Appendix A
Task Description MT-70-23 Eirik Illing - Signed

Appendix A

University of South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Object detection, information extraction and analysis of operator interface images using computer vision and machine learning

USN supervisor: Associate Professor Ole M. Brastein and Professor Nils-Olav Skeie

External partner: Emerson Automation Solutions, Geir Falkevik

Task background:
Migrating from old outdated human machine interfaces (HMI), process displays or operator graphics to new modern high-performance HMI´s (HPHMI) is often time consuming and costly. When creating a proposal for such migration projects, the sales and project team are often given an overview of today´s old displays in configuration files or in plain images. If the input is configuration files, the engineers have tools for extracting data directly from these files, resulting in a good estimate of display complexity and therefore a fair time and cost estimate. However, if the input is plain images, the complexity analysis of these displays is done manually by counting custom and non-custom objects in the display, static and dynamic objects, clustering etc. This manual analysis is very time consuming and has a much higher degree of uncertainty that could result in poor time and cost estimates.

Emerson delivers a world known distributed control system (DCS) known as DeltaV. DeltaV comes with a fully integrated operator graphics tool known as DeltaV Operate. This tool has served its purpose for many years for all of Emerson´s customer and will continue to do so in many years to come. However, this operator graphics tool is based on older technology and a new and better fully integrated operator graphics tool known as DeltaV Live has come to replace it. DeltaV Live is a state-of-the-art modern stable framework for high performance operator graphics, so migrating from DeltaV Operate to DeltaV Live is in high demand. These migration projects are the foundation for this master´s thesis, where Emerson wants to investigate the possibility for creating a tool to do a complexity analysis of old DeltaV Operate operator graphics, to get a good and fair estimate of migration time and cost for its customers.

Task description:
Interim goals:

- Summary of literature review regarding object detection methods in images (containing a large quantity of objects).
- Choose one or more suitable approaches for object detection and object classification to extract components and information from images.
- Describe how to obtain valuable datasets for training, validating, and testing models for this specific task. Look into the possibility of customer adjusted standard dynamo sets for object detection.
- Suggest analytical methods for pre-processing and clean-up/preparations of datasets.
- Develop machine learning models and check the accuracy and repeatability of the models.
- Develop an application focusing on user interface (UI) design for interacting with the model/software.

Student category: IIA (EET, EPE, IIA or PT students)

Is the task suitable for online students (not present at the campus)?
No

Practical arrangements:
This project is reserved for the industry master student at Emerson, Eirik Illing.

Supervision:
As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature): Ole M. Braastein

Student (write clearly in all capitalized letters): EIRIK ILLING

Student (date and signature): 01/02-23
Eirik Illing

Appendix B
GANTT Project Planning

Appendix B

Master's Thesis MT-70-23

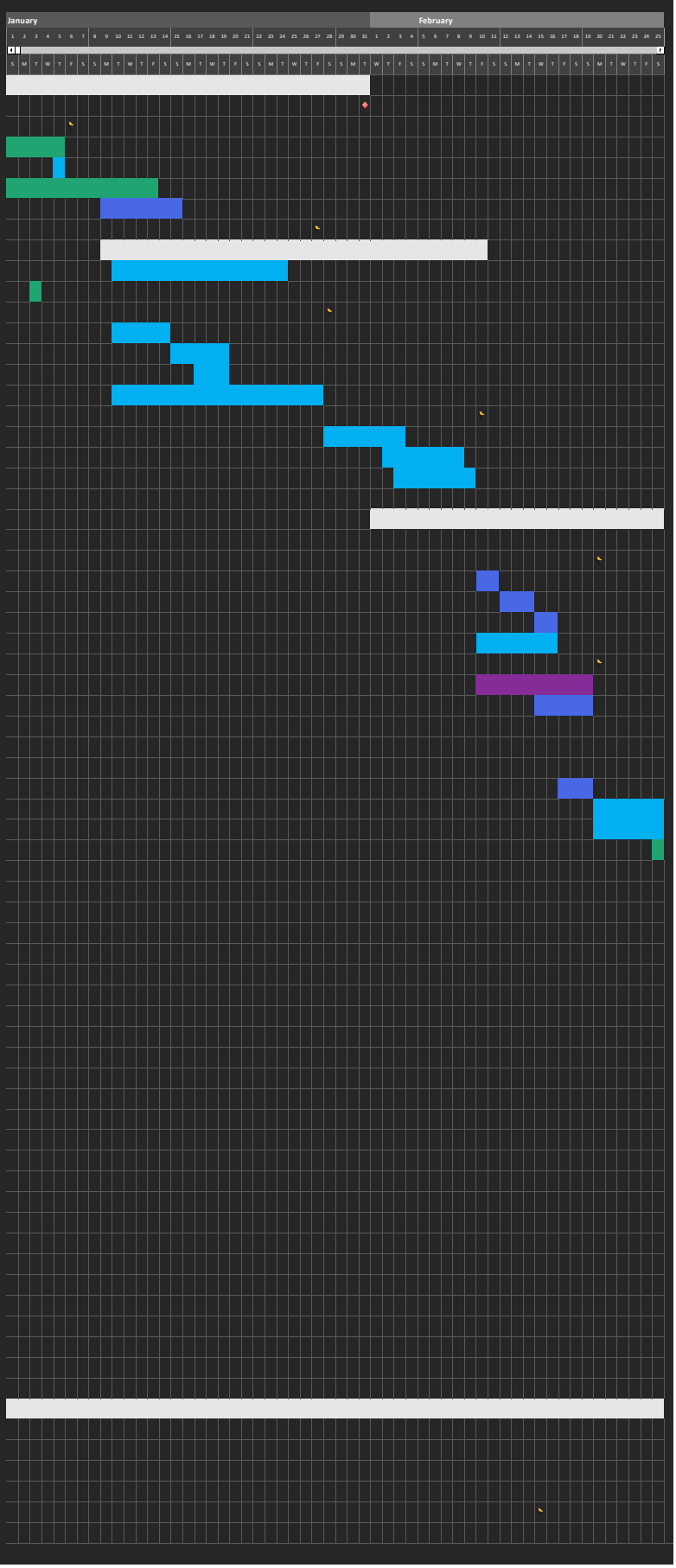
Company name	**Emerson & USN**
Project lead	**Eirik Illing**
Project Start Date:	01/01/2023
Scrolling Increment:	0

Legend: On track | Low risk | Med risk | High risk | Unassigned

Risk:	Slack:
Low Risk	10 days
Medium Risk	5 days
High Risk	0 days

	Milestone description	Category	Assigned to	Progress	Start	Days
1	**PROJECT STARTUP**				01/01/2023	31
1.1	Sign final project description	Goal	Ole, Nils-Olav, Eirik, Geir	100%	31/01/2023	1
1.2	Startup meeting	Milestone	Ole, Nils-Olav, Eirik, Geir	100%	06/01/2023	1
1.3	Project structure setup	On Track	Eirik	100%	01/01/2023	5
1.4	Get familiar with subject (Canvas)	Low Risk	Eirik	100%	05/01/2023	1
1.5	Complete fast.ai course	On Track	Eirik	100%	01/01/2023	13
1.6	Collect datasets	Med Risk	Eirik	100%	09/01/2023	7
1.7	Adjust project description (if applicable)	Milestone	Ole, Nils-Olav, Eirik, Geir	100%	27/01/2023	1
2	**LITERATURE STUDY**				09/01/2023	33
2.1	Note keywords	Low Risk	Eirik	100%	10/01/2023	15
2.2	Setup citing software (Zotero)	On Track	Eirik	100%	03/01/2023	1
2.3	**Literature review**	Milestone	Eirik	100%	28/01/2023	1
2.3.1	Search for previous work or knowledge on the subject	Low Risk	Eirik	100%	10/01/2023	5
2.3.2	Look into detection of objects in large images	Low Risk	Eirik	100%	15/01/2023	5
2.3.3	Look into picture analysis done by using machine learning methods	Low Risk	Eirik	100%	17/01/2023	3
2.3.4	Search, read, note	Low Risk	Eirik	100%	10/01/2023	18
2.4	**Object detection**	Milestone	Eirik	100%	10/02/2023	1
2.4.1	How to train model for detection / bounding of objects	Low Risk	Eirik	100%	28/01/2023	7
2.4.2	How to obtain metadata of the bounded objects	Low Risk	Eirik	100%	02/02/2023	7
2.4.3	How to display detected objects and data related to them	Low Risk	Eirik	100%	03/02/2023	7
2.5	Mandatory meeting. Deside upon method / how to proceed.	Milestone	Eirik	100%	27/02/2023	1
3	**PROJECT EXECUTION**				01/02/2023	98
3.1	Final execution	Milestone	Eirik		10/05/2023	1
3.2	**Obtain datasets**	Milestone	Eirik	100%	20/02/2023	1
3.2.1	Collect available data from Emerson	Med Risk	Eirik	100%	10/02/2023	2
3.2.2	Look into generating new data using GAN, Stable Diff?	Med Risk	Eirik	100%	12/02/2023	3
3.2.3	Verify quality of training, validation and test data	Med Risk	Eirik	100%	15/02/2023	2
3.2.4	Look into transfer learning used to adjust models based on new dataset?	Low Risk	Eirik	100%	10/02/2023	7
3.3	**Pre-process datasets**	Milestone	Eirik	100%	20/02/2023	1
3.3.1	Lable data for image classification	High Risk	Eirik	100%	10/02/2023	10
3.3.2	Look into optimal sizing, shape, orientation, flips, colors etc.	Med Risk	Eirik	100%	15/02/2023	5
3.3.3	Annotate images for object detection	Med Risk	Eirik	100%	05/03/2023	3
3.3.4	Look into auto annotation using image classifier and sliding window	Med Risk	Eirik	100%	05/03/2023	3
3.4	**Develop and test models and tools**	Milestone	Eirik	100%	30/03/2023	1
3.4.1	Set up test environment and notebook for iterative testing	Med Risk	Eirik	100%	17/02/2023	3
3.4.2	Test different deep learning models for classification	Low Risk	Eirik	100%	20/02/2023	30
3.4.3	Compare models and methods based on accuracy and repeatability	Low Risk	Eirik	100%	20/02/2023	38
3.4.4	Develop sliding window object detection using the image classifier	On Track	Eirik	100%	25/02/2023	4
3.4.5	Develop tool for annotating images for object detection	On Track	Eirik	100%	28/02/2023	10
3.4.6	Test different object detection methods and compair them	Low Risk	Eirik	100%	10/03/2023	5
3.4.7	Extract, understand and store text in image	On Track	Eirik	100%	15/03/2023	5
3.4.8	Develop export format tool where object and tag is correlated	Low Risk	Eirik	100%	30/03/2023	3
3.5	**Develop host application**	Milestone	Eirik	100%	25/04/2023	1
3.5.1	UML Diagrams - LCD, Sequence diagram, Class diagram	Low Risk	Eirik	100%	29/03/2023	5
3.5.2	Design multi-platform application using HCD principles	Low Risk	Eirik	100%	01/04/2023	5
3.5.3	Implement solution, develop application, wrap / host model	High Risk	Eirik	100%	04/04/2023	20
3.5.4	Test and build application	High Risk	Eirik	100%	19/04/2023	7
4	**REPORT WRITING**				01/03/2023	75
4.1	Deadline for delivering Thesis	Goal	Eirik		14/05/2023	2
4.2	Create outline of report	Low Risk	Eirik	100%	01/03/2023	2
4.3	Write introduction, literature review and system description	Low Risk	Eirik	100%	03/03/2023	72
4.4	Write method	Med Risk	Eirik	100%	25/03/2023	50
4.5	Write result	High Risk	Eirik	100%	25/03/2023	50
4.6	Write discussion and conclusion	High Risk	Eirik	100%	25/04/2023	7
4.7	Write summary and abstract	High Risk	Eirik	100%	30/04/2023	14
4.8	Spell check, grammar, layout, pdf print and check, cleanup	High Risk	Eirik	100%	02/05/2023	1
4.9	Nils-Olav review of repport	On Track	Nils-Olav	100%	03/05/2023	5
4.10	Review feedback from Nils-Olav, make changes and deliver	On Track	Eirik	100%	08/05/2023	5
5	**EXPO**				15/05/2023	9
5.1	Create poster or video for Expo	High Risk	Eirik	50%	25/04/2023	29
5.2	Expo, thesis are to be presented for the public	Milestone	Eirik		24/05/2023	1
6	**ORAL PRESENTATION**					
6.1	Prepare oral presentation	Med Risk	Eirik	0%	16/05/2023	38
6.2	Deadline for oral presentation and examination	Milestone	Ole, Nils-Olav, Eirik, Geir		23/06/2023	1
7	**OTHER**				01/01/2023	173
7.1	Deadline for supervisor to find external assessor	Milestone	Ole, Nils-Olav	100%	04/04/2023	1
7.2	Deadline for deciding date for oral precentation and examination	Milestone	Ole, Nils-Olav, Eirik, Geir	100%	25/04/2023	1
7.3	Deadline for grading the thesis	Goal	Ole, Nils-Olav, External		23/06/2023	1
7.4	Presentation at Emerson	Milestone	Examination? Open for all June 6.		23/06/2023	1
7.5	Deadline for submitting 250 word abstraction for the paper	Milestone	Nils-Olav, Eirik	100%	15/02/2023	1
7.6	Deadline for submitting the paper to SIMS	Milestone	Nils-Olav, Ole, Eirik		15/05/2023	1

To add more data, Insert new rows ABOVE this one



Appendix B

Master's Thesis MT-70-23

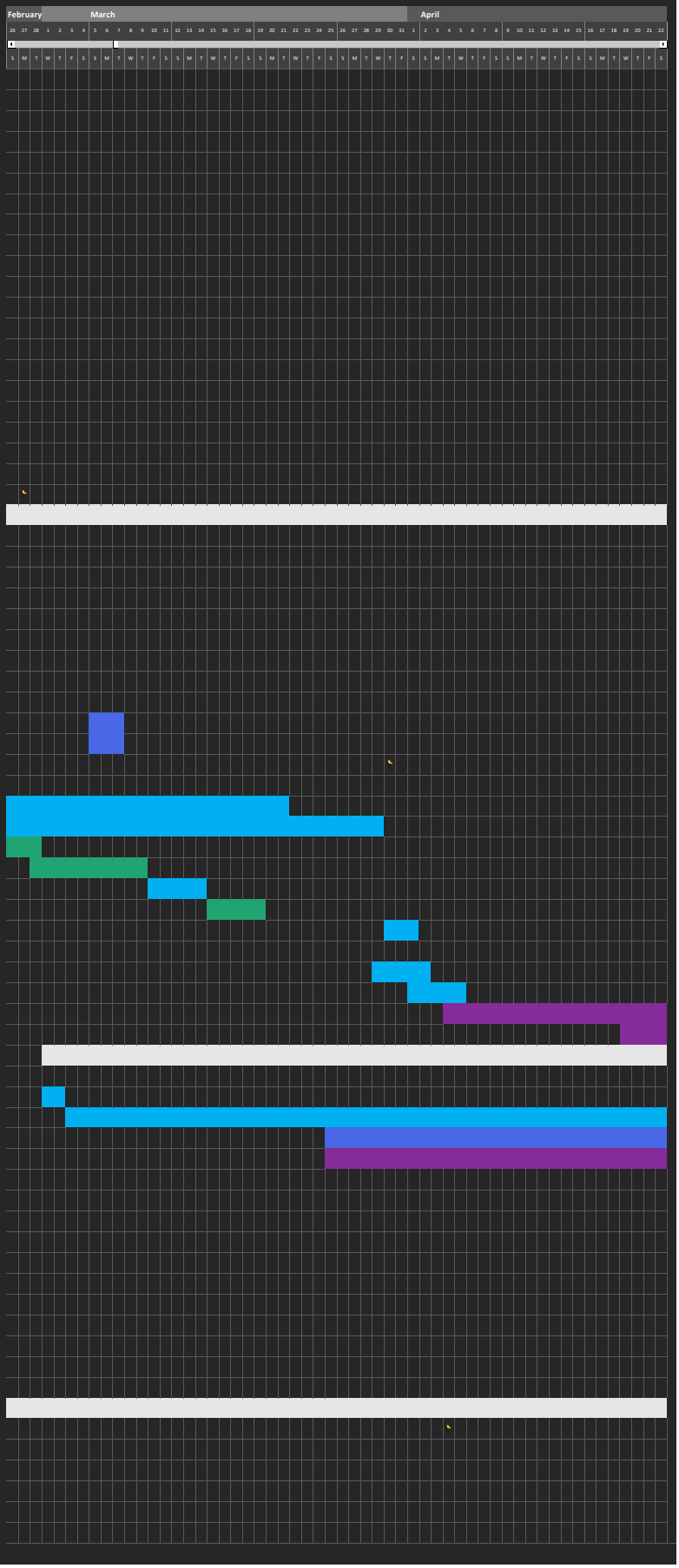
Company name	**Emerson & USN**
Project lead	**Eirik Illing**
Project Start Date:	01/01/2023
Scrolling Increment:	56

Legend: On track | Low risk | Med risk | High risk | Unassigned

Risk:	Slack:
Low Risk	10 days
Medium Risk	5 days
High Risk	0 days

	Milestone description	Category	Assigned to	Progress	Start	Days
1	**PROJECT STARTUP**				01/01/2023	31
1.1	Sign final project description	Goal	Oke, Nils-Olav, Eirik, Geir	100%	31/01/2023	1
1.2	Startup meeting	Milestone	Oke, Nils-Olav, Eirik, Geir	100%	06/01/2023	1
1.3	Project structure setup	On Track	Eirik	100%	01/01/2023	5
1.4	Get familiar with subject (Canvas)	Low Risk	Eirik	100%	05/01/2023	1
1.5	Complete fast.ai course	On Track	Eirik	100%	01/01/2023	13
1.6	Collect datasets	Med Risk	Eirik	100%	09/01/2023	7
1.7	Adjust project description (if applicable)	Milestone	Oke, Nils-Olav, Eirik, Geir	100%	27/01/2023	1
2	**LITERATURE STUDY**				09/01/2023	33
2.1	Note keywords	Low Risk	Eirik	100%	10/01/2023	15
2.2	Setup citing software (Zotero)	On Track	Eirik	100%	03/01/2023	1
2.3	**Literature review**	Milestone	Eirik	100%	28/01/2023	1
2.3.1	Search for previous work or knowledge on the subject	Low Risk	Eirik	100%	10/01/2023	5
2.3.2	Look into detection of objects in large images	Low Risk	Eirik	100%	15/01/2023	5
2.3.3	Look into picture analysis done by using machine learning methods	Low Risk	Eirik	100%	17/01/2023	3
2.3.4	Search, read, note	Low Risk	Eirik	100%	10/01/2023	18
2.4	**Object detection**	Milestone	Eirik	100%	10/02/2023	1
2.4.1	How to train model for detection / bounding of objects	Low Risk	Eirik	100%	28/01/2023	7
2.4.2	How to obtain metadata of the bounded objects	Low Risk	Eirik	100%	02/02/2023	7
2.4.3	How to display detected objects and data related to them	Low Risk	Eirik	100%	03/02/2023	7
2.5	Mandatory meeting. Deside upon method / how to proceed.	Milestone	Eirik	100%	27/02/2023	1
3	**PROJECT EXECUTION**				01/02/2023	98
3.1	Final execution	Milestone	Eirik		10/05/2023	1
3.2	**Obtain datasets**	Milestone	Eirik	100%	20/02/2023	1
3.2.1	Collect available data from Emerson	Med Risk	Eirik	100%	10/02/2023	2
3.2.2	Look into generating new data using GAN, Stable Diff?	Med Risk	Eirik	100%	12/02/2023	3
3.2.3	Verify quality of training, validation and test data	Med Risk	Eirik	100%	15/02/2023	2
3.2.4	Look into transfer learning used to adjust models based on new dataset?	Low Risk	Eirik	100%	10/02/2023	7
3.3	**Pre-process datasets**	Milestone	Eirik	100%	20/02/2023	1
3.3.1	Lable data for image classification	High Risk	Eirik	100%	10/02/2023	10
3.3.2	Look into optimal sizing, shape, orientation, flips, colors etc.	Med Risk	Eirik	100%	15/02/2023	5
3.3.3	Annotate images for object detection	Med Risk	Eirik	100%	05/03/2023	3
3.3.4	Look into auto annotation using image classifier and sliding window	Med Risk	Eirik	100%	05/03/2023	3
3.4	**Develop and test models and tools**	Milestone	Eirik	100%	30/03/2023	1
3.4.1	Set up test environment and notebook for iterative testing	Med Risk	Eirik	100%	17/02/2023	3
3.4.2	Test different deep learning models for classification	Low Risk	Eirik	100%	20/02/2023	30
3.4.3	Compare models and methods based on accuracy and repeatability	Low Risk	Eirik	100%	20/02/2023	38
3.4.4	Develop sliding window object detection using the image classifier	On Track	Eirik	100%	25/02/2023	4
3.4.5	Develop tool for annotating images for object detection	On Track	Eirik	100%	28/02/2023	10
3.4.6	Test different object detection methods and compair them	Low Risk	Eirik	100%	10/03/2023	5
3.4.7	Extract, understand and store text in image	On Track	Eirik	100%	15/03/2023	5
3.4.8	Develop export format tool where object and tag is correlated	Low Risk	Eirik	100%	30/03/2023	3
3.5	**Develop host application**	Milestone	Eirik	100%	25/04/2023	1
3.5.1	UML Diagrams - LCD, Sequence diagram, Class diagram	Low Risk	Eirik	100%	29/03/2023	5
3.5.2	Design multi-platform application using HCD principles	Low Risk	Eirik	100%	01/04/2023	5
3.5.3	Implement solution, develop application, wrap / host model	High Risk	Eirik	100%	04/04/2023	20
3.5.4	Test and build application	High Risk	Eirik	100%	19/04/2023	7
4	**REPORT WRITING**				01/03/2023	75
4.1	Deadline for delivering Thesis	Goal	Eirik		14/05/2023	2
4.2	Create outline of report	Low Risk	Eirik	100%	01/03/2023	2
4.3	Write introduction, literature review and system description	Low Risk	Eirik	100%	03/03/2023	72
4.4	Write method	Med Risk	Eirik	100%	25/03/2023	50
4.5	Write result	High Risk	Eirik	100%	25/03/2023	50
4.6	Write discussion and conclusion	High Risk	Eirik	100%	25/04/2023	7
4.7	Write summary and abstract	High Risk	Eirik	100%	30/04/2023	14
4.8	Spell check, grammar, layout, pdf print and check, cleanup	High Risk	Eirik	100%	02/05/2023	1
4.9	Nils-Olav review of repport	On Track	Nils-Olav	100%	03/05/2023	5
4.10	Review feedback from Nils-Olav, make changes and deliver	On Track	Eirik	100%	08/05/2023	5
5	**EXPO**				15/05/2023	9
5.1	Create poster or video for Expo	High Risk	Eirik	50%	25/04/2023	29
5.2	Expo, thesis are to be presented for the public	Milestone	Eirik		24/05/2023	1
6	**ORAL PRESENTATION**					
6.1	Prepare oral presentation	Med Risk	Eirik	0%	16/05/2023	38
6.2	Deadline for oral presentation and examination	Milestone	Oke, Nils-Olav, Eirik, Geir		23/06/2023	1
7	**OTHER**				01/01/2023	173
7.1	Deadline for supervisor to find external assessor	Milestone	Oke, Nils-Olav	100%	04/04/2023	1
7.2	Deadline for deciding date for oral precentation and examination	Milestone	Oke, Nils-Olav, Eirik, Geir	100%	25/04/2023	1
7.3	Deadline for grading the thesis	Goal	Oke, Nils-Olav, External		23/06/2023	1
7.4	Presentation at Emerson	Milestone	Examination? Open for all June 6.		23/06/2023	1
7.5	Deadline for submitting 250 word abstraction for the paper	Milestone	Nils-Olav, Eirik	100%	15/02/2023	1
7.6	Deadline for submitting the paper to SIMS	Milestone	Nils-Olav, Oke, Eirik		15/05/2023	1

To add more data, Insert new rows ABOVE this one



Appendix B

Master's Thesis MT-70-23

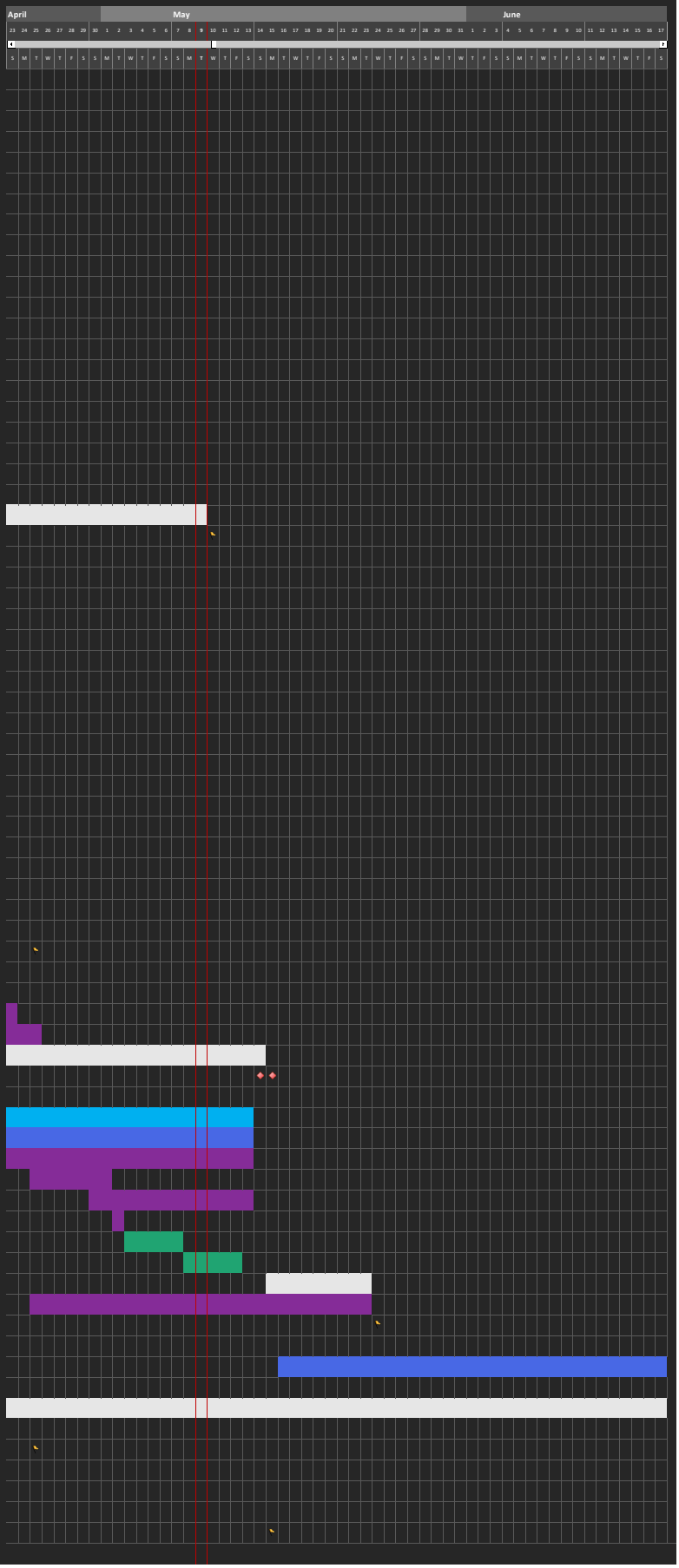
Company name Emerson & USN
Project lead Eirik Illing
Project Start Date: 01/01/2023
Scrolling Increment: 112

Legend: On track | Low risk | Med risk | High risk | Unassigned

Risk: Slack:
Low Risk 10 days
Medium Risk 5 days
High Risk 0 days

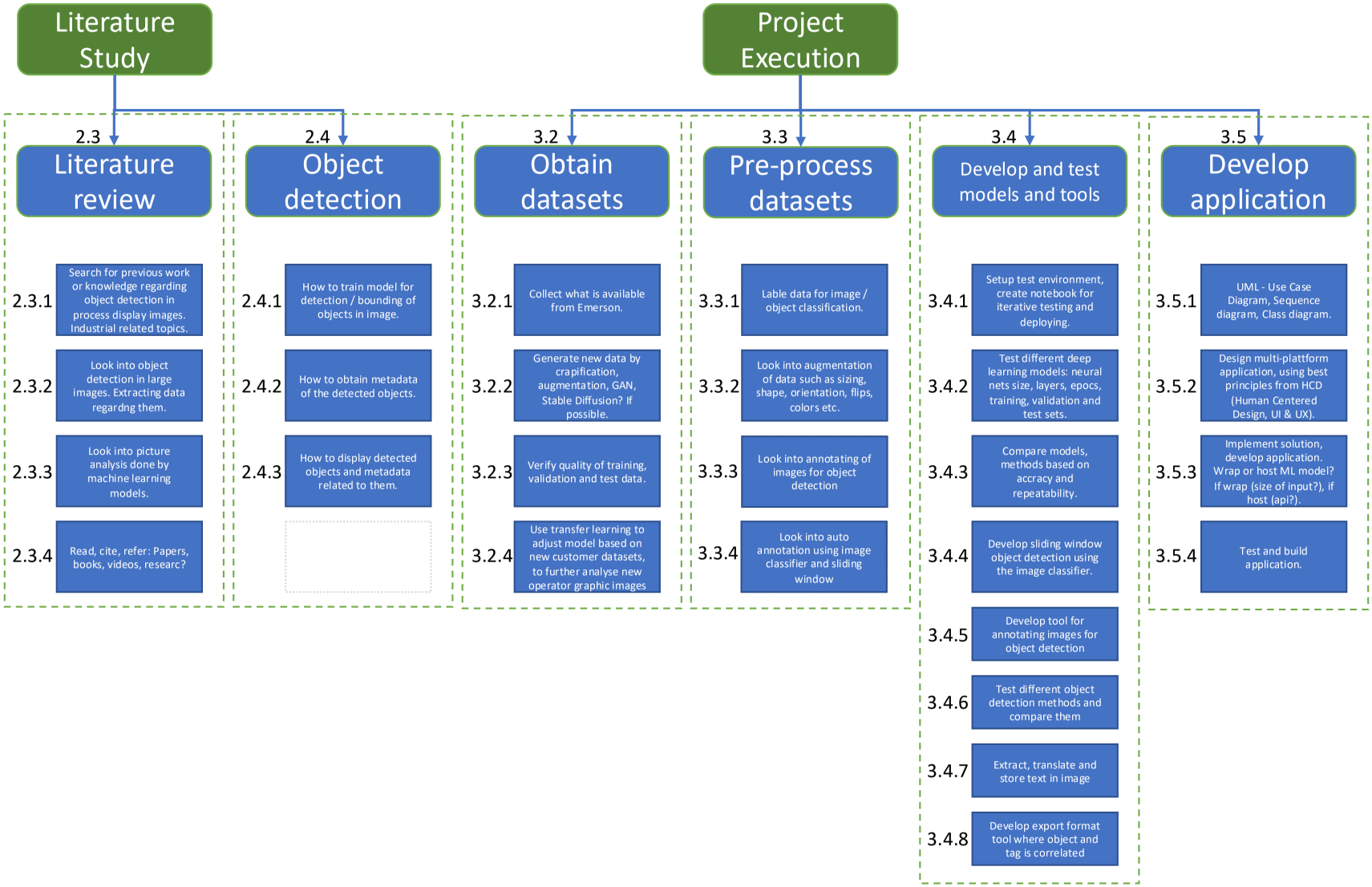
	Milestone description	Category	Assigned to	Progress	Start	Days
1	**PROJECT STARTUP**				01/01/2023	31
1.1	Sign final project description	Goal	Ole, Nils-Olav, Eirik, Geir	100%	31/01/2023	1
1.2	Startup meeting	Milestone	Ole, Nils-Olav, Eirik, Geir	100%	06/01/2023	1
1.3	Project structure setup	On Track	Eirik	100%	01/01/2023	5
1.4	Get familiar with subject (Canvas)	Low Risk	Eirik	100%	05/01/2023	1
1.5	Complete fast.ai course	On Track	Eirik	100%	01/01/2023	13
1.6	Collect datasets	Med Risk	Eirik	100%	09/01/2023	7
1.7	Adjust project description (if applicable)	Milestone	Ole, Nils-Olav, Eirik, Geir	100%	27/01/2023	1
2	**LITERATURE STUDY**				09/01/2023	33
2.1	Note keywords	Low Risk	Eirik	100%	10/01/2023	15
2.2	Setup citing software (Zotero)	On Track	Eirik	100%	03/01/2023	1
2.3	**Literature review**	Milestone	Eirik	100%	28/01/2023	1
2.3.1	Search for previous work or knowledge on the subject	Low Risk	Eirik	100%	10/01/2023	5
2.3.2	Look into detection of objects in large images	Low Risk	Eirik	100%	15/01/2023	5
2.3.3	Look into picture analysis done by using machine learning methods	Low Risk	Eirik	100%	17/01/2023	3
2.3.4	Search, read, note	Low Risk	Eirik	100%	10/01/2023	18
2.4	**Object detection**	Milestone	Eirik	100%	10/02/2023	1
2.4.1	How to train model for detection / bounding of objects	Low Risk	Eirik	100%	28/01/2023	7
2.4.2	How to obtain metadata of the bounded objects	Low Risk	Eirik	100%	02/02/2023	7
2.4.3	How to display detected objects and data related to them	Low Risk	Eirik	100%	03/02/2023	7
2.5	Mandatory meeting. Deside upon method / how to proceed.	Milestone	Eirik	100%	27/02/2023	1
3	**PROJECT EXECUTION**				01/02/2023	98
3.1	Final execution	Milestone	Eirik		10/05/2023	1
3.2	**Obtain datasets**	Milestone	Eirik	100%	20/02/2023	1
3.2.1	Collect available data from Emerson	Med Risk	Eirik	100%	10/02/2023	2
3.2.2	Look into generating new data using GAN, Stable Diff?	Med Risk	Eirik	100%	12/02/2023	3
3.2.3	Verify quality of training, validation and test data	Med Risk	Eirik	100%	15/02/2023	2
3.2.4	Look into transfer learning used to adjust models based on new dataset?	Low Risk	Eirik	100%	10/02/2023	7
3.3	**Pre-process datasets**	Milestone	Eirik	100%	20/02/2023	1
3.3.1	Lable data for image classification	High Risk	Eirik	100%	10/02/2023	10
3.3.2	Look into optimal sizing, shape, orientation, flips, colors etc.	Med Risk	Eirik	100%	15/02/2023	5
3.3.3	Annotate images for object detection	Med Risk	Eirik	100%	05/03/2023	3
3.3.4	Look into auto annotation using image classifier and sliding window	Med Risk	Eirik	100%	05/03/2023	3
3.4	**Develop and test models and tools**	Milestone	Eirik	100%	30/03/2023	1
3.4.1	Set up test environment and notebook for iterative testing	Med Risk	Eirik	100%	17/02/2023	3
3.4.2	Test different deep learning models for classification	Low Risk	Eirik	100%	20/02/2023	30
3.4.3	Compare models and methods based on accuracy and repeatability	Low Risk	Eirik	100%	20/02/2023	38
3.4.4	Develop sliding window object detection using the image classifier	On Track	Eirik	100%	25/02/2023	4
3.4.5	Develop tool for annotating images for object detection	On Track	Eirik	100%	28/02/2023	10
3.4.6	Test different object detection methods and compair them	Low Risk	Eirik	100%	10/03/2023	5
3.4.7	Extract, understand and store text in image	On Track	Eirik	100%	15/03/2023	5
3.4.8	Develop export format tool where object and tag is correlated	Low Risk	Eirik	100%	30/03/2023	3
3.5	**Develop host application**	Milestone	Eirik	100%	25/04/2023	1
3.5.1	UML Diagrams - UCD, Sequence diagram, Class diagram	Low Risk	Eirik	100%	29/03/2023	5
3.5.2	Design multi-platform application using HCD principles	Low Risk	Eirik	100%	01/04/2023	5
3.5.3	Implement solution, develop application, wrap / host model	High Risk	Eirik	100%	04/04/2023	20
3.5.4	Test and build application	High Risk	Eirik	100%	19/04/2023	7
4	**REPORT WRITING**				01/03/2023	75
4.1	Deadline for delivering Thesis	Goal	Eirik		14/05/2023	2
4.2	Create outline of report	Low Risk	Eirik	100%	01/03/2023	2
4.3	Write introduction, literature review and system description	Low Risk	Eirik	100%	03/03/2023	72
4.4	Write method	Med Risk	Eirik	100%	25/03/2023	50
4.5	Write result	High Risk	Eirik	100%	25/03/2023	50
4.6	Write discussion and conclusion	High Risk	Eirik	100%	25/04/2023	7
4.7	Write summary and abstract	High Risk	Eirik	100%	30/04/2023	14
4.8	Spell check, grammar, layout, pdf print and check, cleanup	High Risk	Eirik	100%	02/05/2023	1
4.9	Nils-Olav review of repport	On Track	Nils-Olav	100%	03/05/2023	5
4.10	Review feedback from Nils-Olav, make changes and deliver	On Track	Eirik	100%	08/05/2023	5
5	**EXPO**				15/05/2023	9
5.1	Create poster or video for Expo	High Risk	Eirik	50%	25/04/2023	29
5.2	Expo, thesis are to be presented for the public	Milestone	Eirik		24/05/2023	1
6	**ORAL PRESENTATION**					
6.1	Prepare oral presentation	Med Risk	Eirik	0%	16/05/2023	38
6.2	Deadline for oral presentation and examination	Milestone	Ole, Nils-Olav, Eirik, Geir		23/06/2023	1
7	**OTHER**				01/01/2023	173
7.1	Deadline for supervisor to find external assessor	Milestone	Ole, Nils-Olav	100%	04/04/2023	1
7.2	Deadline for deciding date for oral precentation and examination	Milestone	Ole, Nils-Olav, Eirik, Geir	100%	25/04/2023	1
7.3	Deadline for grading the thesis	Goal	Ole, Nils-Olav, External		23/06/2023	1
7.4	Presentation at Emerson	Milestone	Examination? Open for all June 6.		23/06/2023	1
7.5	Deadline for submitting 250 word abstraction for the paper	Milestone	Nils-Olav, Eirik	100%	15/02/2023	1
7.6	Deadline for submitting the paper to SIMS	Milestone	Nils-Olav, Ole, Eirik		15/05/2023	1

To add more data, Insert new rows ABOVE this one



Appendix C
WBS Project Planning

Appendix C


Literature Study
Project Execution
2.3
Literature review
2.3.1
Search for previous work or knowledge regarding object detection in process display images. Industrial related topics.
2.3.2
Look into object detection in large images. Extracting data regardng them.
2.3.3
Look into picture analysis done by machine learning models.
2.3.4
Read, cite, refer: Papers, books, videos, researc?
2.4
Object detection
2.4.1
How to train model for detection / bounding of objects in image.
2.4.2
How to obtain metadata of the detected objects.
2.4.3
How to display detected objects and metadata related to them.
3.2
Obtain datasets
3.2.1
Collect what is available from Emerson.
3.2.2
Generate new data by crapification, augmentation, GAN, Stable Diffusion? If possible.
3.2.3
Verify quality of training, validation and test data.
3.2.4
Use transfer learning to adjust model based on new customer datasets, to further analyse new operator graphic images
3.3
Pre-process datasets
3.3.1
Lable data for image / object classification.
3.3.2
Look into augmentation of data such as sizing, shape, orientation, flips, colors etc.
3.3.3
Look into annotating of images for object detection
3.3.4
Look into auto annotation using image classifier and sliding window
3.4
Develop and test models and tools
3.4.1
Setup test environment, create notebook for iterative testing and deploying.
3.4.2
Test different deep learning models: neural nets size, layers, epocs, training, validation and test sets.
3.4.3
Compare models, methods based on accracy and repeatability.
3.4.4
Develop sliding window object detection using the image classifier.
3.4.5
Develop tool for annotating images for object detection
3.4.6
Test different object detection methods and compare them
3.4.7
Extract, translate and store text in image
3.4.8
Develop export format tool where object and tag is correlated
3.5
Develop application
3.5.1
UML - Use Case Diagram, Sequence diagram, Class diagram.
3.5.2
Design multi-plattform application, using best principles from HCD (Human Centered Design, UI & UX).
3.5.3
Implement solution, develop application. Wrap or host ML model? If wrap (size of input?), if host (api?).
3.5.4
Test and build application.

Appendix D
Development Environment Elaborated

1 Development environment

Machine learning tasks can be computationally heavy to perform. Specially during development of certain applications while training and testing. A decent hardware and software environment is key for efficiency and performance. This development station and environment will be hosted on a local computer in the office, with remote access via TeamViewer. The computer will also be connected to a Raspberry PI4 that is configured to reboot if/after power loss. This Raspberry PI4 can also be reached with TeamViewer, where a wake on LAN magic package can be sent from the Raspberry PI4 to the development station, thus turning it on. The development station is configured with Wake On LAN in bios and on the Ethernet Controller.

1.1 Hardware environment

The most demanding task while developing machine learning models is the training of models and predicting large quantity of information. For this process, GPUs are key components, as they are built to perform complex parallel computation. GPUs are more suited for these kinds of tasks compared to CPU because they are specifically designed for calculations related to graphics and rendering. GPUs are equipped with more cores and higher bandwidth than CPUs, thus able to perform a lot more tasks at once. CPUs are on the other hand equipped with more powerful cores, better suited for sequential processing. One significant difference between these two is that GPUs does not dynamically allocate and dump memory the same way that CPUs does, so memory management is a key factor when working with GPU computation. There are varies methods for handling these “out of memory” error cases when working with machine learning, such as reducing batch size in training, use smaller/less complex model, mixed precision training and killing processes. So, when deciding upon hardware components for machine learning development, GPU and cooling will be the most crucial components.

For this project, an old gaming computer seemed to be a good fit. The computer has a GTX1080 overclocked GPU, an Intel Core i5-8400 processor, 16gib of DDR4 RAM, 250gib M.2 SSD. Table 1 gives an overview of components and part numbers used in the development machine.

Table 1: List of development environment hardware

Part name	**Part number**	**Description**
MSI B360I Gaming Pro AC, Socket-1151	B360I GAMING PRO AC	Motherboard
Intel Core i5-8400 Processor	BX80684I58400	CPU
Asus GeForce GTX 1080 Rog Strix	ROG STRIX-GTX1080-A8G-GAMING	GPU
Corsair Vengance LPX DDR4 2400MHz 16gb	CMK16GX4M2A2400C14	RAM
WD Black SSD 250GB M.2 PCIe	WDS250G2X0C	SSD

Cooler Master MasterWATT 650	MPX-6501-AMAAB-EU	PS

1.2 Software environment

The pc was reinstalled with Windows 10, student edition. Windows 10 is a perfectly fine multipurpose OS designed for everything from everyday use to development. However, more advanced development requiring a large quantity of open-source packages and flexibility can get tedious when working with Windows. This is mainly because Windows focus on a graphical user interface experience, while developing software often limits itself to working with command line tools. Using some sort of Linux distro therefore seems like a more appealing approach.

One thing to note about Windows is that it has better commercial software and hardware drives support. Some sort of mix, running Windows as main OS and virtualizing an Ubuntu environment is a good idea. However, running Ubuntu as a virtual machine will result in hardware limitations as it is predefined with a specific amount of computing power when set up. A virtual machine also requires some recourses just to run, and this could affect the overall machine performance. It is also tedious to set up, allocate memory and configure file sharing between Windows and virtual machine.

Second option is to dual boot the system with a native Ubuntu distro. This will give the distro full access to computing power, but the disk space needs to be partitioned giving 50/50 to Windows and Ubuntu. File sharing between these two OS's is also a hassle, and it requires the user to turn the machine on and off to switch environment. The hardware drives can also become an issue on the Linux system.

The final and most diffidently best approach is to set up a Windows Subsystem for Linux directly from Windows 10 terminal (CMD). WSL is Microsoft's answer to more flexible open-source Linux environments directly on Windows. Preventing developers from switching to Linux distros as they advance in their carrier and making it more appealing for Linux users to switch too Windows. WSL is a lightweight and integrated solution running Linux on a Windows operating system. It can directly access files and share resources with the Windows host. And since WSL also shares the same kernel as the Windows host, it also inherits the security protections. This is not the case for a virtual machine running on Windows services such as Hyper-V, VirtualBox or WMware Workstation.

Setting up WSL and installing a distro is easy. Find a good tutorial online, such as the one referred to in this section [1]. Follow it and do adjustments required for different hardware specifications. It is recommended to have some basic understanding of Linux file system and package installation. Otherwise, use the internet to search for help and solve error messages. Start by installing Docker Desktop on Windows, this is handy for containerizing projects running on the Linux kernel using the WSL as backend. It is not required to have Docker installed, but recommended. Next install WSL by running the wsl --install -d Ubuntu. Where Ubuntu specifies the Linux distro for installation. Ubuntu will then be installed on the machine, and can be opened by searching for "Ubuntu" in the Windows menu. A new terminal with the Ubuntu terminal will open, representing the Ubuntu machine. Next it is recommended to set up git and connect to a online git source-code storage and management service such as GitKraken or GitHub. Then install Visual Studio Code as a code editor on Windows, and connect it to WSL by adding the Remote Development extension pack. This gives the possibility to open any folder from the Ubuntu terminal in VSC by running the

“code .” command. After the IDE or Code editor is integrated, it is time to install development environment and packages in Ubuntu. Install MiniConda or Mamba, which is lightweight Python Conda package manager. This will give the bare minimum to create Conda environments and start Python development. Create a new Conda environment by running the “conda create -n newEnv” command. It is recommended to work in separate environments when developing to easier manage packages, prevent conflicts and backup. Finally there is one last thing that needs to be taken care of to access the processing power of the GPU hardware both in Windows and on the Ubuntu distro.

Installing packages for NVIDIA CUDA toolkit and cuDNN drivers. Go to the NVIDIA for developers website, download and install the latest CUDA driver on the Windows OS. Then download and install the cuDNN drivers for the Windows OS. Extract the cuDNN drivers from the installation folder and move them into and overwrite exiting driver folders in the \Program Files\NVIDIA GPU Computing Toolkit\CUDA\driver folder on the Windows machine. Both the bin and libnvvp folder need to be added to the Environment Variable path. A complete guide written by Bex T. can be found at towardsdatascience.com referenced here [2]. When installation on Windows machine is done, it is recommended to test it locally before installing the same driver support on the WSL Ubuntu system. This was found to be unnecessary in this project.

Next, install the same support on WSL in the Ubuntu terminal using a few simple commands shown in step 16 by Bex T. in towardsdatascience.com referenced here [1]. Then install the preferred Machine Learning libraries such as PyTorch, Tensorflow, Keras in the Conda environment created earlier or separate environments. It is recommended to keep some these separated as they may cause conflict with each other. This, however, needs to be tested and researched before use. If a mistake is made and conflicts occur, simply create new Conda environment and reinstall. Remember to install the packages that are supported for WSL and with GPU support. This can be found on the packages official cites. A list of packages used in this project can be seen in …. In this project, a WSL Ubuntu distro was created, set up with Git and MiniConda and multiple new template Conda environments were created with all packages and GPU functionality. This template is then copied into new development environments for testing and developing. This way, a fresh working environment is always available if something should go wrong in the developing environment. This environment can also be exported to a .yaml file and imported on other machines running a Conda setup on Ubuntu distro.

[1] B. T, “How to Create Perfect Machine Learning Development Environment With WSL2 on Windows 10/11,” *Medium*, Dec. 09, 2022. https://towardsdatascience.com/how-to-create-perfect-machine-learning-development-environment-with-wsl2-on-windows-10-11-2c80f8ea1f31 (accessed Feb. 14, 2023).

[2] B. T, “How to Finally Install TensorFlow 2 GPU on Windows 10 in 2022,” *Medium*, Dec. 09, 2022. https://towardsdatascience.com/how-to-finally-install-tensorflow-gpu-on-windows-10-63527910f255 (accessed Feb. 14, 2023).

Appendix E
Single-Label Classifier

Jupyter Notebook

In [1]:
```
from fastai.vision.all import *
from fastbook import *
from fastai.vision.widgets import *
from fastai.callback.fp16 import *
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
path = Path("/home/engineirik/git/classify_singl_obj/Classification_small_singleobj
Path.BASE_PATH = path
path.ls()
```

Out[2]:
```
(#20) [Path('pump_isa'),Path('status'),Path('chart'),Path('valve_m'),Path('mixe
r'),Path('valve_pr'),Path('valve'),Path('valve_p'),Path('valve_m_3w'),Path('valve_
h')...]
```

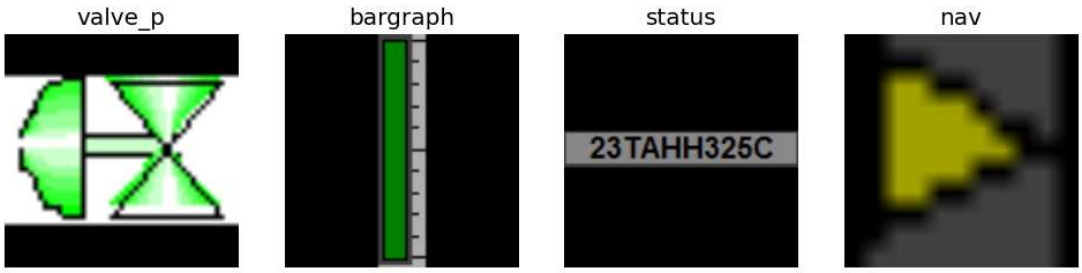
In [3]:
```
fns = get_image_files(path/"status")
len(fns)
```

Out[3]:
```
125
```

Model and Preprocessing

In [4]:
```
data = DataBlock(
    blocks=(ImageBlock, CategoryBlock),
    get_items=get_image_files,
    splitter=RandomSplitter(valid_pct=0.2, seed=42),
    get_y=parent_label,
    item_tfms=Resize(224, ResizeMethod.Pad, pad_mode='zeros')
)
dls = data.dataloaders(path)
```

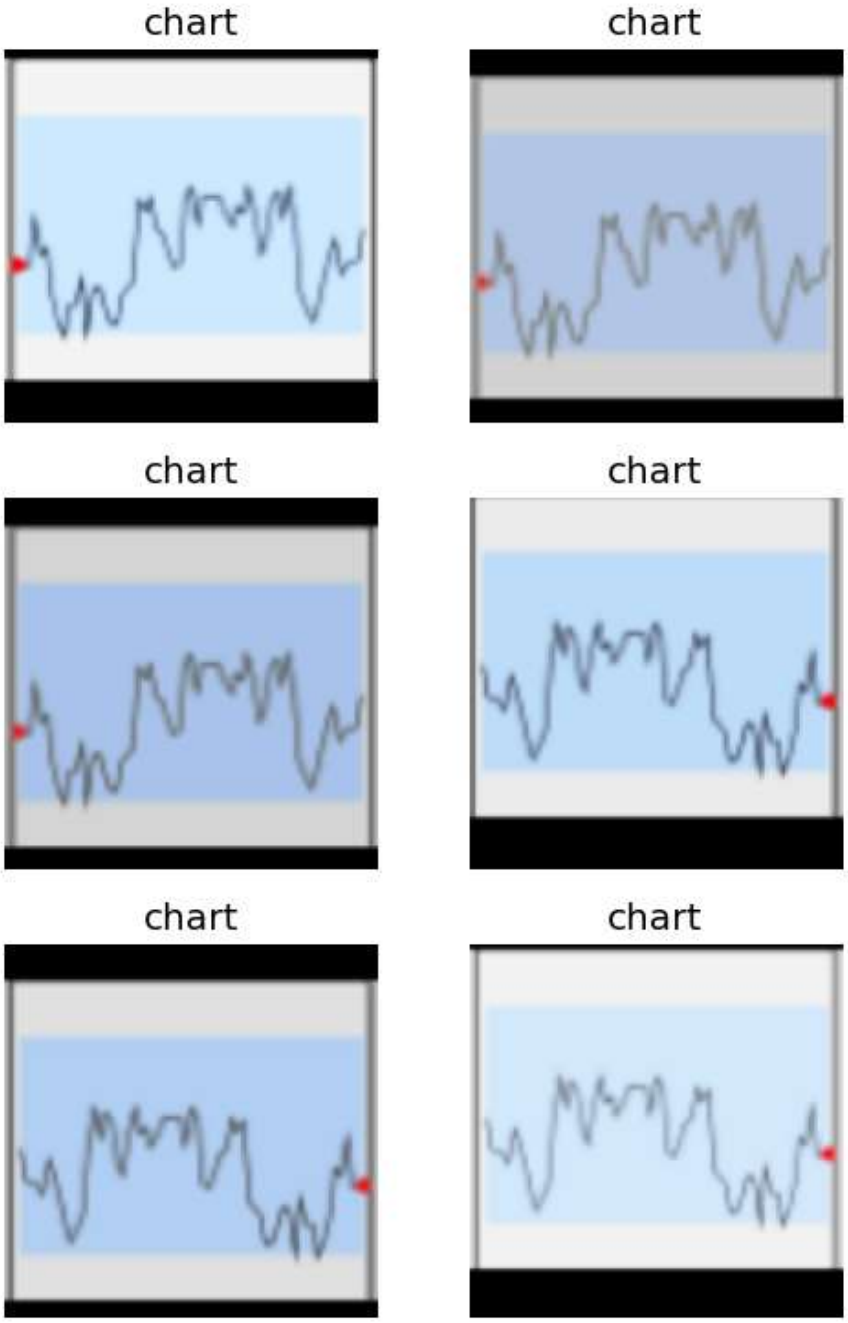
In [5]:
```
dls.valid.show_batch(max_n=4, nrows=1)
```



Augmentation

In [6]:
```
data = data.new(
    item_tfms=Resize(224, ResizeMethod.Pad, pad_mode='zeros'),
    batch_tfms=aug_transforms(size=128, min_scale=1, mult=2, max_warp=0,
                              do_flip=True, flip_vert=False, max_zoom=1.02,
                              pad_mode="zeros", max_rotate=0))
```

```
dls = data.dataloaders(path, bs=16)
dls.train.show_batch(max_n=6, nrows=3, unique=True)
```



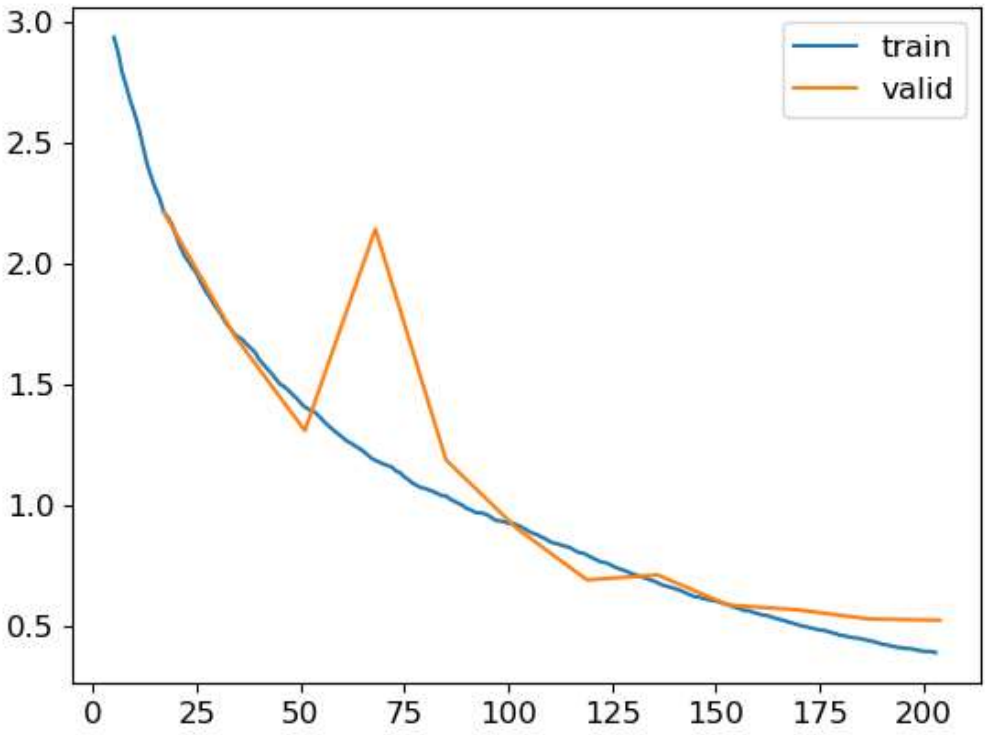
Train non pretrained model

In [6]:
```
model = xresnet50(n_out=dls.c)
learn = Learner(dls, model, loss_func=CrossEntropyLossFlat(),
                metrics=accuracy)
#Learn.fine_tune(9, freeze_epochs=3)
```

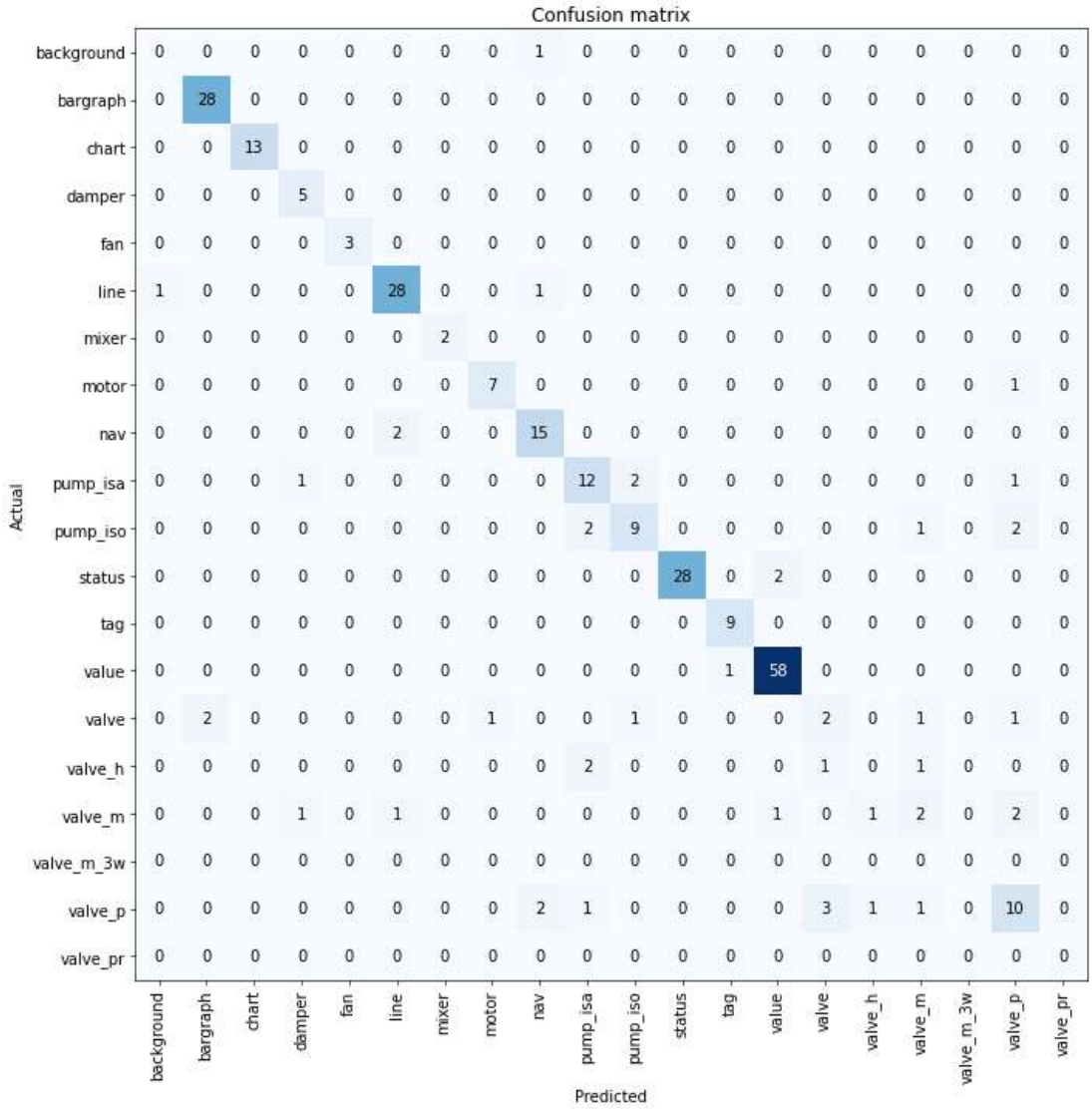
```
learn.fit_one_cycle(12)
#learn.fit_one_cycle(5, 3e-3)
```

epoch	train_loss	valid_loss	accuracy	time
0	2.268960	2.210411	0.241758	00:21
1	1.729840	1.700031	0.494505	00:19
2	1.421203	1.306692	0.600733	00:19
3	1.192690	2.140647	0.560440	00:19
4	1.037965	1.185952	0.666667	00:19
5	0.923955	0.903173	0.761905	00:19
6	0.799859	0.688817	0.776557	00:19
7	0.684407	0.709529	0.761905	00:19
8	0.591151	0.585039	0.798535	00:19
9	0.507836	0.564602	0.831502	00:19
10	0.440936	0.527033	0.846154	00:19
11	0.387851	0.522242	0.846154	00:19

In [7]:
```
learn.recorder.plot_loss()
```



In [8]:
```
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
```



In [9]:
```
#interp.plot_top_losses(20, nrows=5)
interp.most_confused(min_val=1)
```

Out[9]:
```
[('valve_p', 'valve', 3),
 ('nav', 'line', 2),
 ('pump_isa', 'pump_iso', 2),
 ('pump_iso', 'pump_isa', 2),
 ('pump_iso', 'valve_p', 2),
 ('status', 'value', 2),
 ('valve', 'bargraph', 2),
 ('valve_h', 'pump_isa', 2),
 ('valve_m', 'valve_p', 2),
 ('valve_p', 'nav', 2),
 ('background', 'nav', 1),
 ('line', 'background', 1),
 ('line', 'nav', 1),
 ('motor', 'valve_p', 1),
 ('pump_isa', 'damper', 1),
 ('pump_isa', 'valve_p', 1),
 ('pump_iso', 'valve_m', 1),
 ('value', 'tag', 1),
 ('valve', 'motor', 1),
 ('valve', 'pump_iso', 1),
 ('valve', 'valve_m', 1),
 ('valve', 'valve_p', 1),
 ('valve_h', 'valve', 1),
 ('valve_h', 'valve_m', 1),
 ('valve_m', 'damper', 1),
 ('valve_m', 'line', 1),
 ('valve_m', 'value', 1),
 ('valve_m', 'valve_h', 1),
 ('valve_p', 'pump_isa', 1),
 ('valve_p', 'valve_h', 1),
 ('valve_p', 'valve_m', 1)]
```

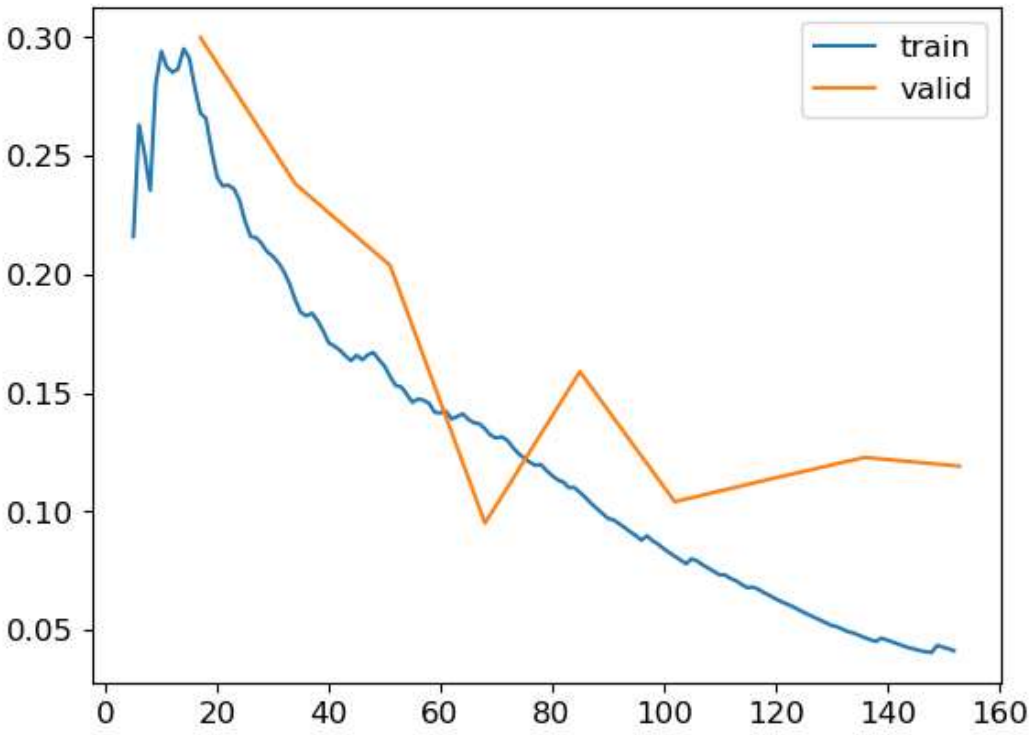
Train pretrained model

In [10]:
```
learn = vision_learner(dls, resnet50, metrics=accuracy).to_fp16()
learn.fine_tune(9, freeze_epochs=3)
```

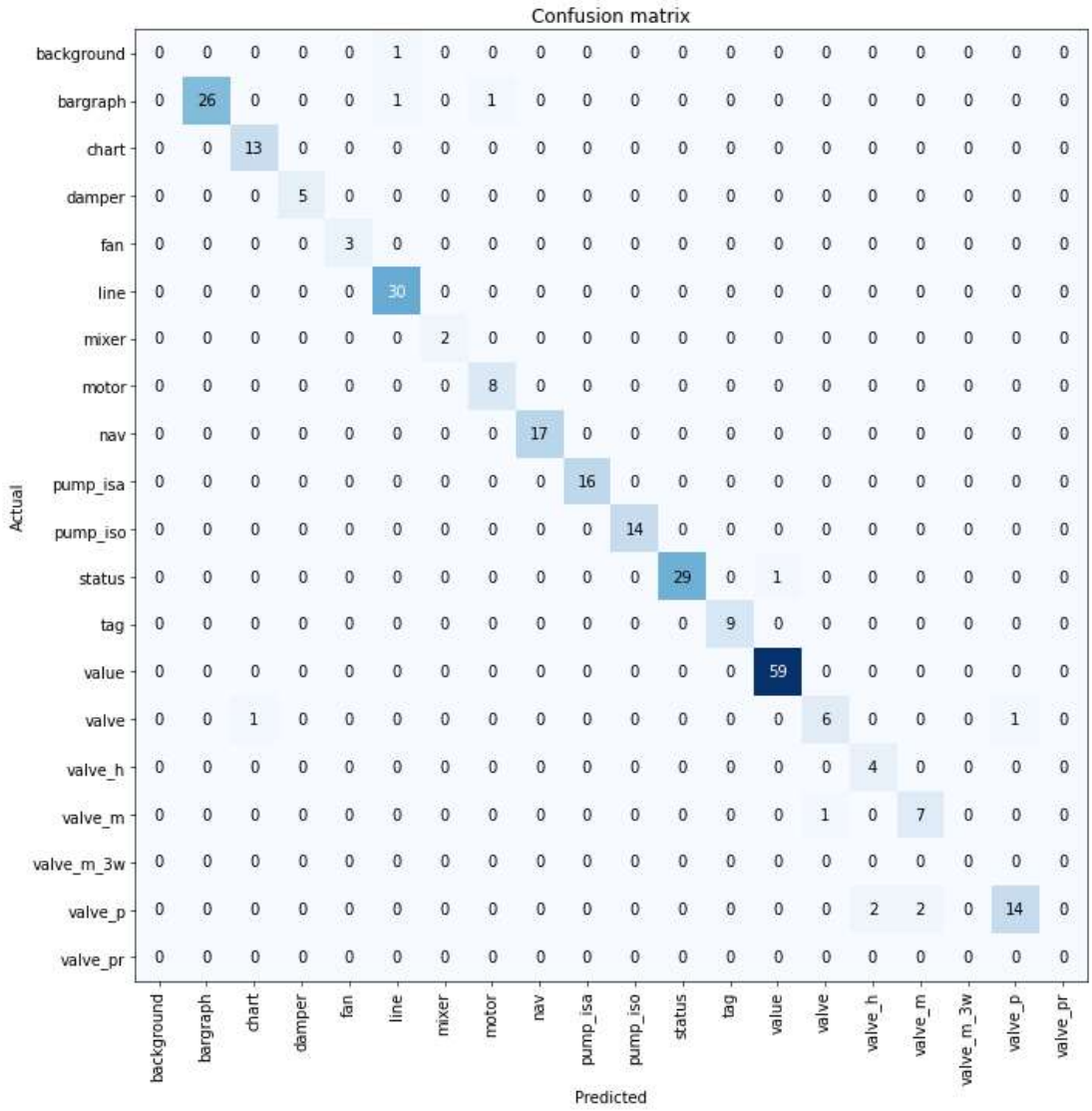
epoch	train_loss	valid_loss	accuracy	time
0	2.896408	0.995524	0.736264	00:14
1	1.646108	0.458982	0.868132	00:14
2	1.059790	0.362602	0.886447	00:14

epoch	train_loss	valid_loss	accuracy	time
0	0.278839	0.299870	0.890110	00:16
1	0.195774	0.238112	0.915751	00:16
2	0.161242	0.203671	0.937729	00:16
3	0.136923	0.095021	0.967033	00:16
4	0.109998	0.158871	0.963370	00:16
5	0.082578	0.104049	0.974359	00:16
6	0.065645	0.113539	0.959707	00:16
7	0.047687	0.122736	0.959707	00:16
8	0.041199	0.119036	0.959707	00:16

In [11]:
```
learn.recorder.plot_loss()
```



In [12]:
```
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
```



In [13]:
```
#interp.plot_top_losses(20, nrows=5)
interp.most_confused(min_val=1)
```

Out[13]:
```
[('valve_p', 'valve_h', 2),
 ('valve_p', 'valve_m', 2),
 ('background', 'line', 1),
 ('bargraph', 'line', 1),
 ('bargraph', 'motor', 1),
 ('status', 'value', 1),
 ('valve', 'chart', 1),
 ('valve', 'valve_p', 1),
 ('valve_m', 'valve', 1)]
```

Export model

In [14]:
```
filename = "classify_singl_pipeline1.pkl"
```

In [16]:
```
learn.export(fname=filename)
path = Path()
path.ls(file_exts='.pkl')
```

Out[16]:
```
(#2) [Path('classify_singl_pipeline1.pkl'),Path('simple_classifier_aug1.pkl')]
```

In [17]:
```
learn_inf = load_learner(path/filename)
```

In [18]:
```
num = 6
imP_path = Path('/home/engineirik/git/classify_singl_obj/test_img/'+
                str(num)+".png")
imP = Image.open(imP_path)
imP
```

Out[18]:


In [19]:
```
learn_inf.predict(imP_path)
```

Out[19]:
```
('pump_iso',
 TensorBase(10),
 TensorBase([1.6992e-08, 7.3694e-09, 2.5086e-08, 2.6695e-08, 5.7377e-07, 2.3314e-1
0, 9.6678e-08, 7.8886e-09, 1.8278e-08, 3.2490e-07, 1.0000e+00, 1.7124e-09, 1.9605e
-07, 6.4292e-08, 1.7851e-09,
             1.5711e-08, 1.0282e-06, 6.4365e-08, 5.8607e-08, 4.0035e-08]))
```

In [20]:
```
learn_inf.dls.vocab
```

Out[20]:
```
['background', 'bargraph', 'chart', 'damper', 'fan', 'line', 'mixer', 'motor', 'na
v', 'pump_isa', 'pump_iso', 'status', 'tag', 'value', 'valve', 'valve_h', 'valve_
m', 'valve_m_3w', 'valve_p', 'valve_pr']
```

In [ ]:

Appendix F
Multi-Label Classifier

Jupyter Notebook

Setup

In [1]:
```
from fastai.vision.all import *
from fastbook import *
from fastai.vision.widgets import *
from fastai.callback.fp16 import *
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
path = Path("/home/engineirik/git/classify_multi_obj")
Path.BASE_PATH = path
path.ls()
```

Out[2]:
```
(#13) [Path('rename_files_in_folder.ipynb'),Path('.git'),Path('.ipynb_checkpoint
s'),Path('v2_multiobj_classifier.pkl'),Path('folder_csv_generator.ipynb'),Path('mu
ltiobj_classifier.ipynb'),Path('v1_multiobj_classifier.pkl'),Path('test'),Path('it
ems.csv'),Path('copyfiles_from_to_folder.ipynb')...]
```

In [3]:
```
df = pd.read_csv(path/'items.csv')
df.head()
```

Out[3]:

	fname	labels	is_valid
0	pump_isa 8.png	pump_isa	False
1	pump_isa 61.png	pump_isa	True
2	pump_isa 45.png	pump_isa	False
3	pump_isa 69.bmp	pump_isa	False
4	pump_isa 2.png	pump_isa	False

In [4]:
```
df.iloc[:,0]
```

Out[4]:
```
0        pump_isa 8.png
1       pump_isa 61.png
2       pump_isa 45.png
3       pump_isa 69.bmp
4        pump_isa 2.png
             ...
1741      value 240.PNG
1742      value 114.png
1743       value 24.png
1744      value 209.bmp
1745       value 93.png
Name: fname, Length: 1746, dtype: object
```

In [5]:
```
df.iloc[0]
```

Out[5]:
```
fname       pump_isa 8.png
labels            pump_isa
is_valid             False
Name: 0, dtype: object
```

In [6]:
```
df['fname']
```

Out[6]:
```
0        pump_isa 8.png
1       pump_isa 61.png
2       pump_isa 45.png
3       pump_isa 69.bmp
4        pump_isa 2.png
            ...
1741      value 240.PNG
1742      value 114.png
1743       value 24.png
1744      value 209.bmp
1745       value 93.png
Name: fname, Length: 1746, dtype: object
```

Model and Preprocessing

In [7]:
```
def get_x(r): return path/'train'/r['fname']
def get_y(r): return r['labels'].split(' ')
# Check that it works
data = DataBlock(get_x = get_x, get_y = get_y)
dsets = data.datasets(df)
len(dsets.valid)
```

Out[7]:
```
349
```

In [8]:
```
# Check the tensors
data = DataBlock(blocks=(ImageBlock, MultiCategoryBlock),
                 get_x = get_x, get_y = get_y)
dsets = data.datasets(df)
dsets.train[52]
```

Out[8]:
```
(PILImage mode=RGB size=204x48,
 TensorMultiCategory([0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0.]))
```

In [9]:
```
# Check the vocabulary that the indexing is correct
idxs = torch.where(dsets.train[0][1]==1.)[0]
dsets.train.vocab[idxs]
```

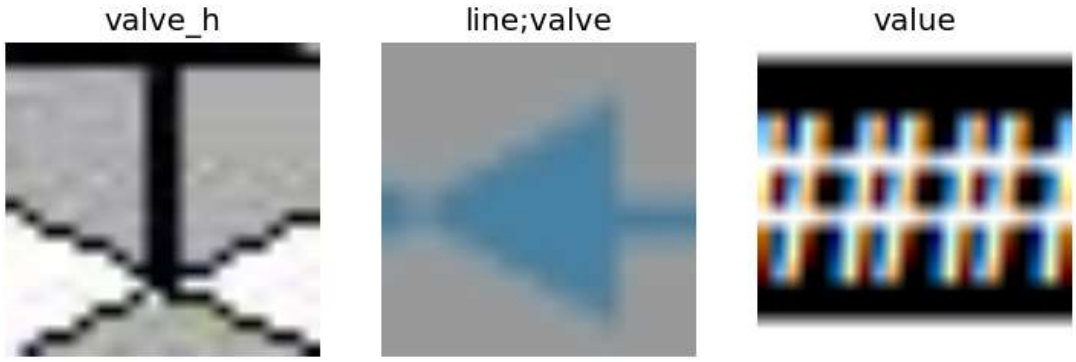
Out[9]:
```
(#1) ['bargraph']
```

In [10]:
```
# Define a test valid splitter
def splitter(df):
    train = df.index[~df['is_valid']].tolist()
    valid = df.index[df['is_valid']].tolist()
    return train,valid
```

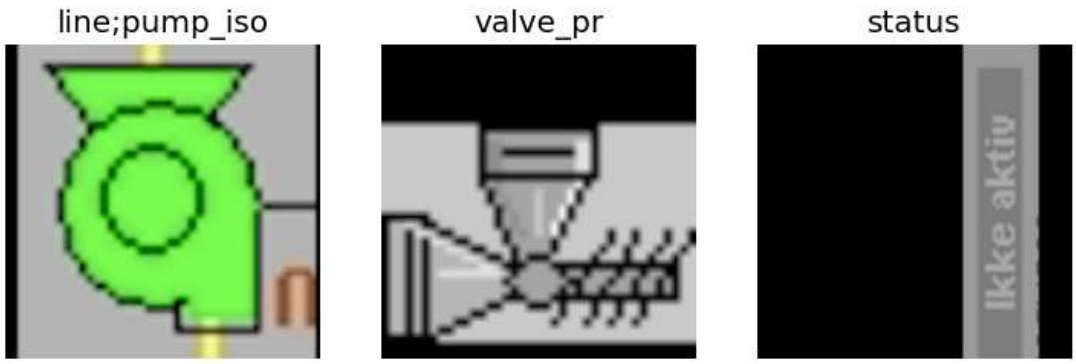
```
data = DataBlock(blocks=(ImageBlock, MultiCategoryBlock),
                 splitter=splitter,
                 get_x=get_x,
                 get_y=get_y)
dsets = data.datasets(df)
# Again, check the length to make sure it works
len(dsets.valid)
```

Out[10]: 326

In [11]:
```
data = DataBlock(blocks=(ImageBlock, MultiCategoryBlock),
                 splitter=splitter,
                 get_x=get_x,
                 get_y=get_y,
                 #item_tfms = Resize(128, ResizeMethod.Pad, pad_mode='zeros')
                 item_tfms = RandomResizedCrop(224, min_scale=0.35))
dls = data.dataloaders(df, bs=16)
dls.show_batch(nrows=1, ncols=3)
```



In [12]:
```
data = data.new(
    item_tfms=Resize(224, ResizeMethod.Pad, pad_mode='zeros'),
    batch_tfms=aug_transforms(size=224, min_scale=1, mult=2,
                              max_warp=0, do_flip=True,
                              flip_vert=True, max_zoom=1.0,
                              pad_mode="zeros", max_rotate=0)
)
dls = data.dataloaders(df, bs=16)
# Show batch, unique=true will return same object in dif augmentations
dls.show_batch(nrows=1, ncols=3)
#dls.valid.show_batch(max_n=3, nrows=2, unique=True)
```



Train

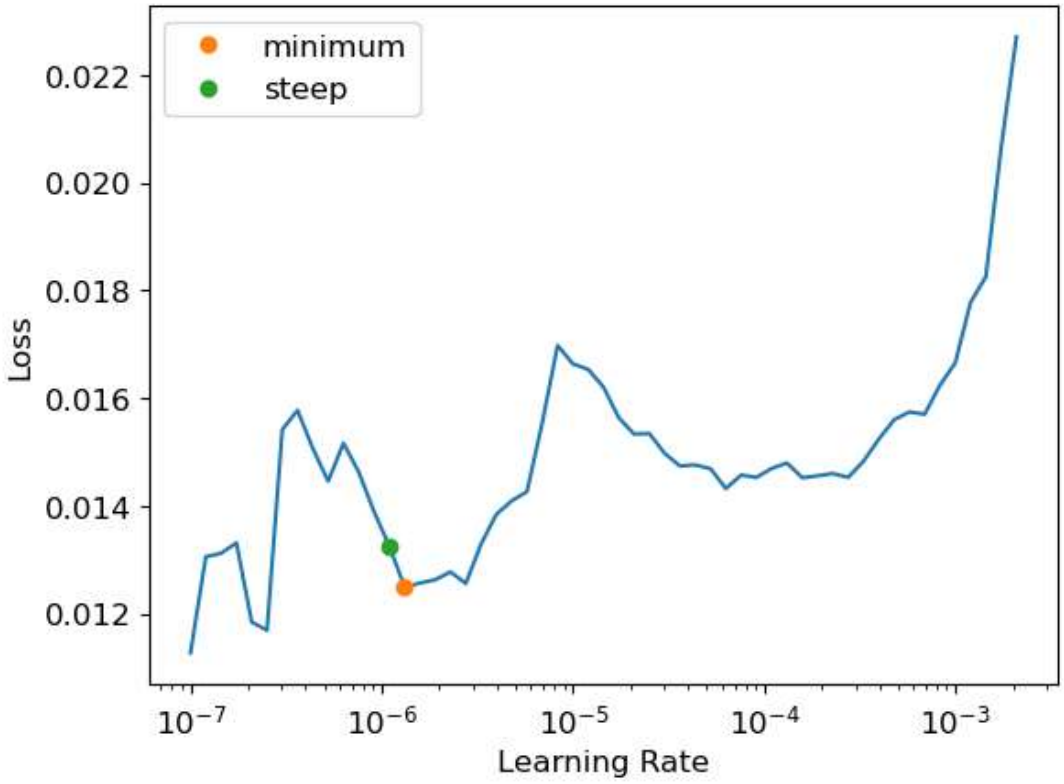
In [13]:
```
learn = vision_learner(dls, resnet50, metrics=partial(accuracy_multi, thresh=0.5))
learn.fine_tune(7, base_lr=3e-3, freeze_epochs=4)
```

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.934290	0.682573	0.615588	00:13
1	0.690487	0.355597	0.887479	00:11
2	0.273667	0.080241	0.971974	00:11
3	0.135597	0.061409	0.979643	00:12

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.079864	0.050636	0.983687	00:16
1	0.068066	0.045865	0.982989	00:16
2	0.051531	0.031916	0.988009	00:16
3	0.039019	0.025347	0.990101	00:16
4	0.029049	0.017391	0.994841	00:16
5	0.020180	0.015425	0.993726	00:16
6	0.017092	0.014262	0.994841	00:16

Learning rate finder

In [14]:
```
lr_min,lr_steep = learn.lr_find(suggest_funcs=(minimum, steep))
```



In [15]:
```
print(f"Minimum/10: {lr_min:.2e}, steepest point: {lr_steep:.2e}")
```

```
Minimum/10: 1.32e-07, steepest point: 1.10e-06
```

Threshold

In [16]:
```
# Check the learning threshold metrics
learn.metrics = partial(accuracy_multi, thresh=0.1)
learn.validate()
```

Out[16]:
```
(#2) [0.014261656440794468,0.9919130206108093]
```

In [17]:
```
learn.metrics = partial(accuracy_multi, thresh=0.99)
learn.validate()
```

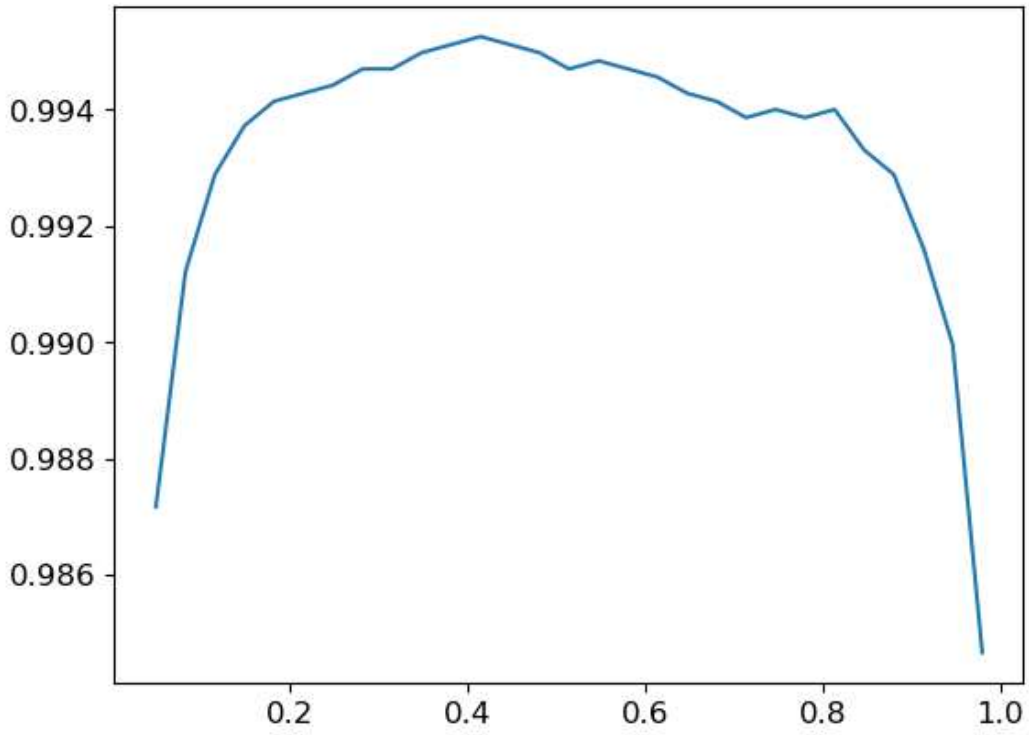
Out[17]:
```
(#2) [0.014261656440794468,0.9796431064605713]
```

In [18]:
```
preds,targs = learn.get_preds()
```

In [19]:
```
accuracy_multi(preds, targs, thresh=0.9, sigmoid=False)
#telling it to not apply activation function sigmoid
```

Out[19]: TensorBase(0.9921)

In [20]:
```
xs = torch.linspace(0.05,0.98,29)
accs = [accuracy_multi(preds, targs, thresh=i, sigmoid=False) for i in xs]
plt.plot(xs,accs);
```



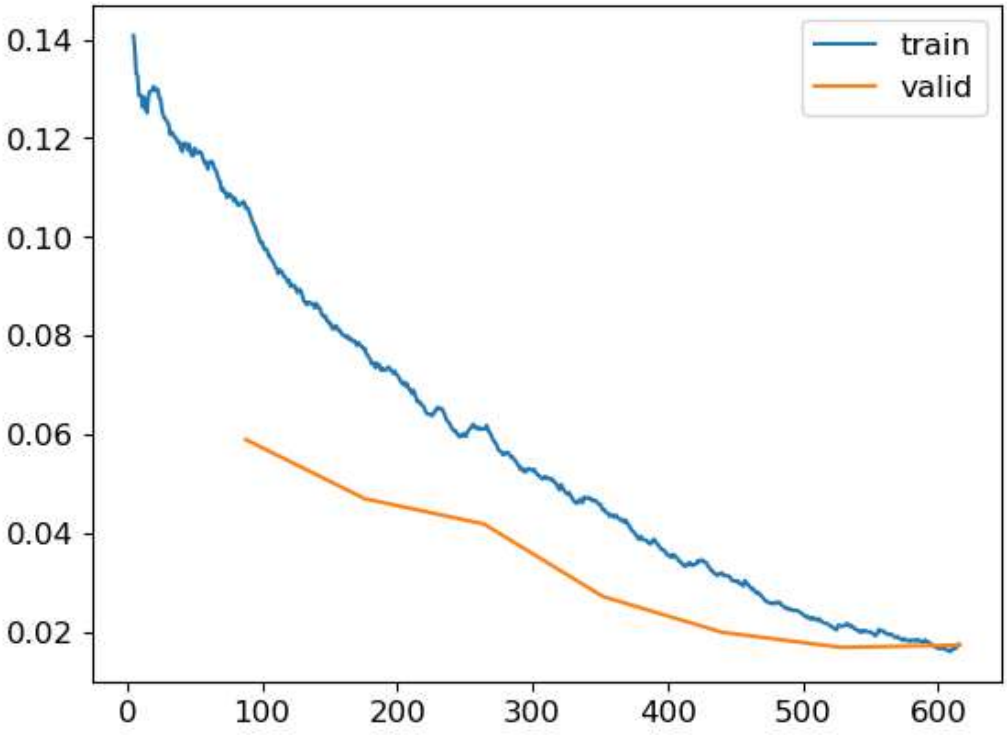
Re-Train

In [21]:
```
#Automatically change learningrate, select num freeze
learn = vision_learner(dls, resnet50, metrics=partial(accuracy_multi, thresh=0.8))
learn.fine_tune(7, base_lr=3e-03, freeze_epochs=4)

#changing the threshold between 90-100 does not really effect the result. Need more
```

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.938557	0.701670	0.906860	00:11
1	0.699822	0.371826	0.969883	00:11
2	0.272820	0.086045	0.967931	00:11
3	0.131944	0.065084	0.974205	00:11

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.079951	0.054881	0.977831	00:16
1	0.065731	0.044130	0.981735	00:16
2	0.055285	0.034544	0.987172	00:16
3	0.039843	0.025235	0.989961	00:16
4	0.029929	0.020573	0.992331	00:16
5	0.021101	0.018158	0.993029	00:16
6	0.016511	0.017393	0.993865	00:16

In [19]:
```
learn.recorder.plot_loss()
```



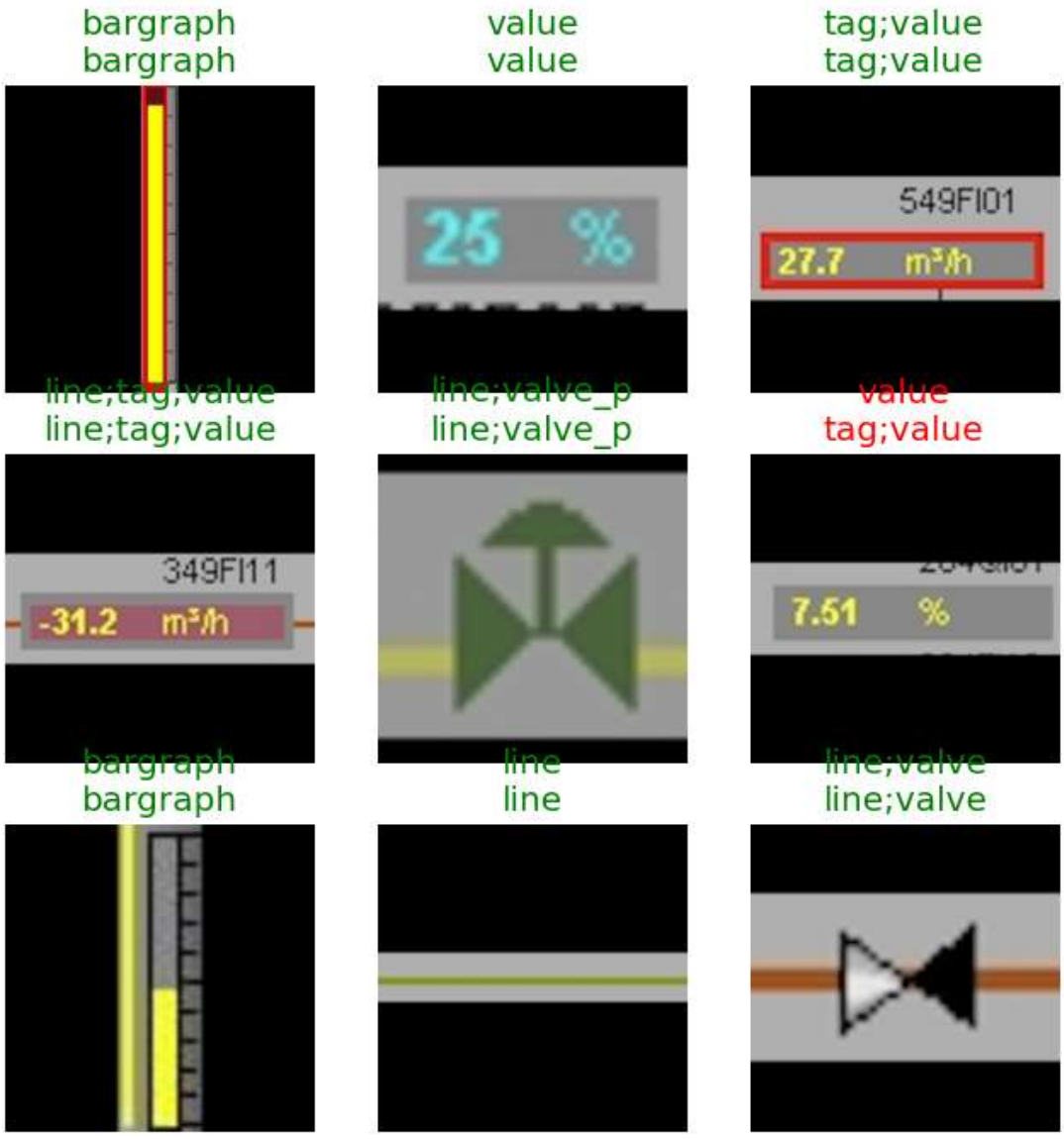
Manually validate training

In [22]:
```
interp = ClassificationInterpretation.from_learner(learn)
#interp.plot_multi_top_losses(figsz(12,12), dpi=60)
#interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
```

In [23]:
```
#interp.plot_top_losses(20, nrows=5)
interp.most_confused(min_val=1)
```

Out[23]:
```
[('bargraph', 'background', 29), ('background', 'bargraph', 17)]
```

In [24]:
```
learn.show_results(ds_idx=1, nrows=3, figsize=(8,8))
```



I would argue that the missclassification abow is actually correct as there actually is a tag in the value object (column 3, row 2). So tag;value classification is correct.

Export model

In [25]:
```
learn.export(fname="v5_multiobj_classifier.pkl")
```

In [26]:
```
#Learn.export??
```

In [27]:
```
path = Path()
path.ls(file_exts='.pkl')
```

Out[27]:
```
(#3) [Path('v2_multiobj_classifier.pkl'),Path('v1_multiobj_classifier.pkl'),Path
('v5_multiobj_classifier.pkl')]
```

In [28]:
```
learn_inf = torch.load(path/'v5_multiobj_classifier.pkl', map_location='cuda:0')
#learn_inf = load_learner(path/'v1_multiobj_classifier.pkl').cuda()
```

In [29]:
```
num = 42
imP_path = Path('/home/engineirik/git/classify_multi_obj/test/'+str(num)+".PNG")
imP = Image.open(imP_path)
imP
```

Out[29]:


In [30]:
```
learn_inf.predict(imP_path)
```

Out[30]:
```
((#2) ['pump_isa','tag'],
 TensorBase([False, False, False, False, False, False, False, False, False,  True,
False, False,  True, False, False, False, False, False, False, False, False, Fals
e]),
 TensorBase([1.3426e-03, 1.7562e-03, 6.2061e-04, 2.5043e-03, 5.1339e-03, 2.6294e-0
1, 2.2042e-03, 3.2258e-03, 9.4121e-04, 7.2583e-01, 3.1395e-02, 1.9519e-02, 8.4550e
-01, 8.5509e-02, 3.9538e-04,
             3.8589e-04, 6.0479e-03, 9.7700e-04, 6.3403e-02, 3.1402e-03, 1.7011e-0
3, 1.9481e-03]))
```

In [27]:
```
learn_inf.dls.vocab
```

Out[27]:
```
['background', 'bargraph', 'chart', 'damper', 'fan', 'line', 'mixer', 'motor', 'na
v', 'pump_isa', 'pump_iso', 'status', 'tag', 'value', 'valve', 'valve_3w', 'valve_
h', 'valve_h_3w', 'valve_m', 'valve_m_3w', 'valve_p', 'valve_pr']
```

Save model

In [ ]:
```

```

Appendix G
Pyramid Scaled Sliding Window NMS Classifier

Jupyter Notebook

In [ ]:
```
import cv2
import numpy as np
from fastai.vision.all import *
from fastbook import *
from fastai.vision.widgets import *
from fastai.callback.fp16 import *
import pickle
import argparse
import imutils
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'
import pytesseract
import keras_ocr
import math
import warnings
warnings.filterwarnings("ignore")
```

In this code, the sliding_window function generates windows of a certain window_size over an input image, with a step of step_size. The classify_image function uses the sliding_window function to generate windows, resizes them to the expected size for the pre-trained model, and classifies each window using the model.predict method. The classified windows are stored in the classified_regions list and returned.

In [9]:
```
# Define paths
path = Path("/home/engineirik/git/classify_multi_obj")
imP_path = Path('/home/engineirik/git/classify_obj/OperateDisplay/')
#imP_path.ls()
```

Out[9]:
```
(#2) [Path('/home/engineirik/git/classify_obj/OperateDisplay/2022_02_08_PR_276_10_SYRE_EKSP_LAGER.jpg'),Path('/home/engineirik/git/classify_obj/OperateDisplay/2022_02_08_VL_310_10_TANKER.jpg')]
```

In [10]:
```
# Defined to remove attribute error in model
def get_x(r): return path/'train'/r['fname']
def get_y(r): return r['labels'].split(' ')
```

In [11]:
```
# Load model
learn_inf = torch.load(path/'v2_multiobj_classifier.pkl', map_location='cuda:0')
# Define the window size and step size
#window_size_hz = (80,30)
#window_size_vc = (30,80)
window_size_sq = (40,40)
step_size = (13,13) #1313
WIDTH = 1800
PYR_SCALE = 1.5

# Load the input image
image = cv2.imread(str(imP_path)+'/2022_02_08_PR_276_10_SYRE_EKSP_LAGER.jpg')
image = imutils.resize(image, width=WIDTH)
orig = image.copy()
(H, W) = image.shape[:2]
```

In [12]:
```
# Function for showing the image in jupyter notebook
def show_rgb_image(image, title=None, conversion=cv2.COLOR_BGR2RGB):

    # Converts from one colour space to the other. this is needed as RGB
    # is not the default colour space for OpenCV
    image = cv2.cvtColor(image, conversion)

    # Show the image
    plt.imshow(image)

    # remove the axis / ticks for a clean looking image
    plt.xticks([])
    plt.yticks([])

    # if a title is provided, show it
    if title is not None:
        plt.title(title)

    plt.show()
```

In [13]:
```
# Moves sliding windows
def sliding_window(image, step_size, window_size):
    for y in range(0, image.shape[0]-window_size[1], step_size[1]):
        for x in range(0, image.shape[1]-window_size[0], step_size[0]):
            yield (x, y, image[y:y + window_size[1], x:x + window_size[0]])
```

In [14]:
```
# Scales the image in given pyramid scales
def image_pyramid(image, scale=1.5, minSize=(128, 128)):
    # yield the original image
    yield image
    # keep looping over the image pyramid
    while True:
        # compute the dimensions of the next image in the pyramid
        w = int(image.shape[1] / scale)
        image = imutils.resize(image, width=w)
        # if the resized image does not meet the supplied minimum
        # size, then stop constructing the pyramid
        if image.shape[0] < minSize[1] or image.shape[1] < minSize[0]:
            break
        # yield the next image in the pyramid
        yield image
```

In [16]:
```
# Initialize the image pyramid
pyramid = image_pyramid(image, scale=PYR_SCALE, minSize=window_size_sq)
```

In [17]:
```
rois = []
locs = []
# Classify the image
for image in pyramid:

    # determine the scale factor between the *original* image
    # dimensions and the *current* layer of the pyramid
    scale = W / float(image.shape[1])
```

```
    for (x, y, window) in sliding_window(image, step_size, window_size_sq):
        if window.shape[0] != window_size_sq[1] or window.shape[1]
        != window_size_sq[0]:
            continue

        # scale the (x, y)-coordinates of the ROI with respect to the
        # *original* image dimensions
        x = int(x * scale)
        y = int(y * scale)
        w = int(window_size_sq[0] * scale)
        h = int(window_size_sq[1] * scale)

        # Resize the window to the size expected by the model
        window = cv2.resize(window, (224,224))
        window = np.array(window)
        rois.append(window)
        locs.append((x, y, x + w, y + h))
```

In [18]:
```
# Load data and predict using the Multi-label classifier model
test_dl = learn_inf.dls.test_dl(rois)
preds = learn_inf.get_preds(dl=test_dl)
```

In [19]:
```
labels = learn_inf.dls.vocab
label = []
score = []
classified_regions = []
for i in range(len(preds[0])):
    x1,y1,x2,y2 = locs[i]
    label = (labels[preds[0][i].argmax()]) #.argmax orig
    score = (preds[0][i].max())
    classified_regions.append((x1, y1, x2, y2, label, score))
#score
#locs
#classified_regions[2]
```

In [20]:
```
# Number of classifications
len(classified_regions)
```

Out[20]: 14416

To merge similar bounding boxes into one bounding box with one label, you can use a technique called non-maximum suppression. The idea is to compare each bounding box with all other bounding boxes and remove those that have a high overlap with another bounding box.

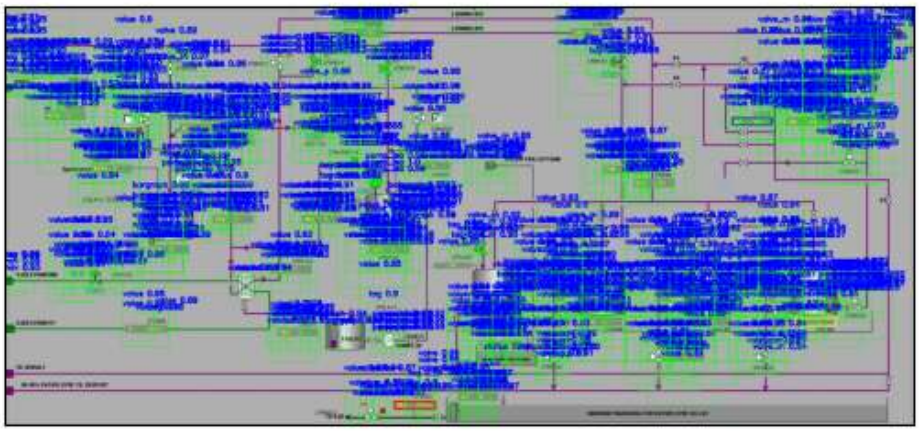
In [31]:
```
# Used for NMS, merging/removing overlapping boxes with low score
def merge_bounding_boxes(bboxes, scores, scoreThreshold=0.1, nms_threshold=0.1):
    # create a list to store the indices of the bounding boxes to keep
    keep = []
    # Convert the bounding boxes to a format that can be used by the
    # cv2.dnn.NMSBoxes function
    #bboxes = [box.astype("int") for box in bboxes]
```

```
    bboxes = [np.around(box).astype("int") for box in bboxes]
    # use the cv2.dnn.NMSBoxes function to suppress overlapping bounding boxes
    scores = np.array(scores, dtype="float")
    indices = cv2.dnn.softNMSBoxes(bboxes, scores, scoreThreshold, nms_threshold)
    # keep the indices of the bounding boxes that were not suppressed
    for i in indices:
        keep.append(i)
    # return the indices of the bounding boxes to keep
    return keep
```

No NMS

In [32]:
```
# Draw bounding boxes around the classified regions, show all without NMS
boxes = []
scores = []
labels = []

copy3_image = orig.copy()

for (x1, y1, x2, y2, label, score) in classified_regions:
    if score >= 0.8 and label != "chart" and label != "line" and label != "arrow":
        boxes.append([x1, y1, x2, y2])
        scores.append(score)
        labels.append(label)
        cv2.rectangle(copy3_image, (x1,y1), (x2,y2), (0,255,0), 1)
        cv2.putText(copy3_image, str(label)+" "+str(round(float(score), 2)),
                    (x1, y1-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
```

In [33]:
```
cv2.imwrite("v2_NONMS.jpg", copy3_image)
show_rgb_image(copy3_image)
```



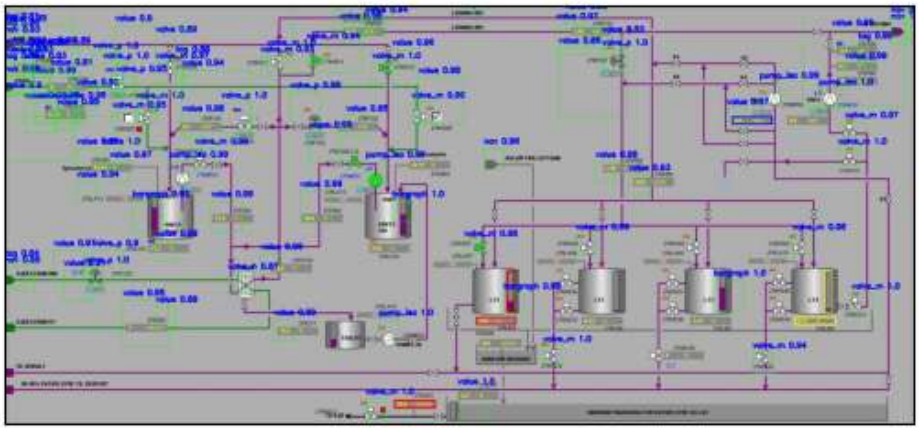
NMS

In [34]:
```
# Copy boxes before NMS
keep = merge_bounding_boxes(boxes, scores)
bboxes = [boxes[i] for i in keep[1]]
```

```
bscores = [scores[i] for i in keep[1]]
bblabel = [labels[i] for i in keep[1]]
bscores = np.array(bscores, dtype="float")
```

In [35]:
```
# Draw bounding boxes around the classified regions, only the one with heighest sco
copy4_image = orig.copy()
i = 0
for (x1, y1, x2, y2) in bboxes:
    cv2.rectangle(copy4_image, (x1,y1), (x2,y2), (0,255,0), 1)
    cv2.putText(copy4_image, str(bblabel[i])+" "+str(round(bscores[i], 2)),
                (x1, y1-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
    i+=1
```

In [36]:
```
cv2.imwrite("v2_NMS.jpg", copy4_image)
show_rgb_image(copy4_image)
```



NMS label

In [37]:
```
# Group the boxes by label
grouped_boxes = defaultdict(list)
for box, label in zip(boxes, labels):
    grouped_boxes[label].append(box)

print(len(grouped_boxes))
```

```
10
```

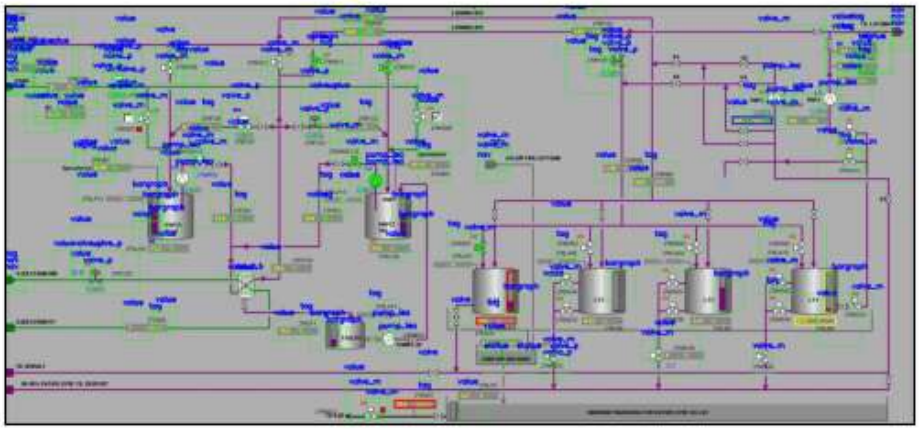
In [38]:
```
# Group the scores by boxes
grouped_scores = defaultdict(list)
for score, label in zip(scores, labels):
    grouped_scores[label].append(score)

print(len(grouped_scores))
```

```
10
```

In [39]:
```
# Perform NMS on each group
copy5_image = orig.copy()
```

```
for label, boxes in grouped_boxes.items():
    result = []
    scores = []

    #scores = [0.95] * len(boxes)  # score of each box, set to 1.0 for simplicity
    scores = grouped_scores[label]
    scores = np.array(scores, dtype="float")
    indices = cv2.dnn.softNMSBoxes(boxes, scores, score_threshold=0.1, nms_threshol

    #print(scores)
    result.extend([boxes[i] for i in indices[1]])
    for (x1, y1, x2, y2) in result:
        cv2.rectangle(copy5_image, (x1,y1), (x2,y2), (0,255,0), 1)
        cv2.putText(copy5_image, label, (x1, y1-10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
```

In [40]:
```
cv2.imwrite("v2_LNMS.jpg", copy5_image)
show_rgb_image(copy5_image)
```



Appendix I
Split Image Annotation YOLO Prep

Jupyter Notebook

In [ ]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os
```

In [ ]:
```
def show_rgb_image(image, title=None, conversion=cv2.COLOR_BGR2RGB):
    # Converts from one colour space to the other. this is needed as RGB
    # is not the default colour space for OpenCV
    image = cv2.cvtColor(image, conversion)
    # Show the image
    plt.imshow(image)
    # remove the axis / ticks for a clean looking image
    plt.xticks([])
    plt.yticks([])
    # if a title is provided, show it
    if title is not None:
        plt.title(title)
    plt.show()
```

In [ ]:
```
def draw_annotation(image, annotation_file):
    img_height, img_width = image.shape[:2]
    with open(annotation_file, 'r') as f:
        annotation = f.readlines()
    for line in annotation:
        data = line.split()
        x_center, y_center, w, h = map(float, data[1:])
        # Convert normalized coordinates to pixel coordinates
        x = int((x_center - w/2) * img_width)
        y = int((y_center - h/2) * img_height)
        w = int(w * img_width)
        h = int(h * img_height)
        # Draw bounding box
        cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

In [ ]:
```
def split_image(img_path, annotation_path, img_output_path, ann_output_path):
    # Load image and annotation data
    img = cv2.imread(img_path)
    with open(annotation_path, 'r') as f:
        annotation = f.readlines()

    # Get image width and height
    img_height, img_width = img.shape[:2]

    # Split image in half
    left_img = img[:, :img_width//2, :]
    right_img = img[:, img_width//2:, :]
    c_left_img = left_img.copy()
    c_right_img = right_img.copy()

    # Calculate new image widths
    left_width = left_img.shape[1]
    right_width = right_img.shape[1]

    # Split annotation data accordingly
    left_annotation = []
    right_annotation = []
    for line in annotation:
```

```
        data = line.split()
        x_center, y_center, w, h = map(float, data[1:])
        x = (x_center - w/2) * img_width
        y = (y_center - h/2) * img_height
        w = w * img_width
        h = h * img_height
        x_center_p = x_center * img_width
        y_center_p = y_center * img_height

        #x_center /= img_width  # divide by full image width
        #y_center /= img_height  # divide by full image height
        #w /= img_width  # divide by full image width
        #h /= img_height  # divide by full image height

        if x + w < 0.5*img_width:
            left_annotation.append('{} {:.6f} {:.6f} {:.6f} {:.6f}\n'.format
                                   (data[0],
                                    x_center_p/left_width,
                                    y_center_p/img_height,
                                    w/left_width,
                                    h/img_height))
        elif x >= 0.5*img_width:
            right_annotation.append('{} {:.6f} {:.6f} {:.6f} {:.6f}\n'.format
                                    (data[0],
                                     (x_center_p-left_width)/right_width,
                                     y_center_p/img_height,
                                     w/right_width,
                                     h/img_height))
        else:
            # Annotation is split in half, need to update coordinates
            if x < 0.5*img_width:
                x_left = x
                w_left = (0.5*img_width)-x
                if w_left > 0.5*w:
                    x_center_p = (x_left + w_left/2)
                    y_center_p = y_center * img_height
                    left_annotation.append('{} {:.6f} {:.6f} {:.6f} {:.6f}\n'
                                           .format
                                           (data[0], x_center_p/left_width,
                                            y_center_p/img_height,
                                            w_left/left_width, h/img_height))

            if x + w > 0.5*img_width:
                x_right = 0
                w_right = ((x + w) - (left_width))
                if w_right > 0.5*w:
                    x_center_p = (w_right / 2)
                    right_annotation.append('{} {:.6f} {:.6f} {:.6f} {:.6f}\n'
                                            .format
                                            (data[0], x_center_p/right_width,
                                             y_center_p/img_height,
                                             w_right/right_width, h/img_height))

            #if x < 0.5*img_width and x + w > 0.5*img_width:
                #left_annotation.append(line)
                #right_annotation.append(line)


    # Save split images and annotations
    img_filename = os.path.basename(img_path)
```

```
    img_basename, img_extension = os.path.splitext(img_filename)
    left_img_path = os.path.join(img_output_path, img_basename +
                                 '_left' + img_extension)
    right_img_path = os.path.join(img_output_path, img_basename +
                                  '_right' + img_extension)
    cv2.imwrite(left_img_path, left_img)
    cv2.imwrite(right_img_path, right_img)

    ann_filename = os.path.basename(annotation_path)
    ann_basename, ann_extension = os.path.splitext(ann_filename)
    left_ann_path = os.path.join(ann_output_path, ann_basename +
                                 '_left' + ann_extension)
    right_ann_path = os.path.join(ann_output_path, ann_basename +
                                  '_right' + ann_extension)
    with open(left_ann_path, 'w') as f:
        f.writelines(left_annotation)
    with open(right_ann_path, 'w') as f:
        f.writelines(right_annotation)

    # Draw annotations on split images
    draw_annotation(c_left_img, left_ann_path)
    draw_annotation(c_right_img, right_ann_path)

    # Show split images with annotations
    show_rgb_image(c_left_img, title=img_path[:-4]+'_left')
    show_rgb_image(c_right_img, title=img_path[:-4]+'_right')
```

In [ ]:
```
# Splitt just one image
#img = "yolo/train/images/2022_02_08_AB_268_10_SYRE_ABS_analyzed.png"
#ann = "yolo/train/labels/2022_02_08_AB_268_10_SYRE_ABS_analyzed.txt"
#split_image(img, ann)
```

In [ ]:
```
img_dir = "yolo/train/images"
ann_dir = "yolo/train/labels"
img_split_dir = "yolo/train/images_s"
ann_split_dir = "yolo/train/labels_s"

# Get lists of image and annotation file paths
img_files = os.listdir(img_dir)
ann_files = os.listdir(ann_dir)
```

In [ ]:
```
# Loop through image files and call split_image on corresponding annotation file
for img_file in img_files:
    if img_file.endswith('.png'):
        # Get corresponding annotation file
        ann_file = img_file.replace('.png', '.txt')
        if ann_file in ann_files:
            img_path = os.path.join(img_dir, img_file)
            ann_path = os.path.join(ann_dir, ann_file)
            split_image(img_path, ann_path, img_split_dir, ann_split_dir)
```

Appendix J
OCR Prep

Python code

```
import re
import cv2
import numpy as np
import pytesseract
from PIL import Image

img_path = 'img/111.jpg'

# Define a list of regular expression patterns to match the desired formats
"""patterns = [r'\d{3}[a-zA-Z]+\d{2}',
            r'[a-zA-Z]+-\d{4}',
            r'[a-zA-Z]?\d{4}',
            r'[a-zA-Z]{2}-\d{4}',
            r'\d{3},\d{2},\d{2}'
            ]"""

patterns = [
    r'\d{3}[a-zA-Z]+\d{2}',
    r'[a-zA-Z]+-\d{4}',
    r'[a-zA-Z]?\d{4}',
    r'[a-zA-Z]{2}-\d{4}',
    r'\d{3}[, ]\d{1}[a-zA-Z]+[, ]\d{2}',
    r'\d{3}[, ]\d{1}[a-zA-Z]+[, ]\d{2}[a-zA-Z]+',
    r'\d{3}[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-
Z]+[, ]\d{2}[a-zA-Z]+',
    r'\d{3}[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-
Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+',
    r'\d{3}[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-
Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[,
]\d{2}[a-zA-Z]+',
    r'\d{3}[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-
Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[,
]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-
zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{1}[a-zA-Z]+[, ]\d{2}[a-zA-Z]+',
    r'\d{3},\d{2},\d{2}',
    r"\d{3}[A-Za-z]{3}\d{2}",
]


# Set the path to the tesseract executable
pytesseract.pytesseract.tesseract_cmd = '/usr/bin/tesseract'

# Read the image using cv2.imread
image = cv2.imread(img_path)
```

```
# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Get the height and width of the image
height, width = gray.shape[:2]

# Divide the height and width by 2 to get the dimensions of each quadrant
h = height // 2
w = width // 2

# Create an array to store the 4 quadrants of the image
quadrants = [gray[:h, :w], gray[:h, w:], gray[h:, :w], gray[h:, w:]]

# Create a list to store the annotations
annotations = []

# Set the minimum confidence level to 50%
conf_level = 10

# Set the Pytesseract configuration parameters
config = f"--psm 6 --oem 3 -c min_confidence_level={conf_level}"

# Define the scaling factors
scales = [1.5, 2, 4]

def get_range(threshold, sigma=0.33):
    return (1-sigma) * threshold, (1+sigma) * threshold

for scale in scales:
    # Loop over the quadrants
    for j, quadrant in enumerate(quadrants):
        # Randomly scale and rotate the image
        upscaled = cv2.resize(quadrant, None, fx=scale, fy=scale,
interpolation=cv2.INTER_LINEAR)

        #upscaled = cv2.resize(scaled, None, fx=4, fy=4,
interpolation=cv2.INTER_LINEAR)
        q_height, q_width = upscaled.shape[:2]
        # Apply a Laplacian filter to sharpen the image
        laplacian = cv2.Laplacian(upscaled, cv2.CV_8U) #test
        sharpened = cv2.addWeighted(upscaled, 1.5, laplacian, -0.5, 0) #test

        # Apply thresholding to create a binary image
        thresh = cv2.threshold(sharpened, 170, 255, cv2.THRESH_BINARY_INV)[1]
```

```
        # Apply kernel to dilate the image
        kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (1,1))
        # Invert the Canny edges image
        edges_inverted = cv2.bitwise_not(thresh)
        # Apply dilation to make text more visible
        dilated = cv2.dilate(edges_inverted, kernel, iterations=1)

        cv2.imwrite(f'img/quadrant_{j}.jpg', dilated)

        # Use pytesseract to extract text and bounding boxes from the image
        data = pytesseract.image_to_data(dilated,
output_type=pytesseract.Output.DICT, config=config, lang=None) #config=config)
        #print(data['text'])

        # Loop over the words and concatenate bounding boxes that are close
together
        for i in range(len(data['text'])):
            # Extract the text and bounding box coordinates
            text = data['text'][i]
            x, y, w, h = data['left'][i], data['top'][i], data['width'][i],
data['height'][i]

            # Apply the scaling factor used in the loop
            x = x / (scale)
            y = y / (scale)
            w = w / (scale)
            h = h / (scale)

            # Rescale the coordinates and dimensions of the bounding boxes
            if j == 0:  # Top-left quadrant
                x_offset = 0
                y_offset = 0
            elif j == 1:  # Top-right quadrant
                x_offset = width/2
                y_offset = 0
            elif j == 2:  # Bottom-left quadrant
                x_offset = 0
                y_offset = height/2
            else:  # Bottom-right quadrant
                x_offset = width/2
                y_offset = height/2

            x_center = (x + x_offset) / width
            y_center = (y + y_offset) / height
            box_width = w / width
```

```
            box_height = h / height

            if not text:
                continue
            if len(text) < 3:
                continue
            #if text.replace(" ", "") == "":
                #continue

            # Check if the text matches any of the desired patterns
            matches_pattern = False

            for pattern in patterns:
                if re.match(pattern, text):
                    matches_pattern = True
                    break

            if not matches_pattern:
                if re.match(r'\d{3}', text):
                    if i+1 < len(data['text']):
                        text2 = data['text'][i+1]
                        if re.match(r',\d{2}', text2):
                            if i+2 < len(data['text']):
                                text3 = data['text'][i+2]
                                if re.match(r'\d{2}-\d{2}', text3):
                                    text = text + text2 + "_" + text3
                                    matches_pattern = True
                        elif re.match(r',\d{2},\d{2}', text2):
                            if i+2 < len(data['text']):
                                text3 = data['text'][i+2]
                                if re.match(r'\d{2}-\d{2}', text3):
                                    text = text + text2 + "_" + text3
                                    matches_pattern = True
                        elif re.match(r'\d{2}', text2):
                            if i+2 < len(data['text']):
                                text3 = data['text'][i+2]
                                if re.match(r'\d{4}-\d{2}', text3):
                                    text = text + "," + text2 + "_" + text3
                                    matches_pattern = True
                        elif re.match(r',\d{2},\d{2}', text2):
                            text = text + text2
                            matches_pattern = True
                elif re.match(r'\d{3},\d{2},\d{2}', text):
                    if i+1 < len(data['text']):
                        text2 = data['text'][i+1]
```

```
                        if re.match(r'\d{2}-\d{2}', text2):
                            text = text + "_" + text2
                            matches_pattern = True
                    elif re.match(r'\d{3},\d{2}', text):
                        if i+1 < len(data['text']):
                            text2 = data['text'][i+1]
                            if re.match(r',\d{2}', text2):
                                if i+2 < len(data['text']):
                                    text3 = data['text'][i+2]
                                    if re.match(r'\d{2}-\d{2}', text3):
                                        text = text + text2 + "_" + text3
                                        matches_pattern = True
                            elif re.match(r'\d{2}', text2):
                                if i+2 < len(data['text']):
                                    text3 = data['text'][i+2]
                                    if re.match(r'\d{2}-\d{2}', text3):
                                        text = text + "," + text2 + "_" + text3
                                        matches_pattern = True


                if not matches_pattern:
                    continue

                if any(text in annotation for annotation in annotations):
                    continue
                else:
                    print(text + " " + str(j))
                    # Add the annotation to the list
                    annotations.append(f"{text} {x_center:.6f} {y_center:.6f}
{box_width:.6f} {box_height:.6f}")

# Save the image with the bounding boxes
img.save('image_with_boxes.jpg')

# Save the annotations to a text file
with open('annotations.txt', 'w') as f:
    f.write('\n'.join(annotations))

# Copy image
copy_img = image.copy()

# Load the bounding box data from the text file CHECK
with open("/home/engineirik/git/ocr/annotations.txt") as f:
    lines = f.readlines()[1:] # Skip the header line
    for line in lines:
```

```
        cols = line.strip().split()
        x, y, w, h = map(float, cols[1:5])

        # Scale the coordinates to the image size
        x = x * width
        y = y * height
        w = w * width
        h = h * height

        # Draw a rectangle around the object
        cv2.rectangle(copy_img, (int(x), int(y)), (int(x+w), int(y+h)), (0, 255,
0), 2)


# Display the image
cv2.imshow("Image with bounding boxes", copy_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Appendix L
Custom mAP Calculation

Python code

```
import numpy as np
import os

def compute_iou(box1, box2):
    # Calculate the intersection rectangle
    x1 = max(box1[0], box2[0])
    y1 = max(box1[1], box2[1])
    x2 = min(box1[0]+box1[2], box2[0]+box2[2])
    y2 = min(box1[1]+box1[3], box2[1]+box2[3])
    inter_area = max(0, x2-x1) * max(0, y2-y1)

    # Calculate the union area
    box1_area = box1[2] * box1[3]
    box2_area = box2[2] * box2[3]
    union_area = box1_area + box2_area - inter_area

    # Calculate the IoU
    iou = inter_area / union_area

    return iou

def compute_precision_recall(yolo_data, annotated_data, class_id, iou_threshold):
    tp = 0
    fp = 0
    fn = 0

    num_annotated_objects = np.sum(annotated_data[:, 0] == class_id)

    for i in range(len(yolo_data)):
        if yolo_data[i][0] != class_id:
            continue

        yolo_box = [yolo_data[i][1], yolo_data[i][2], yolo_data[i][3], yolo_data[i][4]]
        max_iou = 0
        for j in range(len(annotated_data)):
            if annotated_data[j][0] != class_id:
                continue
```

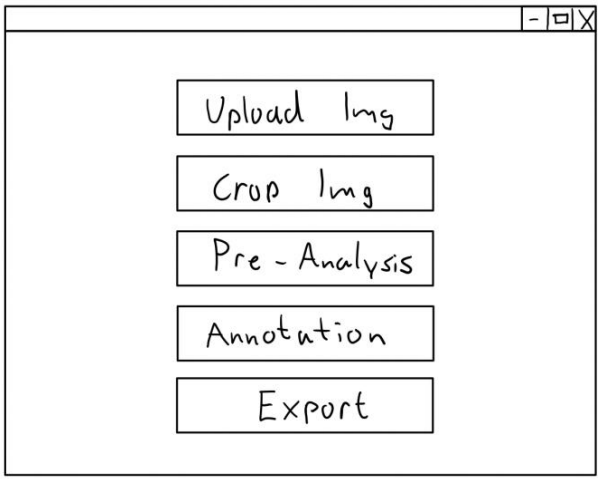
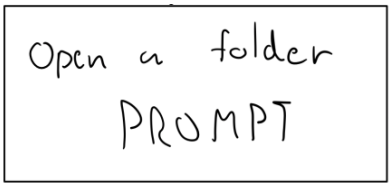
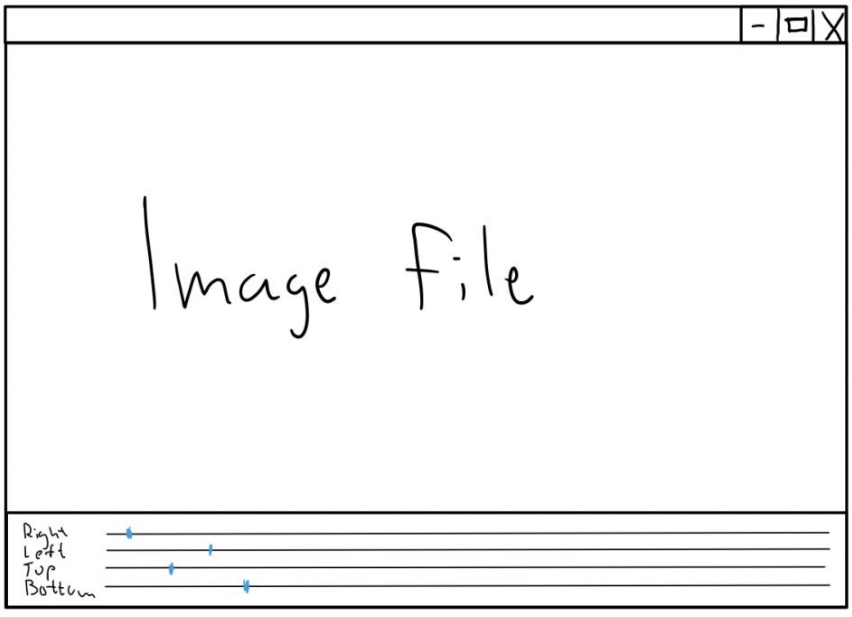
```
            annotated_box = [annotated_data[j][1], annotated_data[j][2], annotated_data[j][3], annotated_data[j][4]]
            iou = compute_iou(yolo_box, annotated_box)
            if iou > max_iou:
                max_iou = iou

        if max_iou >= iou_threshold:
            tp += 1
        else:
            fp += 1

    fn = num_annotated_objects - tp

    if tp + fp > 0:
        precision = tp / (tp + fp)
    else:
        precision = 0

    recall = tp / (tp + fn)

    return precision, recall


def compute_mAP(yolo_file, annotated_file, iou_threshold=0.50):
    yolo_data = np.loadtxt(yolo_file, delimiter=' ')
    annotated_data = np.loadtxt(annotated_file, delimiter=' ')
    class_ids = np.unique(annotated_data[:, 0])
    num_classes = len(class_ids)

    aps = []
    for i, class_id in enumerate(class_ids):
        precision, recall = compute_precision_recall(yolo_data, annotated_data, class_id, iou_threshold)

        ap = 0
        for j in range(11):
            threshold = j / 10
            if recall >= threshold:
```

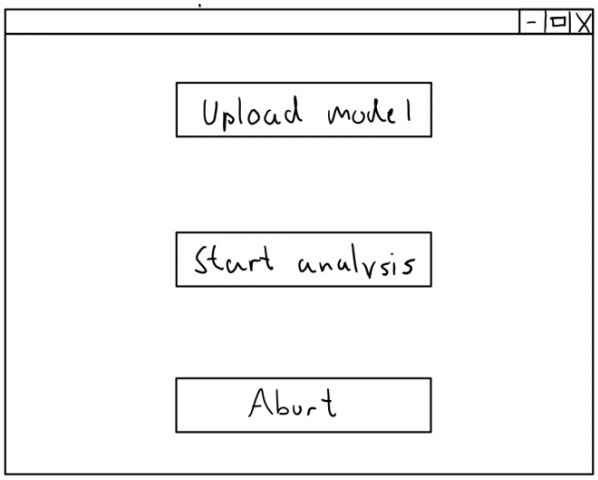
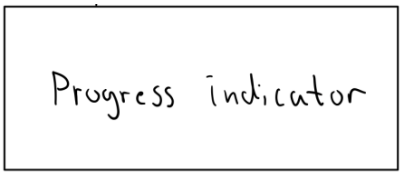
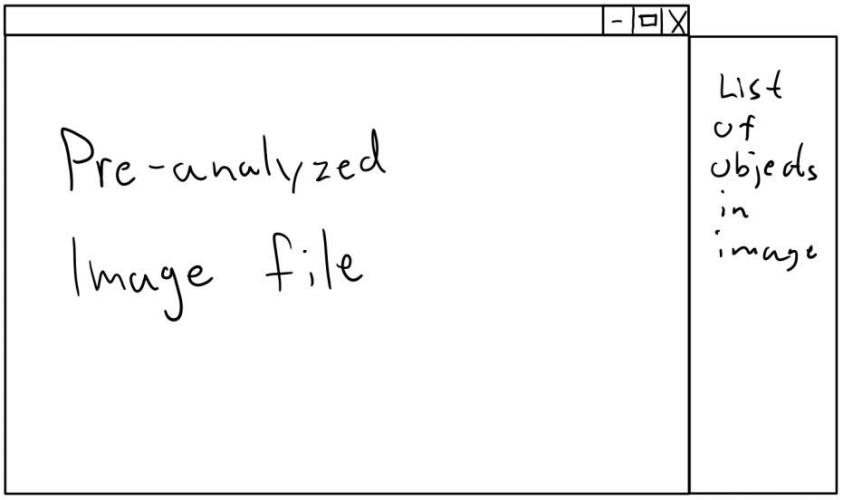
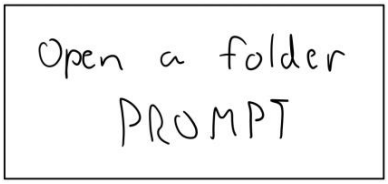
```
            max_precision = 0
            for k in range(len(yolo_data)):
                if yolo_data[k][0] != class_id:
                    continue

                yolo_box = [yolo_data[k][1], yolo_data[k][2], yolo_data[k][3], yolo_data[k][4]]
                max_iou = 0
                for l in range(len(annotated_data)):
                    if annotated_data[l][0] != class_id:
                        continue
                    annotated_box = [annotated_data[l][1], annotated_data[l][2], annotated_data[l][3], annotated_data[l][4]]
                    iou = compute_iou(yolo_box, annotated_box)
                    if iou > max_iou:
                        max_iou = iou
                if max_iou >= iou_threshold:
                    tp = 1
                    fp = 0
                    precision = tp / (tp + fp)
                    if precision > max_precision:
                        max_precision = precision

            ap += max_precision / 11

        aps.append(ap)

    mAP = np.mean(aps)

    return mAP


annotated_folder = 'annotated'
preanalyzed_folder = 'preanalyzed'
iou_threshold = 0.5

avgMAP = 0
numFiles = 0
```

```
for annotated_file in os.listdir(annotated_folder):
    if not annotated_file.endswith('.txt'):
        continue
    preanalyzed_file = os.path.join(preanalyzed_folder, annotated_file)

    if not os.path.exists(preanalyzed_file):
        print(f'Error: preanalyzed file {preanalyzed_file} not found')
        continue

    mAP = compute_mAP(preanalyzed_file, os.path.join(annotated_folder, annotated_file), iou_threshold)
    avgMAP += mAP
    numFiles += 1
    print(f'mAP for file {annotated_file}: {mAP}')

if numFiles > 0:
    avgMAP /= numFiles
    print(f'Average mAP: {avgMAP}')
else:
    print('No files processed')
```

Appendix M
Semi-Automated Annotation Tool Mockup Design

Table 1: Step by step design mockup of annotation software.

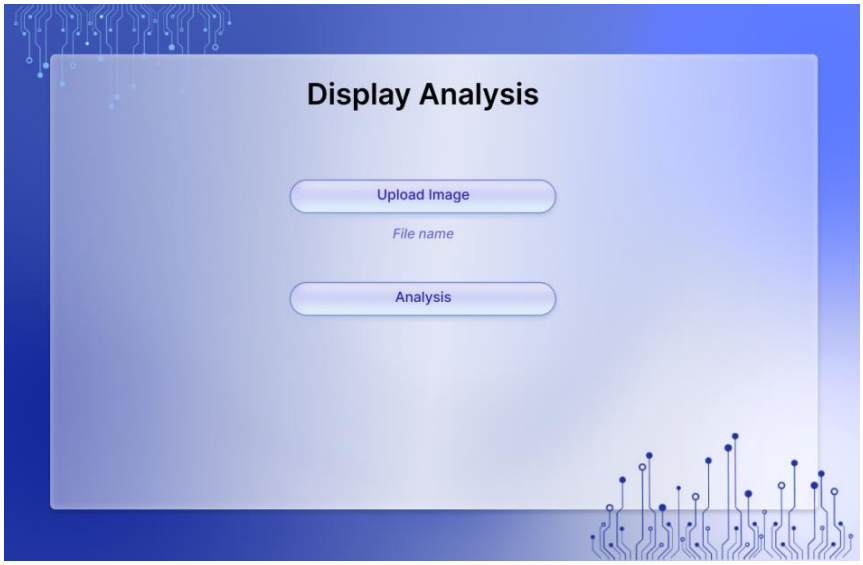
Start menu window	
Upload image prompt	
Crop image window	

Pre-Analysis window	
Progress indicator prompt	
Annotation window	
Folder save export prompt	

Appendix N
UI Figma Design ICE Software

Table 1: Different UI designs for the ICE software

Blue mobile	
Light mobile	

Blue desktop	
Light desktop