
        if source_img is None:
            default_detected_image_path =
str(settings.DEFAULT_DETECT_IMAGE)
            default_detected_image = PIL.Image.open(
                default_detected_image_path)
            st.image(default_detected_image_path,
caption='Detected Image',
                     use_column_width=True)
        else:
            if st.sidebar.button('Detect Image Objects'):
                res = model.predict(uploaded_image,
                                    conf=conf)
                boxes = res[0].boxes
                res_plotted = res[0].plot()[:, :, ::-1]
                st.image(res_plotted, caption='Detected Image',
use_column_width=True)

def detect_video(conf, model):
    source_video = st.sidebar.file_uploader(
        "Upload a video...", type=("mp4", "avi", "mov", "wmv",
"flv", "mkv"))

    is_display_tracker, tracker = display_tracker_options()
```

```
    if source_video is not None:
        st.video(source_video)
        if st.sidebar.button('Detect Video Objects'):
            try:
                temp_file =
tempfile.NamedTemporaryFile(delete=False)
                temp_file.write(source_video.read())
                vid_cap = cv2.VideoCapture(temp_file.name)
                st_frame = st.empty()
                while vid_cap.isOpened():
                    success, image = vid_cap.read()
                    if success:
                        _display_detected_frames(conf, model,
st_frame, image, is_display_tracker, tracker)
                    else:
                        vid_cap.release()
                        break
            except Exception as e:
                st.sidebar.error("Error loading video: " +
str(e))

def detect_webcam(conf, model):
    source_webcam = settings.WEBCAM_PATH
    is_display_tracker, tracker = display_tracker_options()
    if st.sidebar.button('Detect Webcam Objects'):
        try:
            vid_cap = cv2.VideoCapture(source_webcam)
            st_frame = st.empty()
            while (vid_cap.isOpened()):
                success, image = vid_cap.read()
                if success:
                    _display_detected_frames(conf,
                                             model,
                                             st_frame,
                                             image,
                                       is_display_tracker,
                                             tracker)
                else:
                    vid_cap.release()
                    break
        except Exception as e:
            st.sidebar.error("Error loading video: " + str(e))
```

```
# FILE settings.py
from pathlib import Path
import sys

# Paths
file_path = Path(__file__).resolve()
root_path = file_path.parent
if root_path not in sys.path:
    sys.path.append(str(root_path))
ROOT = root_path.relative_to(Path.cwd())

# Sources
IMAGE = 'Image'
VIDEO = 'Video'
WEBCAM = 'Webcam'
RTSP = 'RTSP'
YOUTUBE = 'YouTube'
SOURCES_LIST = [IMAGE, VIDEO, WEBCAM]

# Images config
IMAGES_DIR = ROOT / 'images'
DEFAULT_IMAGE = IMAGES_DIR / 'default_image_ppe.jpg'
DEFAULT_DETECT_IMAGE = IMAGES_DIR /
'default_image_ppe_output.jpg'

# Videos config
VIDEO_DIR = ROOT / 'videos'
VIDEO_1_PATH = VIDEO_DIR / 'video_ppe1.mp4'
VIDEO_2_PATH = VIDEO_DIR / 'video_ppe2.mp4'
VIDEOS_DICT = {
    'video_1': VIDEO_1_PATH,
    'video_2': VIDEO_2_PATH,
}

# Model config
MODEL_DIR = ROOT / 'models'
DETECTION_MODEL = MODEL_DIR / 'ppe_detector.pt'
SEGMENTATION_MODEL = MODEL_DIR / 'ppe_segmentator.pt'

# Webcam
WEBCAM_PATH = 0
```