Centre for Maintenance Optimization & Reliability Engineering

Director
Chi-Guhn Lee

Semi-annual report
December 2020

Mechanical & Industrial Engineering
UNIVERSITY OF TORONTO

cmore.mie.utoronto.ca | 416·946·7867

Table of contents

C-MORE progress meeting agenda

Virtual meeting by Microsoft Teams
Thursday December 10, 2020

Opening remarks, executive summary 1:00-1:20		Chi-Guhn Lee
Feature presentation from Technical University of Denmark: Increasing maintenance effectiveness and efficiency 1:20-1:50		Waqas Khalid, Kristoffer Sigsgaard, Niels Henrik Mortensen
10-minute break 1:50-2:00		
Collaborations with consortium members		
2:00-2:20	Kinross: Determining the impact of maintenance on equipment availability	Janet Lam
2:20-2:40	Kinross: A business case for machine learning methods, a progress report	Kide Zhou
15-minute break 2:40-2:55		
Collaborations with consortium members		
2:55-3:15	UKMOD: Spares selection for aircraft embarked on carriers	Dragan Banjevic
3:15-3:55	DND: Assessing predictive power of oil readings for propulsion diesel engines	Janet Lam
15-minute break 3:55-4:10		
Collaborations with consortium members & guests		
4:10-4:30	TTC: Track re-inspection frequency analysis, a 2018-2020 update	Janet Lam
4:30-4:50	CEA: An audit of data encoding for multi-site maintenance records	Kimia Taghvaei Ganjali
Closing remarks 4:50-5:00		Chi-Guhn Lee

Executive summary
Chi-Guhn Lee, C-MORE Director

Introduction

After a fast learning curve, I am happy to say that C-MORE researchers, staff, students, collaborators, and company representatives have adapted very well to the exigencies of the Covid-19 pandemic. Teaching, conference presentations, and discussions with our collaborating companies are all taking place online – with no loss in the topnotch quality expected at C-MORE. The following report summarizes work undertaken since the meeting in June 2020.

The C-MORE team

Janet Lam, Assistant Director
Through the balance of 2020, Janet has pressed on with member projects either directly or through supervision of students. In particular, she worked with DND in the transition of staff, with all new collaborators from DND working on the propulsion diesel engine project. She also served as the lead writer in several grant proposals, from within the university to national calls for proposal. She (virtually) met with several potential collaborators to discuss ways to combine forces, ranging from research groups from other universities (Technical University of Denmark), maintenance solution providers (Celonis, Fiix Software) to direct consumers of C-MORE services (Primaris, Sonafi Foods). At the 2020 MainTrain virtual conference, she presented on “Machine learning approaches to take your asset management strategies to the next level.” In other work, she taught a graduate course in Statistical Methods of Quality Assurance, expanding the library topics of C-MORE’s expertise.

Andrew K. S. Jardine, Professor Emeritus
Andrew has remained busy over the past six months. Like everyone else, his teaching and speaking engagements have been virtual. In August, he delivered a virtual keynote address, “An analytic toolbox for optimizing condition-based maintenance (CBM) Decisions,” at APARM 2020, 9th Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modelling, Vancouver. In August, he also served on the PhD committee of Gaowei Xu, supervised by Fae Azhari. In September, Andrew was awarded a Life Membership by PEMAC Asset Management Association of Canada, in recognition of his significant contributions to PEMAC. In the fall term, he taught graduate class MIE 1723, Engineering Asset Management, with Sharareh Taghipour (online). In November, Andrew, Don Barry, and Sharareh Taghipour delivered the annual Physical Asset Management program under the auspices of the School of Continuing Studies (online). Another November project was a virtual seminar presentation, “Evidence based physical

asset management: Preventive replacement and inspection," at National University of Comahue, Argentina. In the week after the Progress Meeting, Gaowei Xu, Andrew, and Fae Azhari will make a virtual presentation, "Optimizing bridge maintenance management based on routine inspection data: Role of hazard modelling," at OMAINTEC 2020: International Operations & Maintenance Conference in the Arab Countries. Saudi Arabia. Andrew's final presentation scheduled for 2020 will be a keynote (virtual) address, "An analytic toolbox for predictive maintenance decisions," at Seminar on Asset Management, Universidad de Chile, Santiago, Chile.

Dragan Banjevic, C-MORE Consultant
Dragan continued to collaborate with C-MORE on projects with consortium members, mostly with Kinross Gold, TTC, and MOD. He also provided help in other projects with C-MORE students, as well as in their research.

Sharareh Taghipour, Ryerson, External Collaborator
In the fall term, Sharareh co-taught a graduate course with Andrew Jardine at the University of Toronto and was an instructor for the annual PAM course offered through the School of Continuing Studies in November. She is working on a project entitled "Decentralized data analytics and optimization methods for Physical Asset Management" (NSERC Discovery grant), as well as two collaborative projects with industry: "Developing methods for measuring social, economic, and environmental impacts of maintenance activities for physical assets," with Fiix Inc. (NSERC CRD), and "Real-time optimization of production scheduling," with Axiom Group (NSERC Alliance). In addition, she is using the "Industry 4.0 Smart Factory System" to develop predictive maintenance models and real-time optimization of production scheduling (funding from Ministry of Economic Development, Job Creation and Trade and John R. Evans Leaders Fund).

Scott Sanner, University of Toronto
Scott's group continues work on a range of applied projects covering power grid security (journal article accepted by IEEE *Transactions on Smart Grid*), predictive modelling for residential HVAC (journal article published in *Science and Technology for the Built Environment*), and predictive clustering for geographical applications (journal article published in *Geographical Analysis*). Scott's group also continues fundamental AI research on control methods leveraging deep learning (with two journal articles published in *Artificial Intelligence* as well as *Journal of AI Research*). Finally, in the past year, Scott's group has made significant advances in the next generation of personalized conversational assistants with three conference papers accepted to the leading research venues for this area (WWW-20, SIGIR-20, and RecSys 2020).

Fae Azhari, University of Toronto
Fae's research group now consists of 4 doctoral students, 3 MASc students, and 1 undergraduate student. Her projects include: complex naval asset management using sensor data, optimizing the fabrication and performance of multifunctional cementitious composites, fibre optic sensors for vibration monitoring, sensing system for gait analysis, bridge scour monitoring, condition-based maintenance of bridges, and compression creep behaviour of lead-free solder alloys. Her group members submitted a number of conference abstracts this past year, which will be now be presented virtually.

Jue Wang, Affiliate Professor
Jue Wang is an Assistant Professor at Smith School of Business, Queen's University. He is currently supervising/co-supervising two PhD students on sequential decision making, in collaboration with Scotiabank. He continues to work on methodological research related to sensor-based online fault diagnosis. Two recent papers have been accepted by top journals: one

by *Operations Research* (November 25) and the other by *Production & Operations Management* (October 16).

Ali Zuashkiani, Director of Educational Programs
Ali has been active providing consulting services to various industries such as oil and gas, power generation and distribution, mining, and petrochemical. He has been especially busy working with a group of 10 petrochemical plants and one major steel milling company to improve their Operation and Maintenance business processes and procedures. Ali has also been working with LEORON Training Company to develop C-MORE's educational programs. There are two upcoming programs, a 5-day Asset Management course and a 3-day Spare Parts management course which will be held in December 2020 and February 2021, respectively. Both will be virtual. Ali will be helping with another new C-MORE program in May 2021, Machine Learning and AI Applications in Physical Asset Management, recently developed by Janet and Chi-Guhn and offered through the School of Continuing Studies (online).

C-MORE students and postdoctoral fellows
We have students at all levels working with us – from undergraduate to doctoral. C-MORE students have been extremely busy over the past six months, with many working on projects specifically related to Consortium members' concerns. For more information on student research activities, see the section "Overall Project Direction" (page 16). We will also have students presenting their work at the December meeting.

C-MORE activities with consortium members

Defence Science and Technology Laboratory (DSTL)
Tim Jefferis proposed a new question on a complex spares management problem when there is a combination of replenishable and non-replenishable spare parts over a fixed mission duration. Arguably, parts from one non-operating unit could be used as spare parts for another non-operating unit to create one operating unit. This project will be presented at the December 2020 meeting by Dragan Banjevic.

Department of National Defence (DND)
The DND team experienced a full turnover on the staff associated with the propulsion diesel engine health analysis project. C-MORE and DND handled the transition and continued work on the project. We found the data did not demonstrate strong age-based relationships, but we are working on discovering relationships among the oil analysis variables. The project will be presented at the December meeting.

Kinross Gold Corporation
The KPI project wrapped up this fall, including an analysis of how the previous month's maintenance activities affect the present operational availability. We have started a project on the business case for machine learning in maintenance. Our student Kide Zhou is working through the data processing part of the project and will present his progress at the December meeting.

Toronto Transit Commission (TTC)
We received updated data on the TTC re-inspection project, doubling the size of the data set. The data have been incorporated into the existing project. The results of this work are reported in this document.

C-MORE educational programs

Andrew Jardine and Ali Zuashkiani continue to act as educational ambassadors for C-MORE. In November, Andrew, Don Barry, and Sharareh Taghipour delivered the annual Physical Asset

Management program under the auspices of the School of Continuing Studies (online). For his part, Ali has been working with LEORON Training Company to develop C-MORE's educational programs. Two upcoming programs, a 5-day Asset Management course and a 3-day Spare Parts management course, are scheduled for December 2020 and February 2021, respectively. Both will be held virtually. Last but by no means least, Janet and Chi-Guhn have developed a new, complementary five-day course, Machine Learning and AI Applications in Physical Asset Management, or "PAM2," scheduled for May 2021 and offered through the School of Continuing Studies (online). We are very excited about this new course and see it as an important addition to the C-MORE repertoire. The brochure is available at: https://learn.utoronto.ca/programs-courses/courses/3705-machine-learning-ai-applications-physical-asset-management.

Conclusion

Looking back at what we have accomplished in the past six months, I am proud of C-MORE's adaptability. In effect, we have managed to change direction mid-stride. Our output has exceeded what anyone might have anticipated in the first weeks of the 2020 lockdowns – with no loss of quality.

I wish everyone a safe holiday season and look forward to an equally productive but much less restrictive New Year.

Chi-Guhn Lee
December 2020

Visits and interactions with consortium members and others

June 2020 – December 2020

Note that all meetings during this period were virtual, in accordance with safe COVID-19 protocols.

Biweekly throughout **LG Sciencepark**
Chi-Guhn Lee, Hyun-Rok Lee, Songci Xu, Bin Yang, Ram Sreenivasan, Jongsung Jan, and Y.J. Jeong met to discuss on two research directions: 1) Transfer Learning for Reinforcement Learning; 2) Transfer Learning on Intelligent Fault Diagnosis. These bi-weekly meetings aim at sharing our research progress with LG and discussing technical details.

May 19, 2020 **DND**
Janet had a call with Kulan to discuss status of the PDE CBM project.

May 28, 2020 **Kinross**
Janet, Dragan and Frank met with Emilio, Brian, Theresa, Ethan, Gustavo, and Max to present the results of the KPI project.

June 3, 2020 **CEA**
Janet and Chi-Guhn met with Dan Gent to discuss starting Data Audit project. The results of this project will be presented in the December 2020 progress meeting

June 5, 2020 **Exxon Mobil**
Chi-Guhn and Janet met with Bart Maciszewski and Frank Arthur from Exxon Mobil to discuss a project in optimizing the shutdown procedures of their plants.

June 9, 2020
Our summer progress meeting took place virtually for the first time. The meeting was attended by 39 participants across 15 companies.

June 16, 2020 **Toromont & Safety Power**
Chi-Guhn and Janet met with Denis Gosselin and Carl Marinelli of Toromont and Bob Stelzer of Safety Power to discuss potential projects for Toromont's investment requirement for the Industrial and Technological Benefits policy.

June 16, 2020 **Primaris**
Chi-Guhn and Janet met with Gord Howells from Primaris REIT to discuss developing a relationship by conducting some small-scale projects involving students.

June 17, 2020 **Sonafi Foods**
Janet met with Brian Rossi of Sonafi foods to discuss potential collaborations.

June 23, 2020 **Primaris**
Janet met with Gord Howells to discuss a project for Tableau implementation of key indicators of their commercial properties.

July 8, 2020 **Kinross**
Seyedvahid met with Theresa and Ethan to define four sensors that collect health data for their vehicles for the ML business case project.

July 20, 2020 **Celonis**
Chi-Guhn and Janet met with Josua Vieten and Janina Nakladal of Celonis to introduce each other's primary business areas, and to discuss potential collaborations.

July 20, 2020 **TTC**
Janet and Dragan met with a large team at TTC, including Tauqeer Quarashi, Tim Southworth, Gordon Webster, and Mo Ghaus to discuss progress on the NDT re-inspection project.

August 13, 2020 **CEA**
Janet met with Dan Gent to discuss progress on data audit project. We have spent some time exploring best ways to manage such a large data set. It has been split up into separate files per plant, and data cleansing is underway.

August 18, 2020 **Primaris**
Janet and Gord had a meeting on Primaris' project on Tableau implementation, along with necessary changes due to the significant changes to Primaris' properties throughout the COVID-19 response.

August 22, 2020 **APARM**
Andrew gave a keynote presentation "An Analytic Toolbox for Optimizing Condition Based Maintenance (CBM) Decisions" and the 9th Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modelling, Vancouver, Canada, August 20-23, 2020

August 27, 2020
Andrew served as a committee member on the PhD defence of Gaowei Xu, supervised by Professor Faezeh Azhari

September 10, 2020 **CEA**
Janet and Kimia met with Dan, giving an update on the data cleansing process of the data audit project, and providing a plan for modelling. Kimia started as an Industrial Engineering thesis student for this academic year.

September 15, 2020 **PEMAC**
We have three events to report for this day.

Andrew was awarded a Life Membership by PEMAC Asset Management Association of Canada in recognition of his significant contributions to PEMAC

Chi-Guhn served as a panellist on Technology Panel Discussion on Emerging Technology for Asset Management at PEMAC's MainTrain conference, held virtually.

Janet presented a virtual talk, "Machine Learning Approaches to Take Your Asset Management Strategies to the Next Level," at PEMAC's MainTrain conference.

September 16 – December 9, 2020
Andrew taught graduate class MIE 1723: Engineering Asset Management with Sharareh Taghipour (online)

September 18, 2020 NRC
Chi-Guhn Lee, Dr. Yoon Ko and Sophie Tian met to understand the fire safety project and its relationship with other projects at NRC (specifically the E-Manifest project).

September 24, 2020 Fiix Software
Chi-Guhn and Janet met with Katie Allan and Frank Emery of Fiix Software, discussing ways to collaborate, especially by sponsoring student projects.

September 25, 2020 Kinross
Janet and Dragan met with Emilio Sarno, Brian Wright, Ethan Dabney, Theresa Taylor, and Moataz Negmeldin to present the latest findings on the KPI project.

September 30, 2020 DND
Chi-Guhn and Janet met with Kulan Ambalavanar, Elaine Fitzpatrick, and Abaida Al-Azzawi from DND to meet the new team championing the C-MORE file as Kulan moved on to another department.

October 2, 2020 NRC
Chi-Guhn Lee, Dr. Yoon Ko and Sophie Tian met to share updates on the grant application of the fire safety project and discuss creating a proposal for collaboration with other teams at NRC.

October 8, 2020 CEA
Janet and Kimia met with Dan Gent and Joelle Lancaster to show progress on the data audit project. We discussed probability distributions of uptimes as recorded by various different generating plants. We decided to focus on distributions of downtimes instead of uptimes.

October 16, 2020 NRC
Chi-Guhn Lee, Dr. Yoon Ko, and Sophie Tian met to share updates on the grant application and discuss resource allocation for the project.

October 20, 2020 DND
Janet met with Abaida and Daniel Saulnier to discuss progress on the PDE CBM project. Some records had ambiguous coding, wherein the preventive or corrective status were unclear. The meeting attendees clarified the coding and planned future work.

October 22, 2020 Kinross
Seyedvahid met with Theresa and Ethan to discuss the failure data and maintenance record spreadsheet.

October 28, 2020 **Suncor**
Janet met with Simone Smith of Suncor to discuss issues with detecting leaks in the undercarriage of Suncor's mobile asset fleet.

October 28, 2020 **Suncor**
Professor Chi-Guhn Lee and Professor Yu Zou met with Wei Xing, Jiahui Zhang, Tianyi Lyu, and Katie Xu in the initial meeting for 3D printer control project.

October 30, 2020 **NRC**
Chi-Guhn Lee, Dr. Yoon Ko, Sophie Tian, and Ozkan Elmali met to share updates on the grant application, to request data from NRC, and discuss potential complications with the combustion signature analysis project.

November 2-6, 2020
Andrew was lead instructor of the annual course on Physical Asset Management, given in conjunction with the University of Toronto's School of Continuing Studies program, and taught online this year by Andrew, Don Barry and Sharareh Taghipour

November 12, 2020
Andrew gave a seminar, "Evidence Based Physical Asset Management: Preventive Replacement and Inspection", at National University of Comahue, Argentina.

November 12, 2020 **CEA**
Janet and Kimia met with Dan, Joelle, and Asuka Boehm to discuss progress on the data audit project. We showed the probability distributions of the different downtimes and made a plan to compare the distributions' similarities to assess whether the data are being recorded the same.

November 13, 2020 **NRC**
Chi-Guhn Lee, Dr. Yoon Ko, Sophie Tian, and Ozkan Elmali met to share updates on the grant application and to talk about what to expect from the combustion signature data (no access to data yet).

November 16, 2020
Janet met with Dharmen Dhalia to discuss his work in maintenance in the context of industry and applications. We discussed opportunities to educate graduate students in maintenance applications.

November 17, 2020 **DND**
Janet met with Abaida and Daniel to discuss the latest results on the PDE CBM project. We assessed that the results suggest time-based preventive replacements are necessary due to no aging factor. Since this does not align with our understanding of reality, we are exploring the deficiencies in the data causing this result.

November 18, 2020 **DTU**
Chi-Guhn and Janet met with Niels Henrik Mortensen, Waqas Khalid, and Kristoffer Vanrup Sigsgaard of Technical University of Denmark to discuss our respective research portfolios. We found DTU's systemic approach to maintenance and asset management was a promising complement to C-MORE's research.

November 19, 2020 **PEMAC**
Chi-Guhn and Janet attended the PEMAC GTA's chapter update on the strategic plan for 2021.

November 20, 2020 **TTC**

Chi-Guhn, Janet and Dragan met with Tauqeer, Hossein, and Mostafa Nouri to discuss wrapping up the NDT re-inspection project and involving other departments within TTC for a broader base of projects.

November 26, 2020 **Primaris**

Janet met with Gord to discuss tactical steps in recruiting for the Tableau implementation project.

November 26, 2020

Jiahui Zhang and Katie Xu met to discuss further details of the 3D printer control project, including available data/hardware and general plan moving forward.

December 8, 2020 **DTU**

Janet met with the DTU team to discuss preparations for the December 2020 progress meeting.

December 18, 2020

Andrew will deliver a keynote virtual presentation, "An Analytic Toolbox for Predictive Maintenance Decisions," at Seminar on Asset Management, Universidad de Chile, Santiago, Chile.

C-MORE leadership activities

Chi-Guhn Lee, Director

Chi-Guhn continued to lead C-MORE in research projects and industry partnerships throughout the second half of 2020, albeit virtually. He served as a panelist on the Technology Panel Discussion on Emerging Technology for Asset Management at the MainTrain online conference, sharing recent advancements in machine learning, and how they may impact asset management. He spearheaded several grant proposals on leveraging predictive analytics to support the response to COVID-19 in Canada and is working steadily on developing relationships with various organizations to increase C-MORE's presence in the area of asset management in the face of emerging technologies.

Janet Lam, Assistant Director

Through the balance of 2020, Janet has pressed on with member projects either directly or through supervision of students. In particular, she worked with DND in the transition of staff, with all new persons from DND working on the propulsion diesel engine project. She also served as the lead writer in several grant proposals, from within the university to national calls for proposal. She (virtually) met with several potential collaborators to discuss ways to combine forces, ranging from research groups from other universities (Technical University of Denmark), maintenance solution providers (Celonis, Fiix Software) to direct consumers of C-MORE services (Primaris, Sonafi Foods). At the 2020 MainTrain virtual conference, she presented on “Machine learning approaches to take your asset management strategies to the next level.” In other work, she taught a graduate course in Statistical Methods of Quality Assurance, expanding the library topics of C-MORE’s expertise.

Andrew K. S. Jardine, Professor Emeritus

Andrew has remained busy over the past six months. Like everyone else, his teaching and speaking engagements have been virtual. In August, he delivered a virtual keynote address, “An analytic toolbox for optimizing condition-based maintenance (CBM) Decisions”, at APARM 2020 -The 9th Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modelling, Vancouver. Also in August, he served on the PhD committee of Gaowei Xu, supervised by Professor Faezeh Azhari. In September, Andrew was awarded a Life Membership by PEMAC Asset Management Association of Canada, in recognition of significant contributions to PEMAC. In the fall term, he taught graduate class MIE 1723, Engineering Asset Management, with Sharareh Taghipour (online). In November, Andrew, Don Barry, and Sharareh Taghipour delivered the annual Physical Asset Management program under the auspices of the School of Continuing Studies (online). Another November project was a virtual seminar presentation, “Evidence based physical asset management: Preventive replacement and inspection,” at National University of Comahue, Argentina. In the week after the Progress Meeting, Gaowei Xu, Andrew, and Fae Azhari will make a virtual presentation, “Optimizing bridge maintenance management based on routine inspection data: Role of hazard modelling,” at OMAINTEC 2020: International Operations & Maintenance Conference in the Arab Countries. Saudi Arabia. Andrew’s final presentation scheduled for 2020 will be a keynote (virtual) address, “An analytic toolbox for predictive maintenance decisions,” at Seminar on Asset Management, Universidad de Chile, Santiago, Chile.

Dragan Banjevic, C-MORE Consultant

Dragan continued to collaborate with C-MORE on projects with consortium members, mostly with Kinross Gold, TTC, and MOD. He also provided help in other projects with C-MORE students, as well as in their research.

Sharareh Taghipour, Ryerson University, External Collaborator

In the fall term, Sharareh co-taught a graduate course with Andrew Jardine at the University of Toronto. She was also an instructor for the annual PAM course offered through the School of Continuing Studies. She is working on a project entitled “Decentralized data analytics and optimization methods for Physical Asset Management” (NSERC Discovery grant), as well as two collaborative projects with industry: “Developing methods for measuring social, economic, and environmental impacts of maintenance activities for physical assets,” with Fiix Inc. (NSERC CRD), and “Real-time optimization of production scheduling,” with Axiom Group (NSERC Alliance). In addition, she is using the “Industry 4.0 Smart Factory System” to develop predictive maintenance models and real-time optimization of production scheduling (funding from Ministry of Economic Development, Job Creation and Trade and John R. Evans Leaders Fund).

Scott Sanner, University of Toronto

Scott's group continues work on a range of applied projects covering power grid security (journal article accepted by IEEE *Transactions on Smart Grid*), predictive modelling for residential HVAC (journal article published in *Science and Technology for the Built Environment*), and predictive clustering for geographical applications (journal article published in *Geographical Analysis*). Scott's group also continues fundamental AI research on control methods leveraging deep learning (with two journal articles published in *Artificial Intelligence* as well as *Journal of AI Research*). Finally, in the past year, Scott's group has made significant advances in the next generation of personalized conversational assistants with three conference papers accepted to the leading research venues for this area (WWW-20, SIGIR-20, and RecSys 2020).

Fae Azhari, University of Toronto

Fae’s research group now consists of 4 doctoral students, 3 MASc students, and 1 undergraduate student. Her projects include: complex naval asset management using sensor data, optimizing the fabrication and performance of multifunctional cementitious composites, fibre optic sensors for vibration monitoring, sensing system for gait analysis, bridge scour monitoring, condition-based maintenance of bridges, and compression creep behaviour of lead-free solder alloys. Her group members submitted a number of conference abstracts this past year, which will be now be presented virtually.

Jue Wang, Affiliate Professor

Jue Wang is an Assistant Professor at Smith School of Business, Queen's University. He is currently supervising/co-supervising two PhD students on sequential decision making, in collaboration with Scotiabank. He continues to work on methodological research related to sensor-based online fault diagnosis. Two recent papers have been accepted by top journals: one by *Operations Research* (November 25) and the other by *Production & Operations Management* (October 16).

Ali Zuashkiani, Director of Educational Programs

Ali has been active providing consulting services to various industries, such as oil and gas, power generation and distribution, mining, and petrochemical. He has been especially busy working with a group of 10 petrochemical plants and one major steel milling company to improve their Operation and Maintenance business processes and procedures. Ali has also been working with LEORON Training Company to develop C-MORE's educational programs. There are two upcoming programs, a 5-day Asset Management course and a 3-day Spare Parts Management course which will be held in December 2020 and February 2021, respectively. Both will be held virtually. Ali will also be helping with a new course, Machine Learning & AI Applications in Physical Asset Management, recently developed by Janet and Chi-Guhn, offered in May 2021 through the School of Continuing Studies (online).

Overall project direction
Janet Lam, Assistant Director

Goals and retrospectives

This section highlights the some of the main achievements in C-MORE for the period June 2020 – December 2020. Throughout the months of the pandemic, the C-MORE team continued to operate with work-from-home modifications, with all meetings happening virtually.

Like many other organizations, we attended our first virtual conferences and hosted our own progress meeting online. Our first virtual Physical Asset Management course was delivered in November, and our PAM2 course is scheduled for May of 2021; it is being planned for virtual delivery.

As part of the COVID-19 response, there were several government grants and programs targeted toward addressing the pandemic. We submitted a proposal to the Department of National Defence's call for a proposal on "Rapid Response: Real-time insights for pandemic decision-making" and to the University's internal Connaught Innovation Award for commercialization.

Activities

Collaboration with companies and site visits
This section gives details on progress in research conducted with consortium members.

Member	Collaborations
Defence Science and Technology Laboratory	Tim proposed a new question on a complex spares management problem when there is a combination of replenishable and non-replenishable spare parts over a fixed mission duration. Parts from one non-operating unit could be used as spare parts for another non-operating unit to create one operating unit. This project will be presented at the December 2020 meeting by Dragan Banjevic.
Department of National Defence	The DND team experienced a full turnover on the staff associated with our project. C-MORE and DND worked on a full transition process and continued work on the propulsion diesel engine health analysis project. We found the data did not demonstrate strong age-based relationships, but we are working on discovering relationships

Member	Collaborations
	among the oil analysis variables. This project will be presented at the December meeting.
Kinross	The KPI project wrapped up this fall, including analysis of how the previous month's maintenance activities affect the present operational availability. We started a project on the business case for machine learning in maintenance. Our student Kide Zhou is working through the data processing part of the project and will present his progress thus far at the December meeting.
Toronto Transit Commission	We received updated data on the TTC re-inspection project, doubling the size of the data set. The data have been incorporated into the existing project. The results of this work are reported in this document.

Theoretical work

This section on theoretical work is oriented toward students' and postdoctoral fellows' research topics.

Name	Activity
Tushar Aggarwal, MEng student	Tushar is in his first year of MEng in the Mechanical Engineering program and is focusing on artificial intelligence and robotics. He is currently working on anomaly detection on train tracks using thermal imaging. The project was assigned to him in November 2020. He is exploring using Convolution Neural Network (CNN) algorithms such as YoloV4 for object detection and tuning them to custom thermal image dataset received from TTC to detect anomalies in railway tracks.
Kuilin Chen, PhD candidate	Kuilin passed his first committee meeting and achieved candidacy status in October 2020. He completed the project on incremental few-shot learning and submitted a paper to ICLR 2021. His research interest is few-shot learning.
Ozkan Elmali, MASc student	Ozkan is a new MASc student working on a reinforcement learning approach to the vehicle routing and safe routing problem. As vehicle routing is known to be a computationally hard problem, there is great potential in taking a RL approach. Safe routing is a special case of vehicle routing for transporting hazardous materials that have additional restrictions and a heavier weight to reduce crash risks. As a newly registered student, he is currently taking several courses.
Anmol Garg, MEng student	Anmol is in the final year of MEng in Mechanical & Industrial Engineering. In the pursuit of a career in data analytics, he has taken introductory courses. Now, in a real-world setting, he has been involved in a project under Professor Chi-Guhn Lee since March 2020. The project deals with detecting banned items in baggage using Convolutional Neural Network (CNN) concepts applied via the YOLOv3 algorithm. The model is meant to be deployed in scanning

Name	Activity
	systems at transportation hubs (mainly airports). The project also delves into Domain Adaptation integrated with Object Detection problem, a novel concept by itself as Domain Adaptation is usually concerned with Classification.
Michael Gimelfarb, PhD candidate	Michael has continued his doctoral work on knowledge transfer in reinforcement learning using graph-structured data, Bayesian approaches and hierarchical RL. He is supervised by Scott Sanner and Chi-Guhn Lee.
Scott Koshman, PhD candidate	Scott has continued his doctoral research on equipment health monitoring for Halifax Class frigates. He is supervised by Professor Fae Azhari.
Seyedvahid Najafi, PhD candidate	Vahid is a full-time PhD student who is working on the maintenance modelling and optimization of multi-unit series systems. In his research, an opportunistic maintenance policy with general repair is developed for a two-unit series system, in which the condition of one unit is monitored, and only the age information is available for the other unit. The problem is formulated in a semi-Markov decision process framework, and an algorithm is developed to find the optimal control limits and the long-run average cost per unit time. A paper entitled "An optimal maintenance policy for a two-unit series system with general repair" was accepted by IISE Annual Conference & Expo 2020, New Orleans, Louisiana, and ICOR 2020 conference, Toronto. He applies reinforcement learning algorithms to find optimal solutions for maintenance problems modeled in the SMDP framework. Vahid will be ready for his first annual progress review meeting at the beginning of 2021.
Shashank Saurav, MEng student	Shashank is a final year MEng Student in Electrical and Computer Engineering. Since April 2020, he has been working under the supervision of Dr. Chi-Guhn Lee on a project which employs the state-of-the-art YOLO algorithm to detect banned items in security scanned images at airports. The project also explores a Domain Adaptation approach in the object detection paradigm.
Jahyun Shin, MEng student	Jahyun is a first year MEng student in Mechanical and Industrial Engineering program with an emphasis on analytics and machine learning. Under Professor Chi-Guhn Lee, she is taking charge of an early-stage deep learning research project to develop a model that can automatically detect and classify restricted items for an airport's X-ray baggage scanner. She is also applying adversarial domain adaptation techniques to compensate for the small number of labelled X-ray images provided by the stakeholder
Avi Sokol, PhD candidate	A flex-time PhD student and a full-time Business Data Scientist and Inventory Specialist, Avi is researching the integration of Reinforcement Learning and Inventory Control to reduce waste in supply chains.

Name	Activity
Kimia Taghvaei Ganjali, IndE thesis student	Kimia is a fourth-year Industrial Engineering student and is working on the CEA data audit project for the individual thesis. The objective of the project is to build a framework for CEA. This framework will help assess its records and build consistency throughout all sites working with CEA. Large data sets of all CEA sites have been audited and analyzed to generate a model and perform statistical analysis. The project is now in the evaluation phase.
Sophie Tian, MASc student	Sophie is a first year MASc. Since September 2020, she has been taking Introduction to Machine Learning and Stochastic Processes. She is working on the combustion signature analysis project in collaboration with NRC and is reviewing the literature on gas classification methods.
Katie Xu, MASc student	Katie started her MASc in September 2020. She has been working with collaborators in the Department of Materials Science and Engineering to kick-start a project on the use of machine learning techniques for *in-situ* process monitoring and control in 3D printing systems. She has been building background knowledge on this topic by reviewing the literature, while taking courses and studying machine learning fundamentals.
Songci Xu, PhD candidate	Songci started his PhD program in January 2020. His research focuses on applying transfer learning to Intelligent Fault Diagnosis (IFD), one of the projects of LG Sciencepark. He is also working on the explanatory ability of deep domain adaptation via Optimal Transport Theory, which counts towards his thesis.
Ruiqi Yang, MEng student	Ruiqi Yang is a second-year graduate student pursuing his Master's of Engineering in Mechanical and Industrial Engineering. He finished his MEng project, "Generic Real Options Valuation," at the end of August 2020. He and his teammates defined the Real Options Valuation model using option theory, stochastic process, and Markov decision process. They solved the model by using Double-Deep Q Learning and gave the numerical results. Since September, he has tried several different Deep Reinforcement Learning algorithms (all components of the Rainbow) with the Recurrent Neural Network (LSTM) architecture; the best agent he has found is with the D3RQN (Dueling Double Deep Recurrent Q Network) with PER (Prioritized Experience Replay).
Zihan Zhang, MASc student	Zihan is a second-year Master's student. She started to work on the battery prognostics project in April, 2020. She has studied more than 100 papers on the battery prognosis and finished a 30-page literature review report this summer. She is designing a neural-network-driven stochastic degradation process for battery forecasting. In addition to her independent project, she is guiding an undergraduate student from Mechanical Engineering and an MEng student working on the prognostic research project.

Name	Activity
Kide Zhou, IndE thesis student	Kide is a fourth-year Industrial Engineering student currently working on the Kinross replacement policy project. This project aims to build a replacement and maintenance policy helping the company to manage the assets with the lowest operating and maintenance cost. The sensor data have been analyzed in terms of the asset working condition, and the statistical results were generated. Based on the findings, a policy will be generated. The project is now in the modelling process.
Chengjiang Zou, MEng student	Chengjiang Zou is a second-year graduate student pursuing his Master’s degree in Mechanical and Industrial Engineering. He is currently working on the CEA sequential pattern mining project. This project was assigned to him in the beginning of September. The project uses machine learning algorithms and codes to detect patterns and rules of equipment operation by inspecting the continuous records from CEA datasets on Spark. Sequential pattern mining algorithms are used on the CEA datasets to unlock the hidden patterns of the continuous records. As a result, a set of preventive maintenance policies are derived based on the output of sequential pattern mining.

C-MORE publications and presentations in 2020

Journal papers published or accepted

[1] Li, G., Yang, L., Lee, C.-G., Wang, X. & Rong, M. (2020). A Bayesian deep learning RUL framework integrating epistemic and aleatoric uncertainties. IEEE Transactions on Industrial Informatics, online (DOI 10.1109/TIE.2020.3009593).
[2] Wang, J. (2020, forthcoming). Optimal sequential multi-class diagnosis. Operations Research.
[3] Wang, J. (2020, forthcoming). Optimal Bayesian demand learning over short horizons. Production & Operations Management.
[4] Wang, J., Levin, Y., & Nediak, M. (2020). Selling passes to strategic customers. Operations Research, 68(4), 1095–1115.
[5] Huchuk, B., O'Brien, W., & Sanner, S. (2020, in press). Exploring smart thermostat users' schedule override behaviors and the energy consequences. Science and Technology for the Built Environment.
[6] Olson, A. W., Zhang, K., Calderon-Figueroa, F., Yakubov, R., Sanner, S., Silver, D., & Arribas-Bel, D. (2020, in press). Classification and regression via integer optimization for neighborhood change. Geographical Analysis.
[7] Say, B., & Sanner, S. (2020). Compact and efficient encodings for planning in factored state and action spaces with learned binarized neural network transition models. Artificial Intelligence Journal, 285: 103291.
[8] Wu, G., Say, B., & Sanner, S. (2020). Scalable planning with deep neural network learned transition models. Journal of Artificial Intelligence Research, 68.

Journal papers submitted or under review

[1] Aboussalah, A., & Lee, C.-G. (2020). Symmetry-augmented representation for time series. ICLR, under review, October 2020.
[2] Arasteh, M., Alizadeh, S., & Lee, C.-G. (2020). Gravity algorithm for the community detection of large scale network. Journal of Ambient Intelligence and Humanized Computing, under review, October 2020.
[3] Babatunde, G., Lee, C.-G., & Sanner, S. (2020). Simultaneous estimation of discount factor and reward function in inverse reinforcement learning. AAAI, under review, September 2020.
[4] Chen, K., & Lee, C.-G. (2020). Incremental few-shot learning via vector quantization in deep embedded space. ICLR, under review, October 2020.

[5] Gimelfarb, M., Sanner, S., & Lee, C.-G. (2020). Bayesian experience reuse for learning from multiple Demonstrators. AAAI, under review, September, 2020.
[6] Momodu, A., & Lee, C.-G. (2020). Trinomial tree methods for pricing Israeli options with uncertain volatility. Applied Mathematical Finance, under review, October 2020.
[7] Xing, W., Chu, X., Lee, C.-G., Zou, Y., & Rong, Y. (2020). Using machine learning to classify images of melt pool surfaces in a selective laser melting process. Additive Manufacturing, revision under review, November 2020.
[8] Yang, B., Xu, S., Lei, Y., Lee, C.-G., Stewart, E., & Roberts, C. (2020). Multi-source Transfer learning network to complement knowledge for intelligent diagnosis of machines with unseen faults. Mechanical Systems and Signal Processing, under review, August 2020.
[9] Yang, B., Lei, Y., Lee, C.-G., Li, N., & Lu, N. (2020). Deep partial transfer learning network: A method to selectively transfer diagnostic knowledge across related machines. Mechanical Systems and Signal Processing, revision under review, August 2020.

Conference presentations

[1] Jardine, A.K.S. An analytic toolbox for optimizing Condition Based Maintenance (CBM) Decisions. Keynote presentation, APARM 2020, 9th Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modelling, Vancouver, Canada, August 20-23, 2020 (virtual conference).
[2] Jardine, A.K.S. Evidence-based physical asset management: preventive replacement and inspection. Seminar at National University of Comahue, Argentina, November 12, 2020 (virtual presentation).
[3] Jardine, A.K.S. An analytic toolbox for predictive maintenance decisions. Keynote presentation at Seminar on Asset Management, Universidad de Chile, Santiago, Chile, December 18, 2020 (virtual presentation).
[4] Lam, J., & Lee, C.-G. Machine learning approaches to take your asset management strategies to the next Level. Maintenance, Reliability and Asset Management (MainTrain) Online Conference, Canada, September 15-16, 2020
[5] Lee, C.-G. Technology panel discussion on emerging technology for asset management. Maintenance, Reliability and Asset Management (MainTrain) Online Conference, Canada, September 15-16, 2020
[6] Xu, G., Jardine, A.K.S., & Azhari, F. Optimizing bridge maintenance management based on routine inspection data: Role of hazard modelling. OMAINTEC 2020: International Operations & Maintenance Conference in the Arab Countries. Saudi Arabia, December 15, 2020 (virtual presentation).
[7] Li, H., Sanner, S., Luo, K., & Wu, G. A ranking optimization approach to latent linear critiquing in conversational recommender System. 14th International ACM Conference on Recommender Systems (RecSys-20), online, 2020.
[8] Luo, K., Yang, H., Wu, G., & Sanner, S. Deep critiquing for VAE-based recommender systems. 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-20), Xi'an, China, 2020.
[9] Luo, K., Sanner, S., Wu, G., Li, H., & Yang, H. Latent linear critiquing for conversational recommender systems. 29th International Conference on the World Wide Web (WWW-20), Taipei, Taiwan, 2020.
[10] Najafi, S., & Makis, V. An optimal maintenance policy for a two-unit series system with general repair. IISE Annual Conference & Expo 2020, May 30-Jun 2, New Orleans, Louisiana, USA.
[11]Najafi, S., & Makis, V. Comparison of two maintenance policies for a two-unit series system considering general repair. ICOR 2020 (International Conference on Operations Research), Toronto, June 18-19, 2020.

CEA: An audit of data encoding for multi-site maintenance records

Kimia Taghvaei Ganjali

Introduction

Canadian Electricity Association (CEA) is a national leading electricity association in Canada representing many electricity companies. CEA was founded in 1891 and consists of senior executives from its Corporate Utility Members. CEA provides value added projects and services to improve the safety, security and sustainability of the Canadian electricity industry. These strategic projects are research outputs to improve the performance of utility members.

Every electricity company under CEA's wing uses the guidelines provided by CEA to record their equipment operating and outage data along with timestamps. Every system can have different failure modes, hence different types of outages for power plants. The process in which these outages are reported impacts the validity of the reliability measures. A framework is needed to ensure the quality of reported data is not as dependent on organizational and behavioral factors.

The objective of this research thesis is to generate a framework for CEA and their utility members. This framework will help assess their records and build a consistency throughout all sites working with CEA. The framework will be built by auditing the data provided by CEA. Large data sets of all CEA sites will be audited and analyzed to generate a model and perform statistical analysis. Results of the analysis would then be compared to see if there are any similarities between the sites or if there are significant differences.

Progress

The project was broken down into four phases: business understanding, data understanding and prep, modelling, and evaluation. Each phase will be explained in this section. It also includes a summary of results.

Business understanding

In the early stages of the project, the focus was more on understanding the terminologies used by CEA and the guidelines they had for the sites. CEA manuals and reports were studied to acquire a greater background for the project. The important business terms for this project are listed below:

- State code: the state reported for the unit as per the manual.
- Available states: 11, 12, ..16
  - Outage codes (unavailable states): 21, 22, …25
  - Only reporting changes in states (i.e. no consecutive state 11's)
  - The most important states for this study are 24 (maintenance outage) and 25 (planned outage)
- Duration: duration of outage in hours.
- Capacity: capacity of the unit. Have eight capacity groups and units should be classed based on their group

Data understanding

Once the objectives and CEA terms were understood, the project moved into a data analysis stage. The meaning behind certain attributes and conditions under which some entries were recorded was discussed with CEA to understand what is required for the analysis. A database was created to merge the large data files from different plants along with other reference tables needed for the analysis. The final dataset was then investigated to identify issues. Issues such as null values, duplicate entries, wrong data types were resolved. The result of the data prep stage was a clean dataset that was then split to create a separate file for every plant.

Modelling

There are two main models built for this project namely the data model and stats model. The data model takes the clean data file for every plant as an input and outputs a list of downtime durations for every capacity range. Initially, the focus of the model was on the operating codes but when discussed with the client, there was a need to further analyze the downtime codes (24, 25) and hence the adjustments were made.

The model looks at the units within each capacity group and creates a list of maintenance outage and planned outage durations for that specific capacity group. If certain plants do not have units within some capacity groups, the list is empty for those ranges. The list is then sorted to create a histogram of all the data points and pass it to the stats model. This data model is created to avoid the manual work for different plants.

The statistical model takes the list of downtime durations for every capacity group and outputs the name of best fitted distribution and its parameters. The model performs a statistical fit analysis by fitting the empirical distribution to the theoretical ones in the scipy stats package. The model compares the fitted distributions by calculating the parameters and error (SSE) and picks the one with the lowest discrepancy. Initially, all the distributions in the scipy package were studied which resulted in an extensive runtime hence the list was shortened to twenty known distributions.

This model is an efficient method to see the resulting probability distributions for a range of sites. The results will then be evaluated to determine if there are statistically significant differences.

Evaluation

To get some preliminary results, the model was tested with a small number of instances. Ten sites were picked randomly and studied with one state code. Some of the preliminary results discussed with CEA are shown below.

Table 1. Instance results

Plant number = 1405 State code = 24						
Capacity group	0-4	5-23	24-99	100-199	200-299	300+
Distribution name	-	exponential	exponential	-	-	-
Parameters	-	(0.0, 18.589)	(0.0, 12.297)	-	-	-

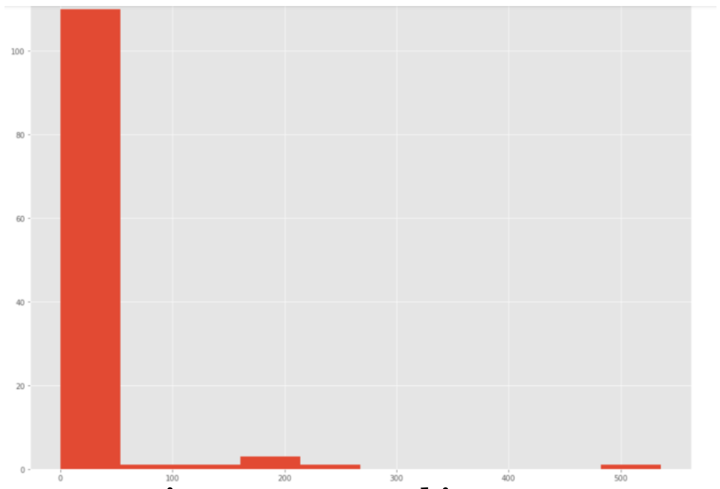


Figure 1. Instance histogram

Table 2. Instance results

Plant number = 6154 State code = 24						
Capacity group	0-4	5-23	24-99	100-199	200-299	300+
Distribution name	-	exponential	-	-	-	-
Parameters	-	(0.0, 7.828)	-	-	-	-

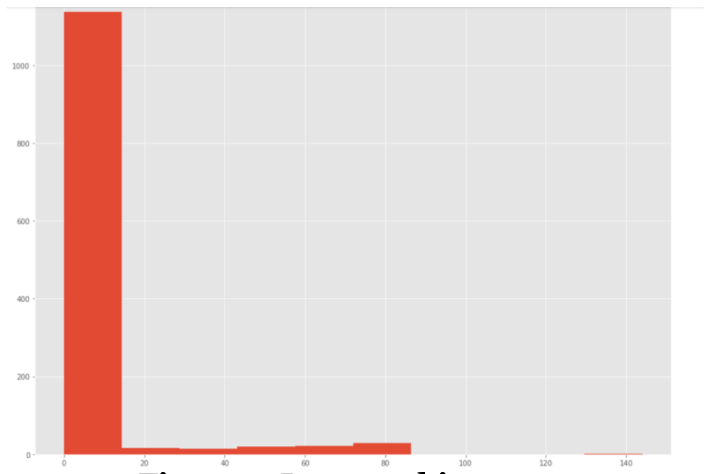


Figure 2. Instance histogram

Table 3. Instance results

Plant number = 2057 State code = 24						
Capacity group	0-4	5-23	24-99	100-199	200-299	300+
Distribution name	-	logistics	-	-	-	-
Parameters	-	(9.29, 7.61)	-	-	-	-

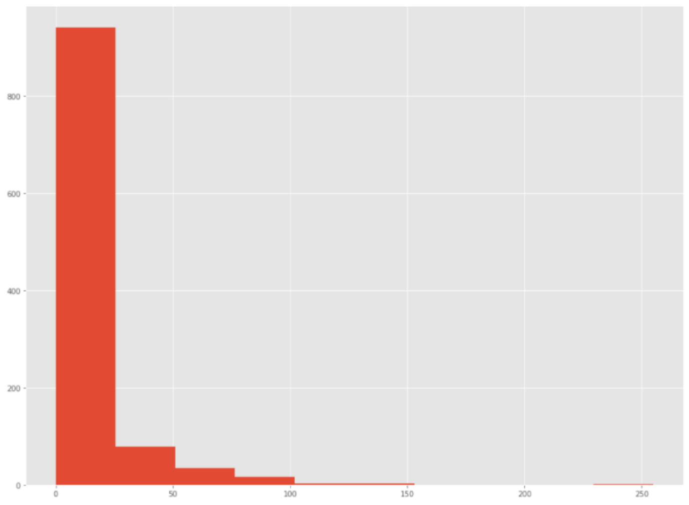


Figure 3. Instance histogram

After presenting these results to CEA, there was a note that the downtime for maintenance outage should not be more than a week and the case of 536 hours (plant 1405) was an error and should not be part of the list. Hence the downtime list should be evaluated for outliers before passing it to the stats model. The evaluation stage is a work in progress and the ideas are discussed in the next section.

Future work

The results obtained for each plant, state code, and capacity group combination should be organized in a chart for evaluation. There are two scenarios that could happen after studying several plant, state code, and capacity combinations:

1. The resulting distributions are very different.
2. The resulting distributions are similar (i.e. both exponential) but have different parameters.

For case 1 the list of theoretical distributions could be further decreased. For case 2 the samples should be evaluated using their sample size, mean, and standard deviation to see if they are truly different. The next step is to present results to CEA as a framework. After reporting the results for downtime analysis to CEA, there is value in investigating the operating codes and understanding the difference between different available states and if all sites are reposting these consistently. The same method could be used as well as developing other methods of analysis.

UKMOD: spares selection for aircraft embarked on carriers

Dragan Banjevic

Background

The problem was initiated by an e-mail from Tim Jefferis to Janet Lam on 2020-06-30:

> Hi Janet
>
> We have an interesting problem about spares optimisation. We are thinking about how we work out what spares to take onto the Carrier to support the embarked aircraft. We are going to start by looking at what the OEM suggests is required and see what system performance this gives us (in terms of flight hours generated, aircraft availability etc.) as we have a big simulation that will do this for us. When we find that this performance is less that we would like we then need to examine which parts / types of parts are causing this shortfall and what could best be done about this. There are naturally constraints that the value and weight of the spares pack.
>
> I am aware that there are simulation packages that will do this work, when one is optimising for something deterministic, such as stock out risk for a large fleet with constant usage, we have an issue that we have to run simulation to see what performance our spares back gives us and so I want to find some structured approach that doesn't result in us layering one simulation on top of another.
>
> Any thoughts about who knows about this would be appreciated.
>
> My feeling is that we probably need to divide the spares into two classes, those with relatively high usage, whose absence will constrain the total number of flight hours achieved and those with low usage whose presence is an insurance policy. One could then trade cost / weight invested between these two types as having more high usage spares will give an increased number of flight hours that might be achieved, but having more insurance spares will increase the probability of actually achieving a given number of flight hours, without hitting a mission critical failure that you haven't provided a spare for.
>
> Many thanks
> Tim Jefferis

Problem description

This e-mail was followed by several exchanges describing the problem and possible approaches to its solution. It is not clear how to summarize all aspects of the problem in a meaningful model that can be solved simply. The exact objective of the optimization is not even clear. Let us try to describe basic aspects of the problem, as we understand them, to help us to suggest an approach to the solution. A more formal mathematical model needed for actual calculation is described later in the document.

The carrier stays on a mission for specified time period, e.g., six months, without returning to the port. It embarks certain number of aircraft that will go on flying missions (sorties) on time to time.

Every aircraft is a complex system consisting of a large number of components. Some component can fail during the mission and has to be replaced to make the aircraft available. It is assumed that those components are not repairable, at least not on the carrier. By our assumption, the problem does not include repairable components. We assume that an aircraft cannot fly without any of those component, that is, it is a **series system** of those components. We are only concerned with aircraft components, not with the carrier components.

The number of replaceable component is large, say, more than 1000. Some of them are consumable component that are replaced on regular bases and their number can be predicted fairly accurately in advance. Their number and storage can be excluded from this consideration. The life of other components cannot be predicted in advance, as it is assumed they fail at random. So we need to store some of them on the carrier, for insurance. If some of the components are subject to planned replacement, that numbers should also be separated from unplanned replacement (similar as for all cases that can be predicted in advance).

The main question is what components to store on the board, and in what numbers. To do this, we need some way to classify them into classes. There are several approaches to classification, as mentioned in e-mails discussion. Some components are small, some are big, some fails more often, and some fail rarely. But, the key point is whether they can be replenished, if needed. Some components cannot be replenished during the carrier mission, and some of them can be flown in on a regular basis. Replenishment mostly depends on the component size. So, we will assume there are periodic replenishments during the mission, and classify all components into "replenishment levels", depending on the frequency of their replenishment. "Level 1" includes bulky components that cannot be replenished during the mission, before returning to the port, "level 2" that can be replenished only once, say in the middle of the mission, "level 3" two times over the mission, and so on. Selection of the levels will depend on actual replenishment strategy that should be discussed with the user. Why is this classification by replenishment important? Because we need to store an appropriate number of spares that can last between replenishments, according to the objectives.

Constraints on the storage of spares. The storage space for spares on the board is limited. The cost of keeping the spares on the board also can be limited. In our approach we will consider space limitation. Every component (one unit) occupies known amount of space. By assumption, the space occupied by one component type = space for one x # of the units. The total space occupied is sum of spaces occupied by all types. Due to limitations and demand some components may even not be stored. Our approach is to fix/reserve storage space for every replenishment level. Otherwise, the complexity of the problem would increase significantly. An interesting question is whether failed spares reoccupy their space in the storage, or that space can be reassigned. This question is not only of practical importance, but may affect the mathematical model, as we shall see later.

Demand for spare parts. One of the difficult problems is to reasonably describe demand for component. It, obviously, depends on the flying hours consumed by aircraft in every mission and the number of missions. This problem would be almost impossible to solve if we don't assume a constant demand per hour for every component. This would be a reasonable assumption if the number of aircraft is not very small (say, it is not less than 10), and if aircraft are not (all) new, that is, the components have some hours on them (at least, a reasonable part of them). Even if those assumptions are not met, the assumptions of constant demand rate would give a conservative solution that will not be far from optimal one. At least, we can use it as an initial assumption. In that case, the demand does not depend on the separate aircraft or the number of sorties, but on the total number of flying hours for each aircraft. We assume this number is known in advance (at least approximately). More technical details about this assumption will be considered later.

Optimization objective. This is another difficult problem that was not cleared up in the e-mail discussion. We need to quantify it somehow. The cost of the mission is part of a different calculation. It can be assumed that the overall cost is not a limiting factor for the spare parts problem, so that the mission success is the main objective. The mission success can be measured in several ways. Useful measures are aircraft availability and mission reliability, that is, the probability that there will not be a shortage of spare parts during the entire mission. We should consider this problem in more details. Let, say, there are 40 aircraft on the carrier, and 20 are required for every mission. It means we need at least 20 units of every component operational (or 20 aircraft with all components working) and spares to support them. The mission reliability will be the probability that we will have at least 20 aircraft available for flying through the mission time. This will be the same as availability of 20 aircraft over mission time. This approach includes possible cannibalization of spare parts, as they can be obtained from aircraft not required for flying. In a case of only non-replenishable spare parts ("level 1" spare parts), possible non-availability time (time with less than 20 aircraft ready to fly) would be at the end of the mission time. With more levels involved, non-availability time may occur prior to ends of replenishment intervals. The reliability of the stock is calculated as the product of reliabilities of all required components. The mathematical formula is described later.

The goal of the study will be to maximize the overall reliability of the stock with constraints on the storage space.

Mathematical model

1. The carrier embarks n aircraft that will go on flying missions on time to time. Every flying mission (sortie) requires k operational aircraft, $k \leq n$, at any time during total mission time $t = T$ (total time for all missions).
2. Every aircraft requires M replaceable nonrepairable components (component classes) to be able to fly. The components fail at random, with constant failure rates (per hour) $\lambda_1, \lambda_2, \ldots, \lambda_M$. We assume that all components are different and that we need one of each. If more than one component of the same class is needed (e.g., on the left and right wing), we can consider it as a single component with doubled failure rate. In that sense, we consider an aircraft as a series system with M different components. We also assume that failures of different components are independent.

3. We classify components by replenishment levels (see above), that is, by time horizon in which they cannot be replenished; e.g., bulky components cannot be replenished through the total mission time T, except at the beginning of the mission. For example, the stock can be replenished two times during the entire mission, making each replenishment interval of length $T/3$ (replenishment takes place at the beginning of an interval). The intervals need not be of the same length. Thus, we have three replenishment intervals $[0, t_1), [t_1, t_2), [t_2, T)$. For every component, we have replenishment times and appropriate replenishment intervals lengths. For example, for bulky components, there is no replenishment time except at time = 0, with one interval of length T. If a component can be replenished two times, at 0 and t_2, it has two replenishment intervals, with lengths t_2 and $T - t_2$. All components with same replenishment strategy belong to that replenishment level. For a component in level i, replenishment interval lengths are $t_{i1}, t_{i2}, \ldots$, with $\sum_j t_{ij} = T$. This is important because the reliability of the component will depend on its set $t_{i1}, t_{i2}, \ldots$. This approach describes a fairly flexible replenishment policy, depending on availability of aircraft. Note some components may not have any replenishment, e.g., highly reliable and/or very bulky. But they are in every aircraft and might be available for cannibalization. We assume replenishment strategies are fixed in advance and are part of other logistics beyond this project.
4. Assume L replenishment strategies/levels $A_1, A_2, \ldots, A_L$ and every component belongs to one of them. Let $t_{i1}, t_{i2}, \ldots$ are replenishment intervals lengths for level i. Denote by a_{ij} the component j belonging to level i, or $a_{ij} \in A_i$, and by λ_{ij} its appropriate failure rate. Let $m_{ijl}, m_{ijl} \geq 0$ be the number of stocked units of component a_{ij} in an interval of length t_{il}.
   (a) In general, if k units are required to work over an interval of length t, and we look at the single component with failure rate λ, the probability that the demand (number of failures) in that interval is less than equal to m (reliability of a stock of size m) is $\sum_{x=0}^{m} \frac{(k\lambda t)^x}{x!} e^{-k\lambda t}$. We now can apply this formula to our case.
   (b) If we consider cannibalization (components are interchangeable), we assume that the components from other aircraft are available as additional spares. If the total number of aircraft is n, and k of them are required for one mission, then the number of additional spares is $n - k$. For the component a_{ij}, with the stock m_{ijl} in the interval of length t_{il}, the reliability of the stock is $R_{ijl}(m_{ijl}) = \sum_{x=0}^{n-k+m_{ijl}} \frac{(k\lambda_{ij}t_{il})^x}{x!} e^{-k\lambda_{ij}t_{il}}$. Then the reliability for that interval (for that level i) is $R_{il} = \prod_j R_{ijl}(m_{ijl})$. Finally, the reliability of the stock for that level is $R_i = \prod_l R_{il}$, and then the overall reliability is $R = \prod_i R_i$. This model assumes that any component that is cannibalized is replaced (restored) at the next replenishment instant, if this exists. Without this assumption, the problem would be quite more complex.
   (c) If we don't assume cannibalization, the problem becomes mathematically more complex. If the stock for one component is exhausted, the aircraft requiring the component cannot fly and should be replaced with another aircraft. We can find a stock optimal for k aircraft in use, if k is not far from n, or we can assume all n aircraft are in use ($k = n$), but with failure rates for components per actual mission hour, not per flying hour. In both cases, the actual reliability will not be smaller than the calculated one. So reliability formula $R_{ijl}(m_{ijl}) = \sum_{x=0}^{m_{ijl}} \frac{(k\lambda_{ij}t_{il})^x}{x!} e^{-k\lambda_{ij}t_{il}}$ can be used.
5. Let the total storage space be C. Assign fixed storage space $C_i, \sum_i C_i = C$ to level i. If one unit of the component a_{ij} requires space c_{ij} for storage, then at the replenishment point of interval t_{il} the occupied space is $\sum_j c_{ij} m_{ijl}$, with constraint $\sum_j c_{ij} m_{ijl} \leq C_i$ for every l. Thus, at every replenishment point we have up to C_i space available. The storage space

constraints can be combined with constraints on the stock cost and handling, penalty for missing stock, etc., but we will not discuss it here.

6. Our objective is to find stock $\{m_{ijl}\}$ and stock boundaries C_i that maximize the overall reliability $R = \prod_i R_i$ under the constraints $\sum_j c_{ij} m_{ijl} \le C_i, i, l = 1,2, \dots$.

Optimization algorithm

In our approach to optimization, we can proceed in two steps. First, we find the optimal stock $\{m_{ijl}\}$ that depend on fixed storage boundaries $\{C_i\}$, and then optimize the storage boundaries $\{C_i\}$ under the condition $\sum_i C_i = C$. For the first step, the optimization reduces to separate optimizations for every time interval t_{il}, with just **one condition**: for given (i, l) find $\{m_{ijl}\}$ that maximizes $R_{il} = \prod_j R_{ijl}(m_{ijl})$ (or $logR_{il} = \sum_j logR_{ijl}(m_{ijl})$, which is more convenient, as an additive function; we can drop the factors $e^{-k\lambda_{ij}t_{il}}$ from optimization procedure, as they don't depend on m_{ijl} or C_i) under the constrain $\sum_j c_{ij} m_{ijl} \le C_i$. Several known algorithms can solve this first step with single constraint (e.g., Kettelle's algorithm and its extensions). In reality, we would not expect more than two or three boundaries C_i, so the second step would not be a big problem.

The dimensionality of the problem (the number of components) can be large (as is it in this real study) and can affect the calculation. It can be reduced significantly by grouping of components that take similar space inside single level. A group of components with equal or similar storage space per unit can be replaced by a "representative component" with the unit storage space equal to the average of the group and failure rate equal to the sum of failure rates of the group. After the final optimization, the number of spares for a single component from the group can be calculated as proportional to its failure rate out of the total number of spares for the group. As a starting point ($m_{il} = m_i$), we can take $m_{ijl} \approx \frac{\lambda_{ij}}{\sum_j \lambda_{ij}} m_i$, and calculate m_i from the condition $\sum_j c_{ij} m_{ijl} \approx C_i$, or $m_i \approx \frac{\sum_j \lambda_{ij}}{\sum_j c_{ij}\lambda_{ij}} C_i$, thus using actual storage sizes c_{ij}. Then, $m_{ijl} \approx \frac{\lambda_{ij}}{\sum_j c_{ij}\lambda_{ij}} C_i$. In this approach, stock sizes do not depend on l (time intervals t_{il}), but on available space C_i. Even if we include time through demand of a component, as $\lambda_{ij} t_{il}$, and divide m_i proportionally to demand, results for m_i and m_{ijl} will remain the same as above; the factor t_{il} will cancel in the ratio. After using those initial values m_{ijl}^0, we can find initial values C_i^0 by minimizing C_i.

The policy may be improved dynamically to assign space from levels that cannot be replenished to upper levels that can, if needed. The same algorithm can be applied to maximize reliability of the stock that can be replenished. Global optimization of this policy would be quite hard, but doing it dynamically, one step at the time would not be a problem. Dynamic programming approach can be considered for global optimization, with optimization steps at every replenishment point. *This approach works, obviously, if failed components can be stored outside the stock space; otherwise, the complete space is occupied all the time, with either unused units or failed units.*

An illustrative example

1. A carrier is on a mission for 6 months. It embarks $n = 18$ aircraft, of which $k = 12$ are required for missions over 6 months (obviously, not always the same 12).
2. There are 50 replaceable components on an aircraft of interest for storage. Some other components may be highly reliable, or cannot be replaced during the mission, and they are not considered for storage. 15 components are large and cannot be replenished during the mission, and 45 can be replenished once, after 3 months in the mission, by fly ins.

3. So we have two replenishment levels, L_1 with one replenishment interval $t_{11} = 6$ months and 15 components, and L_2 with two replenishment intervals $t_{21} = 3$ months and $t_{22} = 3$ months, and 45 components.
4. We initially assign 60% of the available storage space to level 1 components, and 40% to level 2 components. Using components' failure rates and storage spaces per components units, we can find initial stock sizes for level 1 (one interval), $m_{1,1,1}, m_{1,2,1}, \dots, m_{1,15,1}$, and for level 2, for first interval, $m_{2,1,1}, m_{2,2,1}, \dots, m_{2,45,1}$, and for second interval $m_{2,1,2}, m_{2,2,2}, \dots, m_{2,45,2}$. As those two intervals are of the same length, the spare stock sizes will be same (see Figure 1).
5. After final calculation, it is possible that some components are excluded from storage (they are either too bulky, or reliable enough), and that storage levels spaces are assigned differently, e.g., 52% to level 1, and 48% to level 2.
6. After three months, if we can use dynamic policy, and some bulky (level 1) spare component were used, assume that 10% of space was left unoccupied. We can increase storage space for level 2 to 50%, for last three months, if needed (see Figure 2).

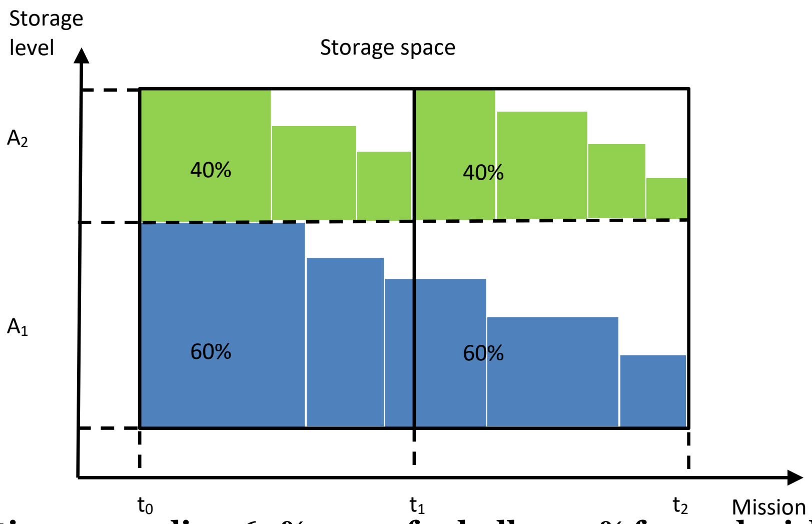


Figure 1. Static space policy, 60% space for bulky, 40% for replenishable

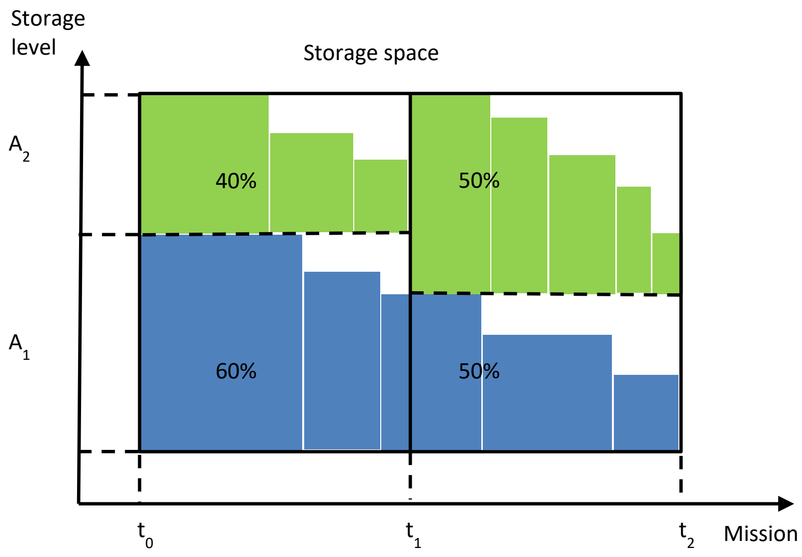


Figure 2. Dynamic space policy, storage space increased by 10% of the total for replenishable components

Kinross: A business case for machine learning methods, a progress report

Hao Zhou

Introduction

Kinross Gold Corporation is a Canadian-based gold and silver mining company founded in 1993 and amalgamated by three companies: Plexus Resources Corporation, CMP Resources and numbered company 1021105 Ontario Corp. Kinross is headquartered in Toronto and currently runs eight active gold mines in multiple countries, including Brazil, Russia, Mauritania, and Ghana. It was ranked the fourth of gold-mining companies in 2017 by InvestNews.

The company possesses a huge number of transportation assets, such as trucks transporting the gold from mining sites to multiple facilities. Because these assets are quite expensive, the maintenance and repair costs are also high. Therefore, the asset management group is seeking a solution or a model to help them manage their assets by giving them an early warning system about assets' conditions to avoid shut downs so that in-progress work will not be influenced. The early warning system allows the company to make specific plans on maintenance to avoid unexpected mechanical issues, which can minimize the operating and repairing cost.

Background and motivation

Replacement and maintenance problems involve items that degenerate with use or the passage of time resulting in failure after a certain amount of time or use. Items that deteriorate are likely to be large and costly to repair such as machines, trucks, and home appliances (refrigerators). Non-deteriorating items are usually small and relatively inexpensive, such as light bulbs or vacuum tubes. Deteriorating items require maintenance more frequently to ensure efficiency and functionality with the increment of use. In addition, the salvage value will be reduced with a longer operating time of such items. Finally, it is more cost-saving to purchase and run a new item than paying high maintenance fees doe old ones. However, if the item is replaced frequently, the investment costs will significantly increase. Thus, the maintenance problem is to determine when to replace such items and how much maintenance (particularly preventive) should be performed so that the sum of the operating, maintenance, and investment costs is minimized.

Analysis of historical data would help Kinross design a maintenance strategy and early warning system before assets shut down while working. This would help to figure out the consumption rate of each component on the assets. Ultimately, based on the records, a general ML model or mathematical model could elicit the factors resulting in the reduction of engine power and come up with a plan that provides maintenance suggestions periodically to prevent shut down.

Under the main target stated above, several sub goals have been proposed by the company in the different stages and listed in the next section. The project will start with exploratory data analysis (EDA), helping the asset management group find the relationship between the "Engine Power Derate %" and other sensor data. The next step is to deeply analyze the root cause of the decrease in engine power in terms of other factors. Finally, when these data are combined with current repair records, we will provide suggestions about when the engine needs maintenance and when other parts of machines need to be replaced to ensure the asset's working functionality.

Secondary objectives

The project has four states. As shown in Table 1, the first stage is to find out when the engine experiences derating. For example, given that the peak air filter restriction is around 8 (normal values are 0 to 6.5 kPa) and atmospheric pressure is 110kpa (normally 101 kPa), the model should be able to recognize when engine power derates. The second stage is to find out the Engine Power Derate % under certain conditions based on the first stage using input data from other sensors. The third stage is based on the Engine Derate %, with the percentage divided into four levels. The final stage is to provide a maintenance and replacement policy helping the company schedule maintenance periodically with the lowest cost followed by an optimal model or equation

Table 1. Targets in various stages

Stage number	Targets
Stage 1	Figure out when engine derating occurs
Stage 2	Figure out Engine Derate %
Stage 3	Divide data into four levels: -Level 1: causes the involved system alert indicator to illuminate -Level 2: operation of the machine should be changed -Level 2s: possible severe damage to components on the machine may occur -Level 3: machine needs to be shut down immediately
Stage 4	Model construction to provide optimal policy

As shown above, stage 3 has four levels; each level corresponds to different engine conditions, requiring different actions to be performed:

- Level 1 is the warning level and will cause the system alert indicator to illuminate. A level (1) warning indicates that the operator should be aware of a condition of one or more of the machine systems.
- Level 2 indicates that the operation of the machine should be changed or maintenance should be performed. Possible damage to components on the machine may occur.
- Level 2-S indicates that the operation of the machine should be changed or maintenance should be performed; there is a possibility of severe damage to components on the machine.

- Level 3 indicates that the machine needs to have an immediate, safe emergency engine shutdown process. Possible injury to the operator or severe damage to components may occur.

Data

Various types of data in terms of the machine are recorded in the raw dataset. The asset management group selected six main features from the raw dataset. Table 2 displays the feature names and detailed descriptions.

Table 2. Feature categories

Feature Name	Description
Asset Number	Unique ID of each asset In total 5 assets are recorded from DT82 to DT86
Timestamp	Time series data record when sensors report data to the backend. 15 are performed per month (12 month from 2019 and 3 month from 2020)
Loaded	Record if the machine is in the loading condition. Reported in binary value. 1: Fully loaded 0: Not fully loaded/unloaded
Moving	Record the moving condition of the machine. Binary value. 1: Moving 0: Stationary
Service Hour	Record the total service hour corresponding to each timestamp.
Speed	Moving speed while operating the asset corresponding to each timestamp.
Sensors	4 essential sensors are included in the raw datasheet, reporting the machine condition. -Engine Power Derate % -Peak Air Filter Restriction -Right minimum Left Exhaust Temperature (RT-LT EXH Temp)

This dataset contains 4 types of sensors. From the company's perspective, "Engine Power Derate %" is the target feature in this project; the rest are the most important and correlated sensors to the target feature. All these sensors record and report different data at 10 to 15 second intervals to reflect the real-time working condition of assets. Each sensor has a threshold value. Once the reported data exceed such values, a warning light or alarm will occur. Detailed descriptions are listed below.

Engine Power Derate % is used to request and read the present engine power derating percentage. 0 value indicates engine power if NOT derated and 25 is the maximum percentage an engine can hold.

Atmospheric Pressure Sensor is a device that generates a signal proportional to atmospheric pressure. This is usually an analog sensor. It is reported in kPa and the default sea level is 101kPa. There will be a reduction in the performance of the engine if there is a restriction in the air inlet system or the exhaust system. The air flow through the air cleaner may have a restriction. The pressure at the restriction of the air flow *must not exceed 6.25 kPa*.

Peak Air Filter Restriction is the device to show the peak air filter restriction for a given period of time. The engine ECM calculates the value by subtracting the atmospheric pressure value from the turbo inlet pressure value. It indicates the degree to which the engine air filter is plugged. *Engine power is derated 2% per kPa of pressure difference above 6.5kPa, up to a maximum value of 20%* total engine derate. If the engine ECM senses that either one of the pressure sensors are sending an incorrect value, the engine power is also derated up to 20 percent.

RT-LT EXH Temp are numerical data calculated by subtracting the left manifold exhaust temperature from the right manifold exhaust temperature. The data are reported in Celsius and the normal range is from -20 to 20. The outbound of the normal range will lead to a stage one alert.

Exploratory data analysis

Initially, I divided the data based on the asset number as well as the sensor types, since all data in the raw dataset are mixed together [Appendix A]. While separating the dataset into smaller one based on the asset number, I found that some assets' engine power derating only contain binary values (either 0 or 25) such as DT83, but others such as DT85 contain continuous values (from 0 to 25 in every 0.5 intervals) [Appendix B]. Based on the company's interpretation, the data in assets such as DT83 were discretized into binary form. As the company wants to use the ML techniques to model the derating percentage in the stage 2, the project will mainly focus on using continuous data. If we take both binary and continuous data into consideration, the model may not be able to learn the correct trends and return a reliable result, because even though the engine derates 8%, it is displayed in the dataset as 25%. Thus, all work in the first two stages will be carried out based on the data of asset DT85.

To find the relations between target feature (Engine Derate %) and others, and when engine derating occurs in the first stage of the work, a new feature named "If Derate" was elicited in binary values. The feature describes if engine power is derating corresponding to the specific timestamp, because in the first stage, the target is to explore when the engine power is derated. This feature can enhance the correlations between target features and other factors regardless of the continuous value of the derating percentage. With this new target feature, the model will only consider if there a deration occurs, but not consider how much it derates.

While cleaning the data, I found that the amount of data was inconsistent. As shown in Figure 1, in 2019, 15245 timestamps were recorded in the raw dataset for DT85. However, none of the sensors contained the same number of records. For example, Engine Power Derate % contained 15239 records and Atmospheric Pressure only had only 15093 records. For consistency, I manually supplemented the missing data by giving a NULL value to ensure all four sensors reported correct data at the correct timestamp, so that I could use them to build the ML model. After removing the NULL value, 15093 data were left in the dataset.

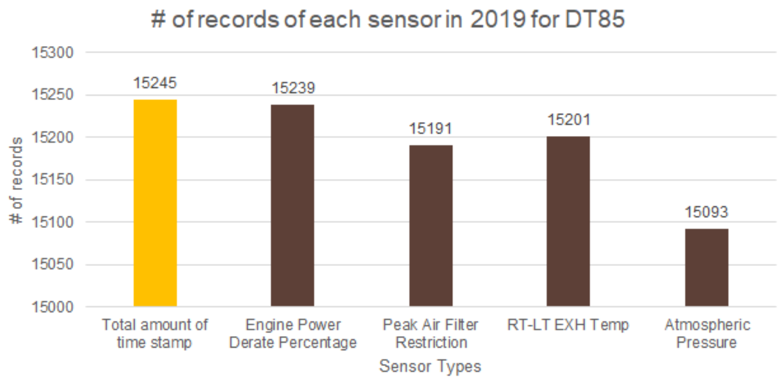
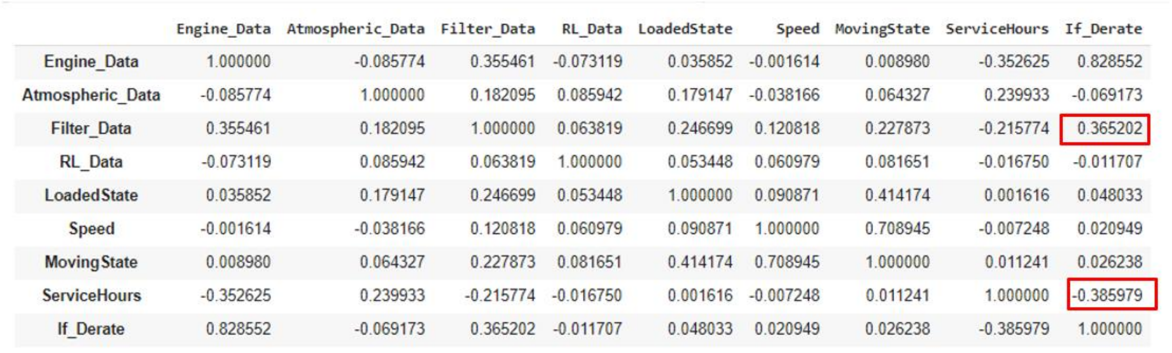


Figure 1. Total count of records of each sensor for DT85

After cleaning the data, I checked the correlations. As shown in Figure 2, the strongest correlation between If_Derate and other sensors was only 0.3652 which is relatively weak in general. To figure out the potential reason for such a low correlation, I created a scatter plot.



	Engine_Data	Atmospheric_Data	Filter_Data	RL_Data	LoadedState	Speed	MovingState	ServiceHours	If_Derate
Engine_Data	1.000000	-0.085774	0.355461	-0.073119	0.035852	-0.001614	0.008980	-0.352625	0.828552
Atmospheric_Data	-0.085774	1.000000	0.182095	0.085942	0.179147	-0.038166	0.064327	0.239933	-0.069173
Filter_Data	0.355461	0.182095	1.000000	0.063819	0.246699	0.120818	0.227873	-0.215774	$\boxed{0.365202}$
RL_Data	-0.073119	0.085942	0.063819	1.000000	0.053448	0.060979	0.081651	-0.016750	-0.011707
LoadedState	0.035852	0.179147	0.246699	0.053448	1.000000	0.090871	0.414174	0.001616	0.048033
Speed	-0.001614	-0.038166	0.120818	0.060979	0.090871	1.000000	0.708945	-0.007248	0.020949
MovingState	0.008980	0.064327	0.227873	0.081651	0.414174	0.708945	1.000000	0.011241	0.026238
ServiceHours	-0.352625	0.239933	-0.215774	-0.016750	0.001616	-0.007248	0.011241	1.000000	$\boxed{-0.385979}$
If_Derate	0.828552	-0.069173	0.365202	-0.011707	0.048033	0.020949	0.026238	-0.385979	1.000000

Figure 3. Correlations before dropping NULL values and removing outliers

In the scatter plot, the target feature (Engine Power Derate %) was the y value and other sensors' data were the x value (Figure 4). From the plot, I found some data were outliers which may adversely impact the model construction in a future stage. For example, the normal value range of RL_LT EXH Temp Sensor was between -20 to 20, but some values less than -30000 appeared in the dataset. Such values are impossible in the general physical environment and should be removed.

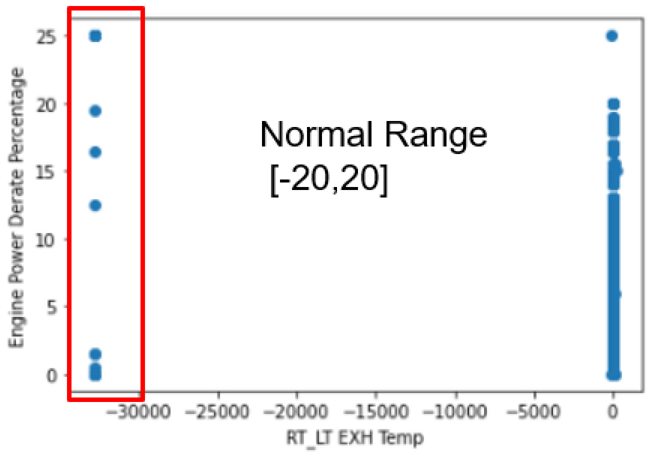
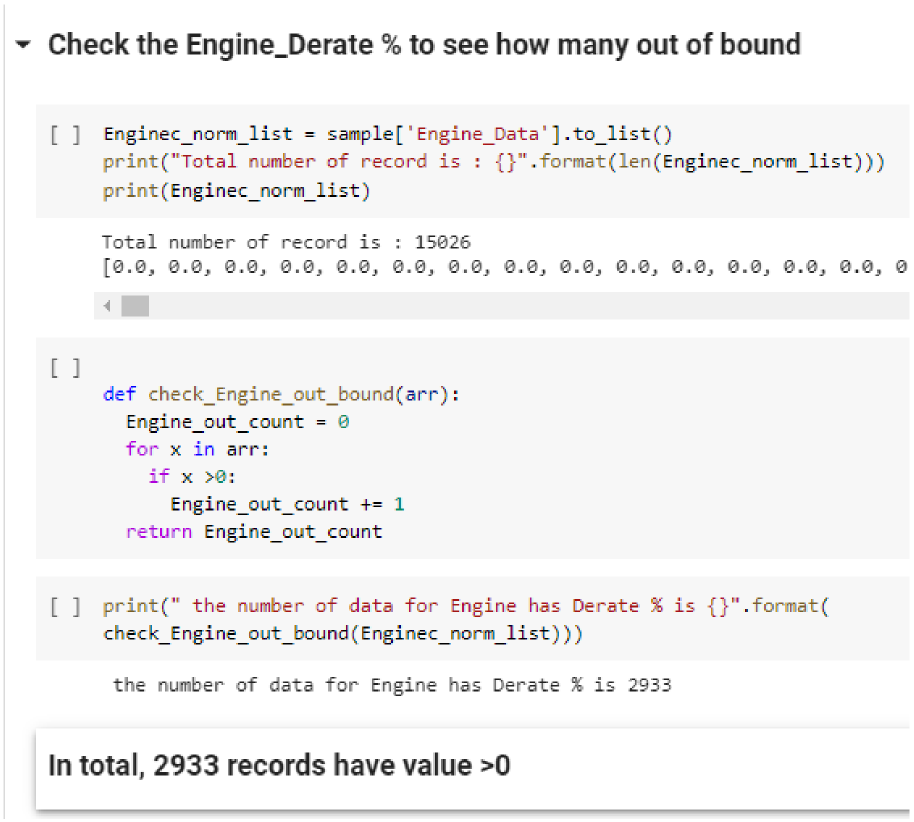


Figure 4. Scatter plot of RL-LT EXH Temp

After removing outliers, I wrote a function to check the amount of data where engine power derates. Out of a total 15026 data, 2933 data have a value greater than 0 (Figure 4). However, this is an imbalanced condition because the number of “0” values is around 5 times more than the “1”s. To improve the imbalance, I applied a heuristic cut to remove some data. For example, all assets under 8400 hrs service time have no deration in engine power. I removed all these data and reformed the dataset.


Check the Engine_Derate % to see how many out of bound

```
[ ] Enginec_norm_list = sample['Engine_Data'].to_list()
    print("Total number of record is : {}".format(len(Enginec_norm_list)))
    print(Enginec_norm_list)
```

```
Total number of record is : 15026
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0
```

```
[ ]
    def check_Engine_out_bound(arr):
      Engine_out_count = 0
      for x in arr:
        if x >0:
          Engine_out_count += 1
      return Engine_out_count
```

```
[ ] print(" the number of data for Engine has Derate % is {}".format(
    check_Engine_out_bound(Enginec_norm_list)))
```

```
 the number of data for Engine has Derate % is 2933
```

In total, 2933 records have value >0

Figure 5. Filter function of Engine Power Derate %

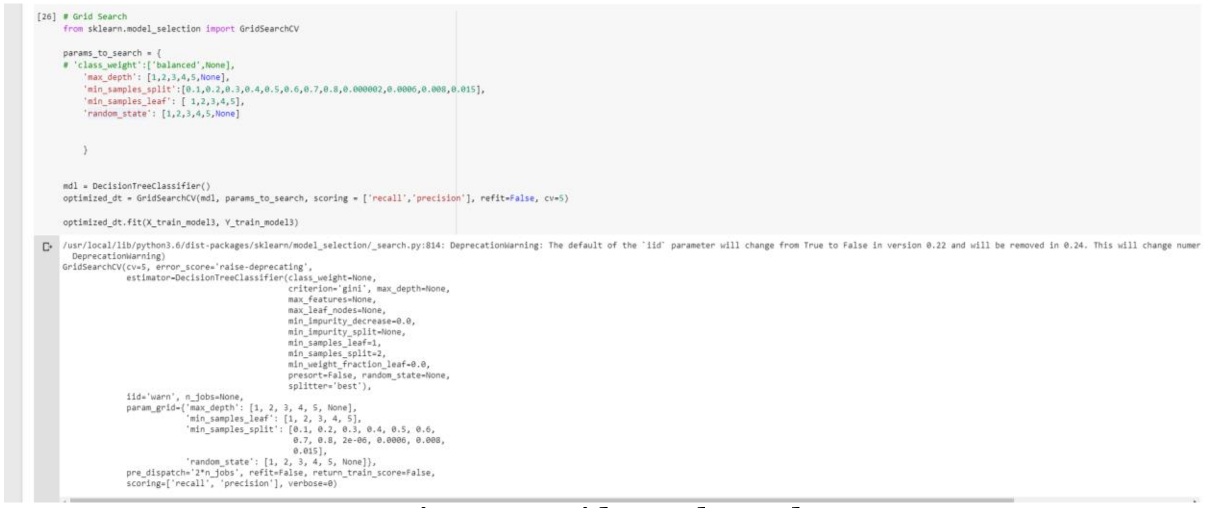
Model construction

After the EDA process, I built a ML model to predict when there will be a reduction in the engine power given a set of input data. Initially, the dataset was divided into a training set which contained about 75% of the data, and a validation set including 25% of the data. In the previous process, I noticed that the imbalance condition still exists in the dataset. To combat this, I decided to upsample the data. Before using upsampling, the amount of “If_Derate ==1” is 2933, but there were 11160 rows where “If_Derate ==0”. The upsampling method returned an equal amount for both types of data, 11160.

I used both logistic regression and Randomforest method to build the model. Logistic regression is a method returning a binary value (either 0 or 1) as the final result. In the mode, we set a threshold value; any predicting result below this threshold would be 0 and above this value would be 1. Compared to logistic regression, the Randomforest method has more parameters allowing us to adjust and optimize by using a grid search to produce the best results. The Randomforest is a supervised learning algorithm. It consists of multiple low correlation decision trees and merges them together to get a more accurate and stable prediction. The Randomforest follows a majority rule helping to make decisions. For example, if 10 decision trees are presented where 7 return the result “1” and the rest return “0”, the final result will be “1”.

I chose the parameters carefully based on the dataset, and provided multiple input values to run through the grid search (Figure 6). The best results were then returned. Other models like linear regression and decision tree are not much help because our target value in stage 1 is “If_Derate".

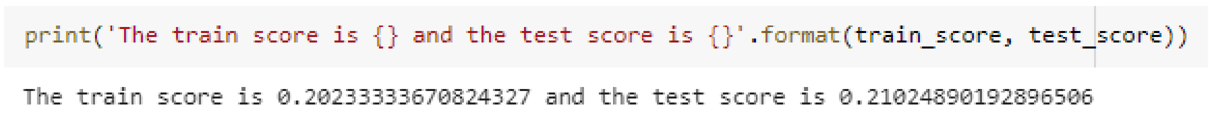
The result of the logistic regression is presented in Figure 7; both the training and testing accuracy are around 0.2. Low correlations between target features and others may lead to such a result, as the trend cannot be studied properly by the model. However, although I applied the Randomforest and the result showed a certain increase, it was still not ideal (Figure 8); the training score was approximately 0.25 and the testing score about 0.37.



```
[26] # Grid Search
     from sklearn.model_selection import GridSearchCV

     params_to_search = {
     # 'class_weight':['balanced',None],
         'max_depth': [1,2,3,4,5,None],
         'min_samples_split':[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.000002,0.0006,0.008,0.015],
         'min_samples_leaf': [ 1,2,3,4,5],
         'random_state': [1,2,3,4,5,None]


         }

     mdl = DecisionTreeClassifier()
     optimized_dt = GridSearchCV(mdl, params_to_search, scoring = ['recall','precision'], refit=False, cv=5)

     optimized_dt.fit(X_train_model3, Y_train_model3)
```

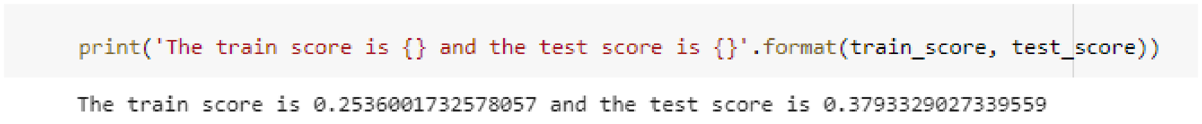
```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_search.py:814: DeprecationWarning: The default of the `iid` parameter will change from True to False in version 0.22 and will be removed in 0.24. This will change numer
  DeprecationWarning)
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=DecisionTreeClassifier(class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort=False, random_state=None,
                                              splitter='best'),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': [1, 2, 3, 4, 5, None],
                         'min_samples_leaf': [1, 2, 3, 4, 5],
                         'min_samples_split': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6,
                                               0.7, 0.8, 2e-06, 0.0006, 0.008,
                                               0.015],
                         'random_state': [1, 2, 3, 4, 5, None]},
             pre_dispatch='2*n_jobs', refit=False, return_train_score=False,
             scoring=['recall', 'precision'], verbose=0)
```

Figure 6. Grid search result



```
print('The train score is {} and the test score is {}'.format(train_score, test_score))

The train score is 0.20233333670824327 and the test score is 0.21024890192896506
```

Figure 7. Predicting accuracy of logistic regression



```
print('The train score is {} and the test score is {}'.format(train_score, test_score))

The train score is 0.2536001732578057 and the test score is 0.3793329027339559
```

Figure 8. Predicting accuracy of Randomforest

The results shown above suggest the ML model may be not an appropriate way to predict when there will be a reduction in engine power. Low data quality may be a contributing factor leading to such a result as the correlations between target features and others are too weak. Some problems related to the data are listed below and future work is proposed later in the report to see how other methods can be applied to find a solution.

Issues and findings

This section mentions some issues we encountered while doing the EDA and building the model. We were left with two major questions in terms of the data.

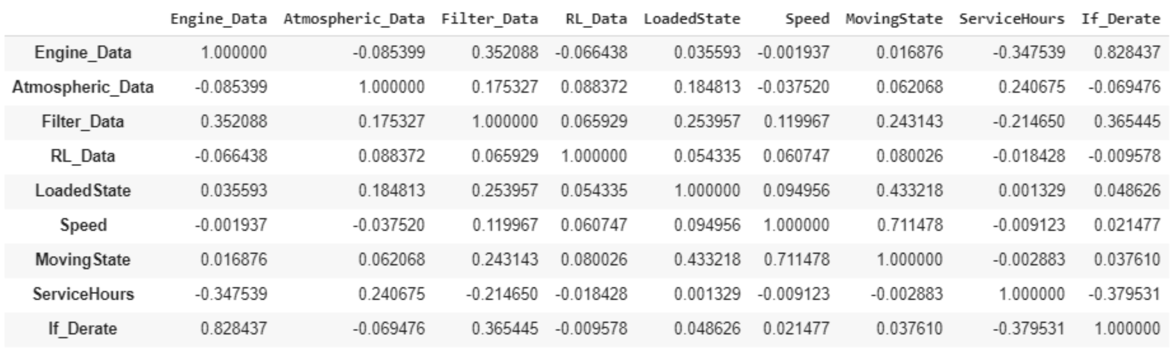
Normally, sensors report data every 10 to 15 seconds. However, sometimes, sensors report NULL values or even do not report real-time data as mentioned previously. A glitch in the sensors and network delay may be a possible interpretation of this problem. As shown in Table 3, there were 15245 records in year 2019, but all sensors reported less than that number.

Table 3. Total count of records of each sensor for DT85

Feature Name	Amount of Data
Timestamp	15245
Engine Power Derate Percentage	15239
Peak Air Filter Restriction	15191
RT-LT EXH Temp	15021
Atmospheric Pressure	15093

The second problem is that the "Engine Power Derate %" displays a deration, but other sensors report normal values.

As stated previously, only asset DT85 has continuous data for Engine Power Derate % displaying at 0.5 intervals from 0 to 25 (Appendix A). Others are binary values (either 0 or 25). Therefore, the analysis focuses on the DT85. While doing the EDA, after removing the outlier and doing the heuristic cuts, the strongest correlation was 0.36544 which increased a little compared to the original one. Nevertheless, this correlation is still quite weak.



	Engine_Data	Atmospheric_Data	Filter_Data	RL_Data	LoadedState	Speed	MovingState	ServiceHours	If_Derate
Engine_Data	1.000000	-0.085399	0.352088	-0.066438	0.035593	-0.001937	0.016876	-0.347539	0.828437
Atmospheric_Data	-0.085399	1.000000	0.175327	0.088372	0.184813	-0.037520	0.062068	0.240675	-0.069476
Filter_Data	0.352088	0.175327	1.000000	0.065929	0.253957	0.119967	0.243143	-0.214650	0.365445
RL_Data	-0.066438	0.088372	0.065929	1.000000	0.054335	0.060747	0.080026	-0.018428	-0.009578
LoadedState	0.035593	0.184813	0.253957	0.054335	1.000000	0.094956	0.433218	0.001329	0.048626
Speed	-0.001937	-0.037520	0.119967	0.060747	0.094956	1.000000	0.711478	-0.009123	0.021477
MovingState	0.016876	0.062068	0.243143	0.080026	0.433218	0.711478	1.000000	-0.002883	0.037610
ServiceHours	-0.347539	0.240675	-0.214650	-0.018428	0.001329	-0.009123	-0.002883	1.000000	-0.379531
If_Derate	0.828437	-0.069476	0.365445	-0.009578	0.048626	0.021477	0.037610	-0.379531	1.000000

Figure 9. Correlations after removing NULL values and heuristic cuts

To figure out why the correlation is so weak among target features and other sensors, I used a scatter plot analysis. Based on Figure 10, for "Peak Air Filter Restriction Sensor", about a third of the data were over 6.25 kPa when engine power derating occurs; this matched the correlations between this sensor and target feature. However, for the Atmospheric Pressure Sensor, the majority of data were in the normal range when power was reduced. Very limited data were out of normal range leading to derating. A similar trend was displayed in RT-LT EXH Temp Sensor, where most are located between -20 to 20 when engine power derates. In addition, when these sensors showed the machine had problems, the decrease in engine power remained.

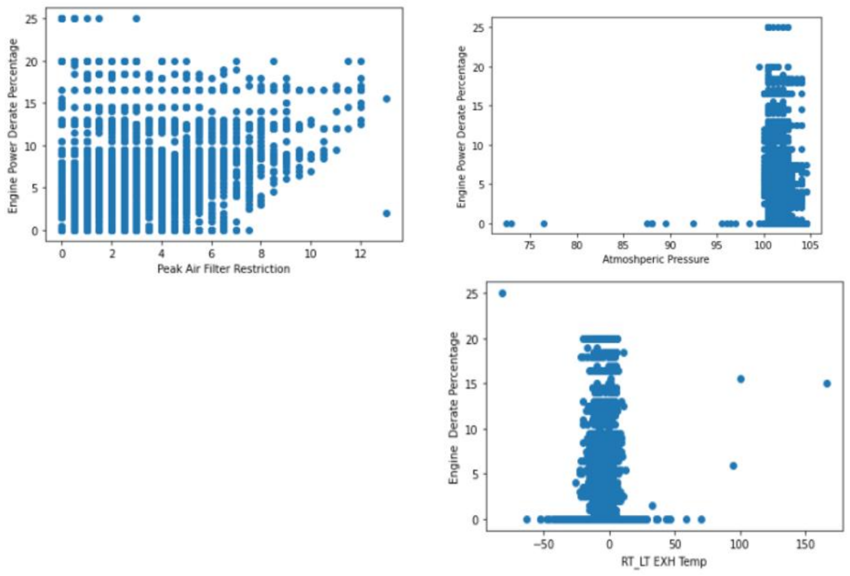


Figure 10. Scatter plot of sensors

To check how many times engine power derating occurs for DT85, I counted the total amount of data which were out of the set threshold. The result is displayed in Figure 11. In total, 2933 times of power reduction happened for the engine, but the atmospheric pressure sensor only reported 8 anomalies, the peak air filter sensor reported 296, and the RT-LT EXH Temp showed 526. After doing the simple subtraction (2933 - 8 - 296 - 526 = 2103), there were still 2103 times that the engine power sensor reported issues, but other sensors reflected everything was working well.

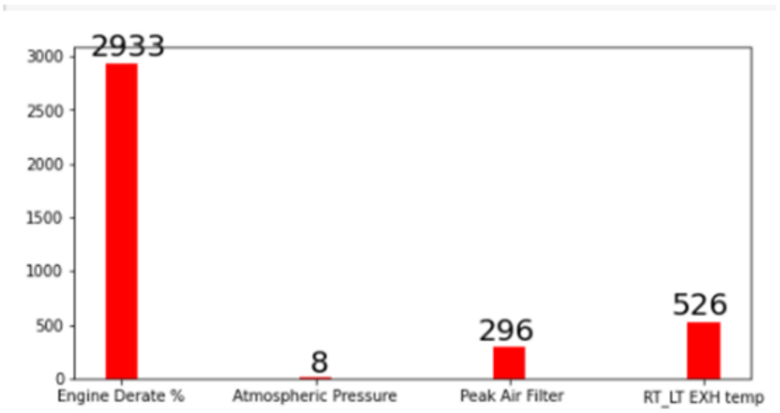


Figure 11. Counting out of control data for each sensor

The findings shown above are quite contradictory, because initially we believed the three sensors would be the main contributing factor leading to derated engine power. One possible interpretation is that there are some other important sensors not presented in the dataset, such as the power supply of the battery, the temperature of engines, and carbon deposition issues. All these factors have a certain impact on the power system leading to the reduction of engine power from a mechanical engineering perspective.

Future work

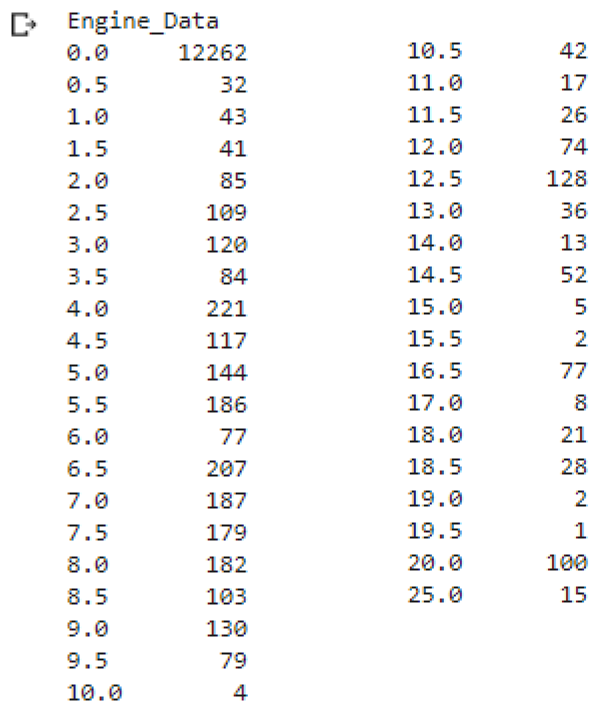
Based on the questions above, too many factors are unknown. The truck has a very complex power system structure, and the engine power is related to all components, so it is hard to draw conclusions based on the data we possess. More information is required on the power system. In addition, the dataset currently only has 15 months of data and is in an imbalanced condition, which may be not sufficient for the model to catch the trend appropriately. Therefore, additional data with higher quality are necessary for future work.

The next step is to improve the current ML model to see if it is a feasible way to find out under what condition the engine power derates, because it is hard to apply ML in the maintenance & replacement field. If the accuracy is still at a low level, then in future work, we have to change the methodology to solve the problems instead of seeking ML solutions.

Once the methodology has been determined, the work can be moved to stage 2, looking for the specific derating % by giving the input.

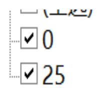
Appendix A: Value of Engine Power Derate %:

DT85's value is in continuous form



```
Engine_Data
0.0     12262           10.5       42
0.5        32           11.0       17
1.0        43           11.5       26
1.5        41           12.0       74
2.0        85           12.5      128
2.5       109           13.0       36
3.0       120           14.0       13
3.5        84           14.5       52
4.0       221           15.0        5
4.5       117           15.5        2
5.0       144           16.5       77
5.5       186           17.0        8
6.0        77           18.0       21
6.5       207           18.5       28
7.0       187           19.0        2
7.5       179           19.5        1
8.0       182           20.0      100
8.5       103           25.0       15
9.0       130
9.5        79
10.0        4
```

Other assets' values are in binary form (either 0 or 25)



☑ 0
☑ 25

TTC: Detecting power rail anomaly from FLIR thermal images

Tushar Aggarwal

Background (from 2018 student report)

TTC Line 3 Scarborough is a 6.4KM light metro line connecting 6 stations from Kennedy to McCowan. It's about 40 years old now and going to be replaced by the planned extension of Line 2 Bloor-Danforth subway to Scarborough Centre. While the new project being built, TTC still need to keep Line 3 running.

In May 2017, a train was damaged during operation, because a power rail anomaly caused high temperature which melted the metal. This resulted in passenger evacuation and line closure.

Problem

As a preventative measure and to avoid any interruptions on Line 3. TTC regularly scan the power rails with FLIR thermal cameras installed on the back of the trains. The thermal camera videos are then reviewed by TTC staff to identify any anomalies and high thermal heat zones on the power rail which are fixed by the maintenance staff.

However, this is a manual and cumbersome task as all the videos are required to be observed which can be difficult as it requires long-span focus. Therefore, using a computer vision system for anomaly detection could replace this cumbersome task. Therefore, TTC has tasked C-More Labs to develop an anomaly detection project that uses the recorded FLIR thermal videos and is able to identify anomalies on power rails if it exists in specific frame.

Approach

For the previous approach the student used a Opensource TensorFlow Object Detection API where the student used two different convolution neural network object detection model: Single Shot Detector (SSD) and Faster RCNN. However, both these models are few years old and since computer vision is a rapidly advancing field we will be using new advance computer vision models.

In April 2020, A popular object detection algorithm You Only Look Once V4 (YoloV4) was published which showed higher accuracy and faster detection time compared to SSD and F-RCNN algorithms. Hence, we will be training a YOLO V4 model to improve the object detection model of the previous students.

Data

In terms of data collection, TTC provided one pair of infrared videos along Line 3, both northbound and southbound, recorded in December 2017. Each video has about 16,000 frames, and 45 anomalies in total were identified by manual observation. To construct a dataset, all frames were exported and those frames with anomaly presented are picked out for labeling. The labelling process was done by using Microsoft VoTT (from 2018 student report)

In addition to the video from December 2017, we have another set of videos from May 2018 with approximately 8000 frames, however, videos from 2018 may summer do not have any anomalies detection. Additionally, the previous students extracted all the frames and applied a colour palette to turn gray scale images into RGB images.

Current progress

We currently have 780 frames from winter data set with anomalies detected. The annotation for the images was converted into YoloV4 Annotations and the dataset was run through the YoloV4 model to remove any bugs from the model/code.

In the first analysis the model ran for 6000 iterations; however, the model was overfitting after 3000.

Our results are as follows using the 3000 iteration's weight:

Detection Counts = 128,
Unique Truth Counts = 99
Class ID = 0, (name = Anomaly)
Average Precision (ap) = 93.74%
True Positive (TP) = 88,
False Positive (FP) = 6
False Negative (FN) = 11

for Confidence Threshold = 0.25,
    Precision = 0.94,
    Recall = 0.89,
    F1-score = 0.91
    Average IoU = 68.74 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.937417, or 93.74 %

Total detection time: 2 seconds

Results using 6000 iteration's weight:

```
Detection Counts = 116,
Unique Truth Counts = 99
Class ID = 0 (name = Anomaly),
Average Precision (ap) = 83.88%
True Positive (TP) = 88,
False Positive (FP) = 10,
False Negative (FN) = 11

for Confidence Threshold = 0.25,
        Precision = 0.90,
        Recall = 0.89,
        F1-score = 0.89
        Average IoU = 66.44 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision
(mAP@0.50) = 0.838818, or 83.88 %

Total detection time: 1 second
```

Next steps

- For the next step, we will be labelling May 2018 videos for anomalies to the best of our ability.
- We will use photo software to paint some additional anomalies into true negative frames to train the model on different types of tracks.
- We will perform cross validation to find the best model.
- We will run the model on summer videos to see if the model is able to predict anomalies correctly.

CEA: Sequence pattern mining

Chengjiang (Gavin) Zou

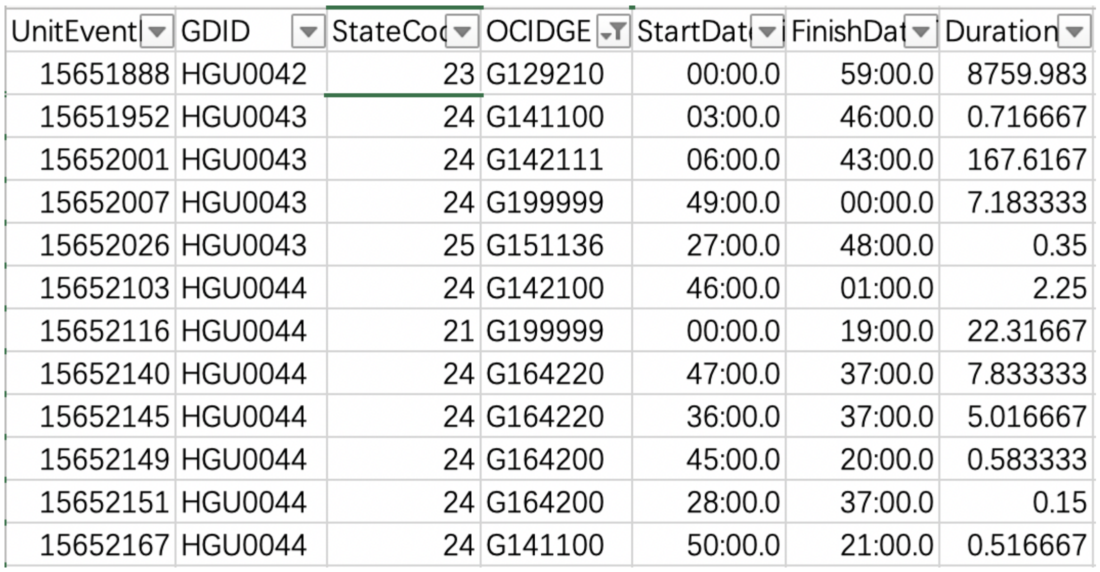
Introduction

Canadian Electricity Association (CEA) is the national voice for safe, secure and sustainable electricity for all Canadians. With today's increasingly advanced technology, large volumes of data are created automatically by the ubiquitous equipment sensors in order to find more insights of equipment operation. Currently, CEA operates an Equipment Reliability Information System (ERIS) to keep a record of continuous state and outage component data of hydroelectric and fossil generating units with timestamps.

This project uses machine learning algorithms and codes to detect patterns and rules of the equipment operation by inspecting the continuous records from CEA datasets. These patterns may indicate equipment health and help to make an appropriate maintenance decision for next time. For CEA datasets, sequential pattern mining algorithms are conducted to unlock the hidden patterns of the continuous records. As a result, a set of preventive maintenance policies are derived based on the output of sequential pattern mining.

Data exploration

The Equipment Reliability Information System (ERIS) records the continuous equipment data that shows the operating state and outage status of each unit with timestamps. For this project, the purpose is to make preventive maintenance policies to save costs of forced outage. Thus, the columns kept are UnitEventID (unique event ID generated by the system), GDID(unique unit ID created by the system), ForcedOutageType(forced outage type as per the manual), OCIDGE(Component outage code), StartDateTime, FinishDateTime, Duration.



UnitEvent	GDID	StateCo	OCIDGE	StartDat	FinishDat	Duration
15651888	HGU0042	23	G129210	00:00.0	59:00.0	8759.983
15651952	HGU0043	24	G141100	03:00.0	46:00.0	0.716667
15652001	HGU0043	24	G142111	06:00.0	43:00.0	167.6167
15652007	HGU0043	24	G199999	49:00.0	00:00.0	7.183333
15652026	HGU0043	25	G151136	27:00.0	48:00.0	0.35
15652103	HGU0044	24	G142100	46:00.0	01:00.0	2.25
15652116	HGU0044	21	G199999	00:00.0	19:00.0	22.31667
15652140	HGU0044	24	G164220	47:00.0	37:00.0	7.833333
15652145	HGU0044	24	G164220	36:00.0	37:00.0	5.016667
15652149	HGU0044	24	G164200	45:00.0	20:00.0	0.583333
15652151	HGU0044	24	G164200	28:00.0	37:00.0	0.15
15652167	HGU0044	24	G141100	50:00.0	21:00.0	0.516667

Figure 1. Sample datasets

Basic concepts and algorithms

Frequent pattern mining

Frequent pattern is a pattern such as itemset or subsequence which frequently occur in a data set. For instance, 'beer' and 'nuts' is a frequent itemset if 'beer' and 'nuts' often appear together in transaction data sets. In terms of itemset, time is not crucial to conduct the frequent pattern mining. However, time is an essential key for subsequence. For frequent pattern mining, it is a process to unlock the hidden frequent patterns or associations rules by using provided datasets.

Transaction ID	Itemset
10	a,b,d
20	a,c,d
30	a,d,e
40	b,e,f

Table 1. Sample dataset of frequent pattern

There are several key terms for frequent pattern mining:

- Itemset: an itemset is a set of one or more items. Items can be a,b,d, etc.
- Support: probability that a transaction contains a and b. Support (a⇒b) =P(a∪b)
- Confidence: conditional probability that a transaction includes both a and b. confidence(a⇒b)= P(b|a) = support(a∪b)/support(a)= support count(a∪b)/support count(a).
- Lift: the probability that a transaction having itemset a given it contains itemset b Lift (a and b) = Support of a and b / (Support of a * Support of b)

- Sequential rule: relationship between two itemset, and it indicates that if a occurs in sequences, b will occur afterward in the same sequence.

Sequential pattern mining

Unlike transaction itemset, sequence data contain a set of event sequences, such as CEA inspection datasets. Sequential pattern mining concentrates on mining sequences. In other words, given a set of sequences and support threshold, it should find the complete set of frequent subsequence.

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

Table 2. Sequence database

An element may contain a set of items. Items within an element are unordered which can be listed alphabetically. For example, <a(bc)dc> is a subsequence of <a(abc)(ac)d(cf)>. Given the support threshold min_sup =2, <(ab)c> is a sequential pattern.

Algorithms for sequential pattern mining

There are six main types of algorithms for sequential pattern mining:

- Horizontal format-based mining: Initial Apriori algorithm and Apriori-based method -- GSP.
- Pattern-growth methods: FreeSpan and PrefixSpan.
- Vertical format-based mining: SPADE, SPAM, CM-SPADE, CM-SPAM.
- Constraint-based sequential pattern mining: SPIRIT and BIDE.
- Mining closed sequential patterns: CloSpan
- Hybrid: LASH

After comparing all algorithms and combining with the case project, the sequential pattern mining algorithms for CEA datasets are: PrefixSpan, CM-SPADE and LASH.

Sequential pattern mining with CEA dataset

Data pre-processing

There are numerous inspecting data produced through the operation and maintenance of generating units which records the operation and outage data. In this project, UnitEventID, GDID, ForcedOutageType, OCIDGE, StartDateTime, FinishDateTime and Duration are kept as model features. The project target is to use sequential pattern mining to get the preventive maintenance policies. Thus, only the generating units without an outage type should be removed.

In other words, the state code: Forced Outage (21), Maintenance Outage (24), Planned Outage (25) will remain.

Meanwhile, according to Generation Manual (Bluebook), some OCIDGE codes mean that the generating unit will appear in the outage due to external factors. These external factors will be useless to make a preventive maintenance policy. Thus, these external factors should be removed too.

Most importantly, the raw CEA data are not a sequence dataset that only records a single item in each event. Thus, the key step in data pre-processing is to conduct the time aggregation of multiple items within a specific time period. After these data pre-processing, the inspection data will require the SPMF library format.

Algorithm implementation

Algorithms for sequential pattern mining and association rule mining are implemented on Spark. And PYPI provides a python-version of SPMF library that contains 174 sequential pattern mining algorithms. For the case study, PrefixSpan, CM-SPADE and LASH are used to conduct data mining. There are several parameters for these algorithms: Minimum antecedent support (%), Minimum rule confidence (%), Minimum Support, Maximum Support, Minimum Confidence, Maximum Confidence. After setting these parameters for different algorithms, the model's output is frequent pattern or association rules.

Conclusion

In the case study, there are some interesting insights besides these CEA datasets by using sequential pattern mining. The results show that there are some hidden rules related to the outage of unique components. In other words, after outage of component A, the probability of outage of component B is much higher than other components. In addition, the frequent pattern mining can help to find more dependencies between the outage component code and unit state code. According to the frequent pattern, it is able to predict the next outage code or state code of a unique component. However, sequential pattern mining algorithms heavily rely on the principal parameters - support. For CEA sequential dataset, it is too general to solve the problem only based on support. Thus, in future work, how to use more constraints to conduct sequential pattern mining will be the key to improve accuracy of preventive maintenance policies.

DND: Steps toward a condition-based maintenance model for the DND city-class frigate fleet PDEs

Janet Lam

Introduction

In previous meetings, we presented our work in modelling the DND city-class frigate fleet with an age-based model. Throughout the balance of the year, we continued to work through the challenges in building a robust age-based model, and are now ready to incorporate the oil and coolant inspection readings. This report summarizes that work leading up to this point.

More data preparation and cleansing

Following the feedback from industry experts, another review of the dataset was performed. The approaches taken are described in this section.

Dates and times

As is so often the case when working with different programs, we encountered some issues with dates being formatted inconsistently throughout the dataset. A common problem is when a program stores its dates as dd-mm-yyyy while many American database programs expect dates to be recorded as mm-dd-yyyy. When we are lucky or careful, we can identify this problem because some dates are simply invalid (e.g. The 10th day of 17th month is certainly supposed to be the 17th day of the 10th month)

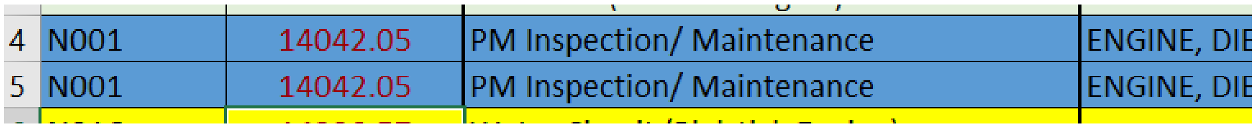
Another challenge we had with the maintenance records is that the hours on the propulsion diesel engine (PDE) were only recorded once per month. This meant that any event that occurred in the middle of the month were assigned to a working age that was taken at the end of that same month. While this in and of itself is not a great issue, a bigger problem occurs when there are multiple events occurring at the same time, as they are assigned the same time stamp, and thus appears to occur simultaneously.

To address this issue, we incorporated the more detailed working age information available in the oil and coolant conditioning analysis (OCCAP) program, and linearly interpolated the workings ages according to the calendar dates for each record.

Recoding PMs and CMs

At the beginning of the project, we were given clear directions on how to interpret the different records. In particular, the order type codes were identified as preventive maintenance codes, or corrective maintenance codes. We were challenged when a group of entries coded as corrective maintenance codes were found to be accompanied by a record description that suggested that it was a preventive maintenance entry.

As seen in Figure 1, the order type is coded "N001", which usually indicates a corrective maintenance action, but it is clearly accompanied by a description that indicates a preventive maintenance action. After clarification from DND, we decided to treat all similar entries as preventive maintenance actions.



4	N001	14042.05	PM Inspection/ Maintenance	ENGINE, DIE
5	N001	14042.05	PM Inspection/ Maintenance	ENGINE, DIE

Figure 1. Sample entry coding challenge

Removing short lives

In our first look at the data, we found that there were several records that showed very short lives of the equipment.

We can see in Figure 2 where the red triangles indicate system failures that they are very close together. This will strongly skew our results to show that the life of the engine is very likely to fail in the early days, and less likely to fail as it ages.

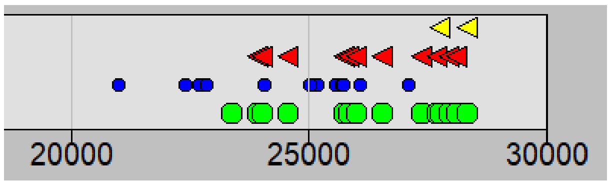


Figure 2. Sample history showing very short lives

Consequently, we manually removed histories that appeared to be duplicates of previous failures, based on the timing and the information associated with the records.

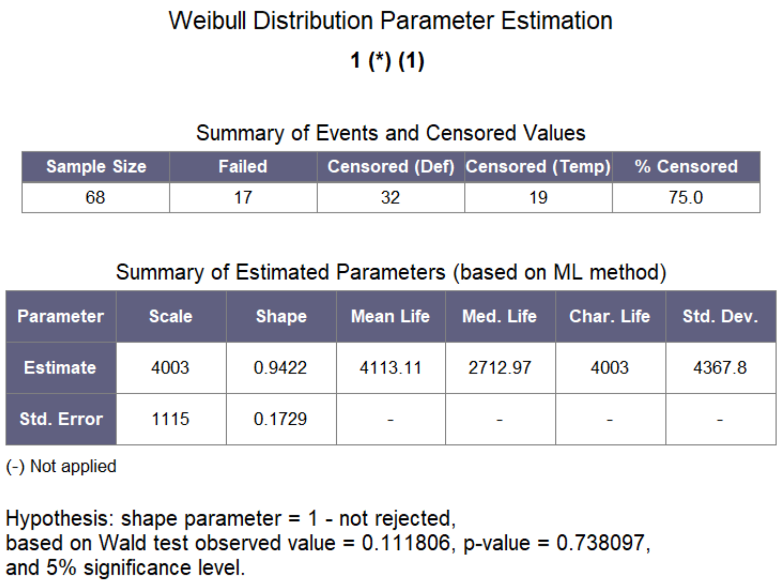
Oil and coolant only

Next, we filtered out our results to focus only on the records that were mostly likely to be affected by, or to affect the oil and coolant conditioning readings. Since we intend to use the OCCAP readings as our covariates, it makes sense to only use events that are related to the oil and coolant systems.

Results

After the data were cleaned up once more, we performed Weibull analysis on the data. The results are given in
Figure 3 below. The key takeaway from these results is that there is no evidence that the shape parameter of the Weibull distribution is not 1. That is, the data as it stands suggests that the failure rates of the engines are equally high when they're brand new than when they're aged.

This is not an ideal result, as our understanding of the physical asset goes against this finding. Thus, our focus is now shifting to how to improve the results through different record keeping protocols.



Weibull Distribution Parameter Estimation

1 (*) (1)

Summary of Events and Censored Values

Sample Size	Failed	Censored (Def)	Censored (Temp)	% Censored
68	17	32	19	75.0

Summary of Estimated Parameters (based on ML method)

Parameter	Scale	Shape	Mean Life	Med. Life	Char. Life	Std. Dev.
Estimate	4003	0.9422	4113.11	2712.97	4003	4367.8
Std. Error	1115	0.1729	-	-	-	-

(-) Not applied

Hypothesis: shape parameter = 1 - not rejected,
based on Wald test observed value = 0.111806, p-value = 0.738097,
and 5% significance level.

Figure 3. EXAKT results on Weibull parameter estimation

First, we are interested in whether the covariates will have a stronger predictive value to the equipment age. It is possible that a condition-based maintenance model will work with the existing data, even if the age is not a statistically significant factor in the model. This can occur when the covariates reflect the age of the equipment, so age itself does not need to be in the model.

Second, we are interested in investigating the rationale in the way events are coded. It is possible that we will leverage our findings from the CEA data audit project to further inform the records in this project.

TTC: A continuation of re-inspection optimization for the TTC NDT team

Janet Lam

Introduction and background

For the past couple of meetings, we have been reporting on the optimality of re-inspection schedules for the TTC NDT inspection team. As a reminder, one of the responsibilities of the TTC's Non-destructive testing (NDT) team is to revisit known defects in the subway rail system according to a defined timetable. Depending on the severity, or priority of the defect, the schedule may be every 21 days, or annually, or something in between. When the defects are revisited, the NDT team notes the updated status, and this process is repeated until the defects are resolved.

As this re-inspection process is known to consume a significant portion of the NDT team's limited time, analysis into the defects was performed. This report is an extension of the project last reported in June 2020.

Summary of previous work

In the previous report, we discussed the model of transitioning from one priority to the next within a given re-inspection period. The results can be seen in the Table 1.

The key takeaways from the previous work is that based on the average number of defects carried each year, a slight increase in the inspection interval can save a significant number of inspections each year. The resulting decrease in reliability is relatively very small. For example, by changing the purple re-inspection interval from 17.4 days to 40 days increases the risk of missing a transition from 0.5% to 1.1%, and yields a saving of 57% in inspection efforts.

In order to bolster the results, TTC sent us more data that continue to the middle of 2020.

Table 1. Summary of results on transition probabilities

Priority	Re-inspection interval (days)	P(no transition)	Re-inspections / year	Savings (%)	Average # of defects per year	Expected saved inspections per year
Purple	17.4	0.995	21		129	
	21	0.994	17.4	17		464
	40	0.989	9	57		1548
	60	0.984	6.1	71		1922
Blue	36.42	0.991	10		33	
	45	0.989	8.1	19		62.7
	60	0.985	6.1	39		129
	80	0.981	4.6	54		178

Extension of work

With the new dataset, we have the benefit of a model that is already built, but the renewed costs of data cleansing. Resultantly, our most recent efforts have been focussed on preparing and cleansing the new dataset to work seamlessly with the existing dataset.

While the cleansing process is still underway, here are some procedures that have been performed.

Recoding failure modes

The work done on for the original results used failure modes that had been codified into a few categories. This was done again on the new data. A summary of the recoded results is given in Table 2.

It revealed that perhaps some of the failure modes might benefit from more granular coding, especially the Weld mode and the Misc mode.

Table 2. Recoded failure modes

Original failure mode	Coded failure mode
Rail: NDT Defect (Bolt Hole / cracked)	Bolt Hole Crack
Rail: NDT Defect (Bond (2/0,4/0,500MCM)/ web crack)	Bond Web Crack
Rail: NDT Defect (Bond (2/0,4/0,500MCM)/intern web crack)	Bond Web Crack
Rail: NDT Defect (Delta/Corrosion)	Corrosion
Rail: NDT Defect (Corrosion Alert)	Corrosion

Rail: NDT Defect (Corr./ underside of base)	Corrosion
Rail: NDT Defect (Corr./ notches at outer base)	Corrosion
Rail: NDT Defect (corroded-NDT check reqd)	Corrosion
Rail: NDT Defect (Corr./ rail head)	Corrosion
Rail: NDT Defect (Delta/Misc.)	Misc
Rail: NDT Defect (Delta/Bonds)	Misc
Rail: NDT Defect (MISC.)	Misc
Rail: NDT Defect (Squat/ downward cracking)	Misc
Rail: NDT Defect (Delta/Squat)	Misc
Rail: NDT Defect (heavily worn-NDT check reqd)	Misc
Rail: NDT Defect (broken-NDT check reqd)	Misc
Rail: NDT Defect (Engine Burn/ transverse cracking)	Misc
Rail: NDT Defect (Base/ cracked)	Misc
Rail: NDT Defect (Squat/ subsurface cracking)	Misc
Rail: NDT Defect (Squat/ transverse cracking)	Misc
Rail: NDT Defect (Rail Head/ damaged)	Misc
Rail: NDT Defect (Rail Web/ Horizontal Split Web)	Misc
Rail: NDT Defect (corrugation-NDT check reqd)	Misc
Rail: NDT Defect (Cracked Frog/ transverse crack)	Misc
Rail: NDT Defect (Rail Web/ Vertical Split Web)	Misc
Rail: NDT Defect (Rail Head/ RCF-head checks)	Misc
Rail: NDT Defect (Cracked Joint Bar)	Misc
Rail: NDT Defect (Engine Burn/ subsurface cracking)	Misc
Rail: NDT Defect (Rail Head/ RCF)	Misc
Rail: NDT Defect (Delta/THW)	Weld
Rail: NDT Defect (Delta/AWR)	Weld
Rail: NDT Defect (Delta/New Th.W. Collar Crack)	Weld
Rail: NDT Defect (Thermite Weld- internal crack)	Weld
Rail: NDT Defect (New T. W.)	Weld
Rail: NDT Defect (New T.W./Collar Crack)	Weld
Rail: NDT Defect (Thermite Weld- damaged head)	Weld
Rail: NDT Defect (New T. W./ porosity)	Weld
Rail: NDT Defect (A. W. Repair/ downward cracking)	Weld
Rail: NDT Defect (Thermite Weld- transverse defect)	Weld
Rail: NDT Defect (Thermite Weld- incomplete fusion)	Weld
Rail: NDT Defect (New T. W./ incomplete fusion)	Weld
Rail: NDT Defect (A.W. repair/ delaminating)	Weld

Rail: NDT Defect (Arc Weld Repair/ transverse cracking)	Weld
Rail: NDT Defect (New T. W./ slag inclusion)	Weld
Rail: NDT Defect (Thermite Weld- Porosity)	Weld
Rail: NDT Defect (Butt Weld/ cracked)	Weld

New defect ID numbers on old defects

TTC's record keeping system is such that when a defect transitions from a low priority defect to a high priority defect, a new ID number is generated. This means that the same physical defect may have two different identifying numbers in the database.

For continuity, we have manually re-labelled these "new" defects with their corresponding original ID number. When we received the new data, it was necessary to find the continuing records of these relabelled defects and match them with their previous records. Table 3 lists the defect numbers that had to be matched with their previous records.

Table 3. ID matching table

New defect ID	Matching ancestral ID
162425	157282
171170	157381
172056	165908
166858	166801
171233	170720

In the two years that have transpired since the previous dataset, there have been more defects that transitioned from low priority to high priority. It has not been possible to identify the ancestral IDs for these three defects. We are awaiting the new information for these defects.

171675	Delta has been upgraded to purple defect 2019.7 and changed to not found for tracibility
179110	Delta has been changed to not found for traceability. Has been updated to purple defect 2020.68.
191574	Delta has been changed to not found for traceability. Has been updated to purple defect 2020.147.

Missing data

The data for 2018 appear to only have records for new defects opened in 2018, and no records of defects that existed prior to 2018, and were updated throughout the year. We are awaiting the missing records.

Data Anomalies

There are several anomalies in the data that need to be addressed in advance of computing the results.

Repeating entries – In the dataset of 15000 entries, there are about 100 records of consecutive entries that are taken on the same date. It needs to be decided whether these should count as two separate visits to the defect, or treated as a single visit.

Entries that flip flop priorities – There are about 60 entries of defects that change priorities back and forth several times throughout its history. As our analysis is centered on the number of days it takes for a defect to transition from a lower priority to a higher priority, it is essential that we address this issue in a way that is most consistent with reality.

Summary of results

Though our project extension is still in the data cleansing stage, a summary of the data is as follows.

Average defect duration	460 days
Average number days to transition	
Blue to purple	429
Purple to yellow	148
Gray to Blue	374
Average number of visits to a defect	5.2

Next steps

The next steps of this project are crystal clear. We must simply proceed through the cleansing process, fill in the missing data and model the results as before.

Then, recommendations will be made in collaboration with the Subway Infrastructure team at TTC.

Appendix 1: Machine learning for process monitoring and control in additive manufacturing

Katie Xu

Introduction

Additive manufacturing, also known as 3D printing, is a fabrication process where 3D objects are built up layer-by-layer. There are a number of different additive manufacturing processes, characterized by the method of layer formation and the materials used. In fused deposition modelling (FDM), layers are formed by melting a thermoplastic material and depositing it in the desired locations where it solidifies and becomes part of the object. In laser powder bed fusion processes such as selective laser sintering (SLS) or selective laser melting (SLM), a layer of metal or thermoplastic powder is deposited and a laser is used selectively heat parts of the powder to fused them together. The final goal of this project is to develop a closed loop system to monitor and control the quality of parts made using a metal SLM machine, and an important intermediate milestone will be to implement the system for a FDM printer. This project is in collaboration with Professor Zou from the Material Science and Engineering department.

SLM and other additive manufacturing processes have important advantages for applications in industrial manufacturing. In particular, the ability to make objects with complex geometries which may be difficult or impossible to make with other processes, reduced waste material compared to traditional machining, and lower tooling costs compared to casting and moulding. These properties are advantageous for creating complex or custom parts that are needed in relatively small quantities. Unfortunately, there remain challenges which limit the current practicality of additive manufacturing, particularly metal SLM, in industry. Parameter search is time consuming because there is a large number of parameters to be found and it is generally done through trial and error. Moreover, optimal parameter values depend on factors such as the material being used and the part geometry, so in the worst case the parameter search process must be repeated for each unique part. Additionally, even with a good set of parameters, external disturbances can result in inconsistent part quality and high rejection rates [1].

Thus, there is a need for automated process monitoring and parameter adjustment to improve the quality and consistency of parts made using SLM. This problem can be broken down into two parts: process monitoring and process control. Process monitoring refers to the use of process signatures such as images or temperature measurements to predict quality metrics of the final part. Process control refers to the use the predicted quality indictors to take corrective action by adjusting process parameters. Machine learning is a strong contender for these tasks, which are difficult to model analytically

Related work

This section will outline some related work in the literature on process monitoring for metal SLM printers. There exists a wide variety of work using different combinations of sensor configurations, target quality predictor, and mapping techniques. Thus far for this project, I have focused on reviewing works which use machine learning-based mapping techniques, with image data as the input. Broadly, these include classifiers with fixed feature extraction, convolutional neural networks (CNNs), deep belief networks, and direct image analysis.

Traditional machine learning classifiers in combination with a feature extraction scheme have been used to classify defects. Aminzadeh and Kurfess [2] used Bayesian inference on features extracted from top-down images of build layers to classify build quality. Zhang *et al.* [3] used a support vector machine (SVM) to classify build quality, after extracting features from images of several regions of interest taken continuously during the build process. In the same paper, these authors also used a convolutional neural network (CNN) for the same task. They found that the CNN was more effective and had the added advantage of not requiring explicit feature extraction. Other authors have also used CNNs to predict properties of the process and final product. Kwon *et al.* [4] used a CNN with a regression model to predict laser power based on images of the melt pool. Yang *et al.* [5] and Zhang *et al.* [6] also use images of the melt pool as the input. Yang *et al.* were interested in detecting irregularities in melt pool size, and Zhang *et al.* were interested in predicting the porosity of the final product. Both approached these as classification problems.

Ye *et al.* [7] sought to address some drawbacks of CNNs. They classified the melt state of the metal using images of the plume and splatter regions of interest. By using deep belief networks, they reduced the need for hyperparameter tuning compared to using CNNs while achieving comparable results. Finally, Yao *et al.* [8] used multifractal analysis on top-down images of entire build layers to extract a statistic representing the quality of the layer. This approach directly analysed the image and was *not* data-driven. The authors have applied this method in a sequential optimization framework to determine the optimal corrective action after each layer [9], and it is the only method listed here which has been applied to some form of process control.

Future work

The literature review thus far has found that process monitoring in SLM has been fairly widely studied. However, many authors cite process control as an end goal of process monitoring but few appear to have applied their methods to this problem. Moving forward, efforts will focus on studying process control methods used in SLM and other additive manufacturing systems, while working with collaborators to establish data requirements and availability.

References

[1] J. Fox, F. Lopez, B. Lane, H. Yeung, and S. Grantham, “On the requirements for model-based thermal control of melt pool geometry in laser powder bed fusion additive manufacturing,” in *Proceedings of the 2016 Material Science & Technology Conference*, 10 2016. [Online]. Available: https://www.nist.gov/publications/requirements-model-based-thermal-control-melt-pool-geometry-laser-powder-bed-fusion

[2] M. Aminzadeh and T. R. Kurfess, “Online quality inspection using Bayesian classification in powder-bed additive manufacturing from high-resolution visual camera images,” *Journal of Intelligent Manufacturing*, vol. 30, no. 6, p. 2505–2523, 2018.
[3] Y. Zhang, G. S. Hong, D. Ye, K. Zhu, and J. Y. Fuh, “Extraction and evaluation of melt pool, plume and spatter information for powder-bed fusion am process monitoring,” *Materials Design*, vol. 156, pp. 458 – 469, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S026412751830532X
[4] O. Kwon, H. G. Kim, M. J. Ham, W. Kim, G.-H. Kim, J.-H. Cho, N. I. Kim, and K. Kim, “A deep neural network for classification of melt-pool images in metal additive manufacturing,” *Journal of Intelligent Manufacturing*, vol. 31, no. 2, p. 375–386, 2018.
[5] Z. Yang, Y. Lu, H. Yeung, and S. Krishnamurty, “Investigation of deep learning for real-time melt pool classification in additive manufacturing,” in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 2019, pp. 640–647.
[6] B. Zhang, S. Liu, and Y. C. Shin, “In-process monitoring of porosity during laser additive manufacturing process,” Additive Manufacturing, vol. 28, pp. 497 – 505, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214860419303653
[7] D. Ye, J. Y. Hsi Fuh, Y. Zhang, G. S. Hong, and K. Zhu, “In situ monitoring of selective laser melting using plume and spatter signatures by deep belief networks,” *ISA Transactions*, vol. 81, pp. 96 – 104, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0019057818302763
[8] B. Yao, F. Imani, A. S. Sakpal, E. W. Reutzel, and H. Yang, “Multifractal Analysis of Image Profiles for the Characterization and Detection of Defects in Additive Manufacturing,” *Journal of Manufacturing Science and Engineering*, vol. 140, no. 3, 01 2018, 031014. [Online]. Available: https://doi.org/10.1115/1.4037891
[9] B. Yao, F. Imani, and H. Yang, “Markov decision process for image-guided additive manufacturing,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2792–2798, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8362941

Appendix 2: NRC combustion signature analysis for freight fires

Sophie Tian

Introduction

The Fire Safety Unit within the National Research Council Canada (NRC) is responsible for the research on increasing resilience to outdoor and infrastructure fires, developing fire safety technology and applying computational technologies in fire safety. Currently, in response to freight transportation fire incidents, first responders survey the scene of the accident to identify potential hazardous materials present, and follows the Emergency Response Guidebook (ERG), which is a 400-page document, to determine the most appropriate mitigation strategy. Since early identification of the scope of the emergency is crucial to protect people, property and the environment from the effects of the fire, and also since first responders routinely confront unknown hazards that might explode or release toxins in response to ordinary fire extinguishers, the Fire Safety Unit in NRC is interested in developing an Artificial Intelligence (AI) enabled tool to provide decision making support. In particular, this tool will assist first responders when they survey the scene of the accident by identifying what materials are burning and determining the most the most appropriate protective gear, and the most effective mitigation measure in response to freight fires.

The proposed tool to be developed will consist of several components. First, it will consist of a chemical sensor to be designed by NRC that gathers real-time information in the case of a freight fire. The proposed tool will also contain a combustion signature analysis (CSA) unit, which will employ machine learning (ML) algorithms to predict what is burning based on combustion signatures collected from the designed chemical sensor. Finally, this tool will also include a decision-making module that uses the predictions from the trained model along with information from the ERG document for first responders to produce actionable intelligence to the first responders, effectively providing guidance in the appropriate mitigation strategies to take in a freight fire.

Problem specification: combustion signature analysis unit

This project is in its early stages of development; its focus will be on the design of the ML models for combustion signature analysis and on how to resolve the issue of applying these models to combustion signatures collected from a new, not-yet-designed chemical sensor specifically for usage in freight fires.

Combustion gas signatures reveal what type of material is burning and how the combustion progresses. NRC has tested a number of materials and analyzed combustion gas signatures using existing chemical sensors for many years, and as a result, has accumulated data in various forms. Using existing combustion signature data collected in laboratory settings, eligible ML models are to be developed to predict the classification of original materials. Since the NRC has collected combustion signature data in different forms and from different types of tests, we will analyze the performance of ML models trained using each type of input data and determine the most effective form of combustion signature data to use.

Then, further analysis will be conducted on the classifiers built to identify the most important dimensions of the input data that characterize the classification decisions made by the ML model. This dimensionality reduction step is important since chemical sensors used in laboratory settings produce high-dimensional data, but are less portable, whereas the new sensor to be developed for freight fires will need to be portable and easily carried by first responders as they survey the scene of the accident. The identification of such characterizing dimensions will be crucial in informing NRC the requirements of the new chemical sensor to be developed, and ideally, we hope that the classification decisions can be made with an acceptable level of confidence and accuracy using only 3 or 4 dimensions in the chemical sensor data.

Since the new sensor designed will produce different combustion signatures from the existing data that the ML models are trained on, we also need to find a mapping between these two types of data. This mapping will enable the ML model to be employed with the data collected from the new chemical sensor, and consequently allow classification decisions to be made at the scene of the freight fire.

Anticipated outcomes

The proposed tool addresses the safety of freight transportation by providing a solution that can be deployed easily to freight fires and by allowing data-driven decisions to be made given real-time information. It reduces the risks to first responders by identifying possible hazardous materials present based on combustion signatures collected at the scene and by providing decision-making support to first responders in determining the most effective mitigation measures.

Conclusions and future work

This report introduces the combustion signature analysis project in collaboration with the Fire Safety Unit of NRC, including the background and objectives of the project, and a detailed description of the problem specification. Future work includes developing ML models to classify the combustion signature data, exploring the effectiveness of the different forms of input data, and exploring domain adaptation techniques to allow the selected ML model to be used with data from the new chemical sensor.

Appendix 3: Generic real options valuation

Ruiqi Yang

Model Definition

This project is an extension of the UKMOD, and in the original UKMOD project, we need to determine an optimized fighter jet procurement policy for the client, the United Kingdom Ministry of Defence. We assume that the price of the fighter jet follows Merton's Jump Diffusion process [14]. Furthermore, there exists only the negative price jump (price increment after jump happens will be negative) due to the technology innovation. We could construct such a Jump Diffusion process by carefully choosing the model parameters. Moreover, when we take a close look at the Real Option, we found two properties for such assets, namely high market impact and risk-sensitive. The competitors could counter the new fighter jet, and the cumulative future rewards should immediately decrease when they do so. This gives extra risk when the buyer considers long the option. Because we are aiming to define a model for pricing such an option which will give buyer options to purchase the fighter jet in the future, this means we should specify the possible future market impact and penalize the long position of the option for the market risk.

To make the Real Options Valuation problem more realistic, we further introduce some of the Swing Options' constraints. The budget constraint determines how much the buyer could spend in before the maturity of the Real Option, the budget is assumed to accumulate compounded interests; the local constraint determines how many assets the buyer could purchase each time, notice that zero is always a choice; the global constraint determines the minimum procurement size the buyer has to achieve before maturity to leave without penalty, and also the maximum procurement size.

Instead of defining the Real Options valuation model directly, inspired by Halperin and Feldshteyn (2018) [1], we choose to construct a replicating portfolio and analyse the value increment for the portfolio in discrete time step. There are three main reasons why we do so. First, this is similar to a vanilla option's definition. Second, this is related to the dynamic programming. And third, the only assumption we need to further introduce is that the buyer and the seller of the option holds same amount of information, and we could use the Merton's Jump Diffusion for the price as the only dynamic for the model.

We assume the state space at time t is $s_t = P_t \times A_t \times B_t \times (T-t)$. $P_t \in \mathbb{R}^+$ is the price of the underlying asset at time t; $A_t \in [0, A_{max}]$ is the number of assets the buyer purchased before time t; $B_t \in [0, B_0(1+r_f)^t]$ is the budget remaining before purchase at time t, where r_f is the risk free interest rate; $T - t \in [0, T]$ is the number of decision times left, we could use t and they are equivalent; we insist to put this in the state space since time is import in the Real Options Valuation and this is showed with a simple example in appendix. Notice that if we further assume that $a_t \in [0, a_{max}]$ is the current decision, we can immediately get the constraints as follows:

- Budget constraint: $(B_t - a_t P_t)(1 + r_f) = B_{t+1} \geq 0$.
- Local constraint: $0 \leq a_t \leq a_{max}$ and $a_t \in \mathbb{Z}$
- Global constraint: $A_t + a_t = A_{t+1} \leq A_{max}$
-

The penalty p introduced by the Swing Option will be applied to buyer at the maturity for each asset that A_T is less than A_{min}. It is equivalent to say that $-p \cdot \max(A_T - A_{min}, 0)$ is the terminal reward.

For the reason that we discussed earlier, we need to access the risk and market impact in the model, we will have three parts in one step reward. They are $R^{(0)}$ for the increment of portfolio, as the main part of the reward; R^{mkt} for retrieving the market impact; and R^{risk} for penalizing the risk.

Main reward

After discretizing the time, we define $u_t = a_t P_t$ as the trading value at the beginning of the interval t, so that assets value x_t^+ immediately after trades are deterministic:

$$x_t^+ = x_t + u_t$$

Since the A_t assets that is already owned worth P_t each at time t, the total portfolio's dollar value is then

$$\Pi_t = A_t P_t + b_t = x_t + b_t$$

And the post-trade portfolio value is therefore

$$\Pi_t^+ = x_t^+ + b_t^+ = x_t^+ + b_t - u_t$$

Due to the existence of the budget, the purchase is financed from the bank cash account. This imposes the following self-financing constraint:

$$u_t + b_t^+ - b_t = 0$$

In post-trade portfolio Π_t^+, assets x_t^+ and cash are accumulating different returns till the end of the current period. The cash is invested in the money market account with risk-free rate r_f and the rate of return for assets is r_t for a non-negative trading a_t. The end-period portfolio value is then obtained as follows:

$$\Pi_t^{end} = (1 + r_t)(x_t + u_t) + (b_t - u_t)(1 + r_f)$$

Given the A_t asset still generate value in the current period, we assume the rate of return for the zero a_t is r_t^0 and in this case $r_t = r_t^0$. We can obtain the portfolio's dollar value increment in period t as:

$$\begin{aligned}\Pi_t &= (1+r_t)(x_t+u_t) + \left(1+r_f\right)(b_t-u_t) - (1+r_t^0)x_t - \left(1+r_f\right)b_t + r_t^0 x_t \\ &= -r_f u_t + r_t(x_t+u_t) \\ &= -r_f a_t P_t + r_t(a_t+A_t)P_t\end{aligned}$$

One thing should be noticed immediately, the $a_t + A_t$ assets on hand will worth $(a_t + A_t)P_{t+1} = A_{t+1}P_{t+1}$ at the beginning of the next period $t+1$, and this relationship reflects the market dynamics.

Market impact and risk penalty

At each period, we will face a random negative market impact, and we assume the dollar value of the impact is linear to the dollar value of the total assets that the buyer holds at time t as follows:

$$-(c_1(a_t+P_t) + c_2 t + \epsilon)(A_t+a_t)P_t$$

where $\mathbb{E}(\epsilon) = 0$ and $Var(\epsilon) = \sigma_M^2$, and ϵ is high enough such that we should penalize the long position of the option for taking such risk, this will yield the R_t^{mkt} and R_t^{risk} as follows:

$$\begin{aligned}R^{mrk} &= -(c_1(a_t+A_t) + c_2 t)(A_t+a_t)P_t \\ R^{risk} &= -\lambda\sigma_M^2(a_t+A_t)^2P_t^2\end{aligned}$$

which are the expected values and portion (λ is the risk-aversion) of the variance of the equation at the beginning of this subsection.

Comments

We can see that the reward at current period t is a quadratic function with respect to the current decision a_t, and this is designed to be so. There are two reasons why we do this. The first one is that the market impact and market risk are important. The second reason is that, obviously, if it's a linear function with respect to the current decision a_t, then the whole option value will also be a linear function with respect to every decision; this will yield the optimal decision tends to be "buy all at the beginning" with a high probability for all trajectories, since by the nature of the dynamic, the price tends to drop in the future.

Moreover, when we try to solve our model, we first need to decide what is the rate of return r_t. In the following section, when we solve the model by Deep Reinforcement Learning, the rate of return could be anything that makes sense. For example, a linear function, a piecewise linear function, a Mean Reverting process, or even another Merton's Jump Diffusion process. For simplicity, we will use a piecewise linear function in our experiments in the following section.

Integer programming model

When we combine all three parts of reward together, along with the terminal reward introduced by the Swing Option assumption, we will have an Integer Programming problem when the market dynamic is known, as follows:

$$\max_{a_t\ \forall t} \qquad \sum_{t=0}^{T-1}\left(1+r_f\right)^{-t}\left(R_t^{(0)}(a_t) + R_t^{mkt}(a_t) + R_t^{risk}(a_t)\right) + \left(1+r_f\right)^{-T}\max(A_{min} - A_T, 0)$$

$$s.t. \quad (b_t - a_t P_t)(1 + r_f) = B_{t+1}$$
$$B_t \geq 0$$
$$A_t \leq A_{max}$$
$$A_t \leq A_{max}$$
$$a_t \in \mathbb{Z}$$
$$\forall t$$
$$\forall t$$
$$\forall t$$
$$\forall t$$
$$\forall t$$

By using Integer Programming, the optimal policy for a single future is knowable. Since the asset's dynamic is the only dynamic in our model, a single future corresponding to an arbitrary future price trajectory, which is starting at the same initial price P_0. Readers might suspect how this would help since the future is unknown, there are thousands of possible futures and many of them have a large difference. However, if we combine the previous discovery with simulation, we could simulate abundant price trajectories and estimate the mean and standard deviation for the optimal policy, and we could compare them with the one found by the trading agent that we will discuss later. The Integer Programming provides a convenience method and could be used to determine how good a trading agent is.

Deep reinforcement learning

We could solve our model by Backward Induction. Amin (1993) [2] introduced a methodology that is analogous to the CRR model to discretize the MJD process, P_t in our case, with guaranteed week convergence to its continuous-time formulation. However, if we tend to use Backward Induction, three problems could be considered. First, P_t is only one of the dimensions in the state under our setting, and we still have the budget B_t as a continuous value; Backward Induction needs a discrete state, it's not clear how should we discretize the budget. Second, even we discretize the budget properly, the weekly convergence property is not guaranteed for the option value. Third, just like other MDP problem, curse of dimensionality exists in our problem; for example, if we assume the maximum budget is 2000 dollars and discretize it to the unit dollar, and assume the global constraint is 25, the state space is around 270 billion for the weekly decision problem.

Inspired by Halperin, I. (2017) [3], we solve this problem by combining simulation and deep reinforcement learning, and there are several advantages. First, we could compare the agent's policy with the optimal policy we discussed in the previous section. Second, we could put more useful information in our state without warning too much about the curse of dimensionality; for example, when we use a vanilla Neural Network as function approximation, we could use the exponential weighted average of the historical price as a part of the state to describe the path of the price trajectory. Third, we could take the advantage of different Neural Network architectures, LSTM or GRU for example.

Methodology

The algorithm we focus on is *Rainbow* [11], which is the combination of several different algorithms and can be seen as an upgraded version of Deep Q-Network. The components of *Rainbow* are:

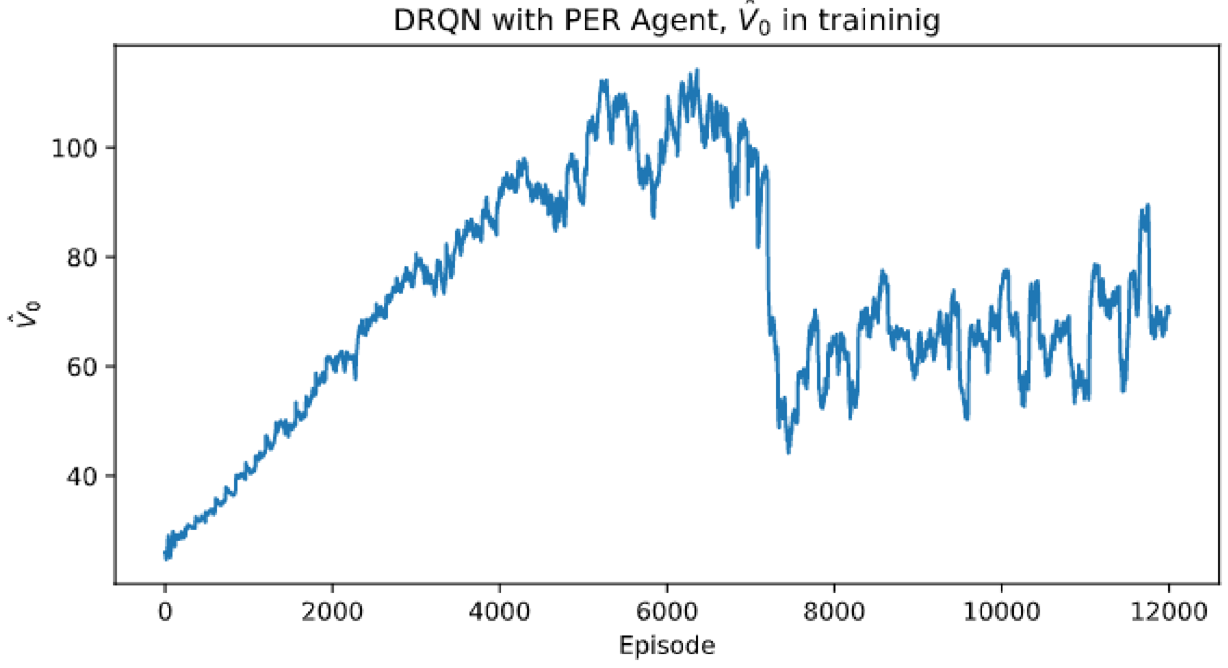
1. *Deep Q-Network (DQN)* [4]

2. *Double DQN (DDQN)* [5]
3. *Prioritized Experience Replay (PER)* [6]
4. *Dueling Network Architecture* [7]
5. *Multi-step bootstrapping* [8]
6. *Distributional Q Learning* [9]
7. *Noisy DQN* [10]

I tried each component of the *Rainbow*, and each individual algorithm added to the *DQN* will cause a longer training time. I also tried the LSTM layer instead of a vanilla Neural Network, this is motivated by D. L. Clare Chen et al. (2016) [12]. All these algorithms are training in a testbed case where we could make 60 decisions in the next 10 year, the maximum total assets we could purchase in this option are 25 (global constraint), the maximum number of assets we could purchase each time is 7 (local constraint), and we have an initial budget of 1800 (budget constraint). The best agent I could find is the one training with *Double Deep Recurrent Q Network* (DDRQN) with *Prioritized Experience Replay* (PER).

Next, we will discuss the big picture of the experiment. We sperate the whole experiment into two steps, training and testing. Notice that no matter what will happen in the future, we have the same initial state s_0, and this is important in the training step.

In training, each episode is corresponding to a new price trajectory, and we will perform the training with different experiences since Q learning is an off-policy algorithm. For example, if we are using the *DDQN* with *PER* and LSTM layers, in each time step, we will record the current state, the action predicted by the agent (Neural Network), the reward, and the next state, along with the Temporal difference error. We will then take a random minibatch of experiences and apply the gradient descent accordingly. After the completeness of each episode, we will feed the initial state s_0 into the Neural Network and take its maximum value, that is $\hat{V}_0 = \max_{a_0} Q(s_0, \cdot)$, and we treat $\hat{V}_0$ as the approximation of true option value and train until this value converge. The learning curve of the *DDQN* with *PER* is as follows:



In the testing step, we will take 1000 different price trajectories, and compare the agent we trained with the optimal policy we found using Integer Programming. The way we use our trained agent is pretty simple, and it is the same as the training step except we need to set the additional penalty (which is the penalty in the following subsection) as zero. For each price trajectory i, at each time step t, we feed the current state s_t into the Neural Network, the NN will give the output as $Q_t^{(i)}\left(s_t^{(i)},\cdot\right)$; we will use the action which yield the maximum Q-value as the optimal action in current period, that is $a_t^{(i)} = \arg\max_{a_t^{(i)}} Q_t^{(i)}\left(s_t^{(i)},\cdot\right)$, and feed into our environment. The reason why we could use the *argmax* of Q value directly is that the corresponding position of the $Q(s,\cdot)$'s output is the same as the action.

Constraints disposal

The traditional Q-Learning with function approximation estimates the Q value for state-action pair, $Q(s,a)$, instead of transforming state-action pair into a feature vector, the Deep-Q Network will have a separate output corresponding to each action. In our problem, for example, if the maximum number of assets could buy in each decision epoch is 7, we will have eight outputs for Deep-Q Network, and this is because that buying 0 assets will always be a valid action. A Deep Q-Network always requires a fixed output dimension.

However, since the existence of the budget constraint and the global constraints, we might have infeasible actions at each time of making decisions. To solve this problem, we introduce an additional penalty p_{-}, and it will apply each time whenever the agent chooses an infeasible action, and the action will be re-chosen with one less. Finally, the reward for taking the original action will be the penalty-adjusted one. Notice that the penalty p_{-} becomes a hyperparameter.

```
Constraints Disposal Algorithm
In the environment, observe current price $P_t$,
budget $B_t$, and assets on hand $A_t$
The agent $Q(s_t,\cdot)$ predict the optimal feasible
action $a_t^* \in [0, a_{max}]$, and the reward for taking
this action $r_t^*$. With the penalty $p_{-}$, do following:
set $a = a_t^*$
while $a \geq 0$
   if $A_t + a \geq A_{max}$ or $B_t - P_t * a < 0$:
      $a = a - 1$
      $r_t^* = r_t^* - p_{-}$
   end if
end while
Return the predicted action $a_t^*$ along with its
reward $r_t^*$
```

The reason why applies the penalty recursively is pretty simple, an infeasible action happens only when the agent's predicted an action is above the maximum feasible action. Given that the action space is discrete non-negative value, we want to punish the agent (a Neural Network) with the size between the action it performs and the maximum feasible action at the current state.

We will set the extra penalty p_{-} as a large positive number in the training step, and it will be zero in the test step. However, due to the nature of this simple algorithm, we have to introduce an additional value that should be controlled in the learning, which is the error the agent makes.

Numerical results

Due to the high computational cost for an individual algorithm in *rainbow* applies to our problem, I don't have a well-tuned agent for every algorithm. As I mentioned previously, the best agent I have is using DDRQN with PER, reader can find details of this algorithm in appendix with the extra conditions for using Dueling network architecture in the appendix. The following table presents all agents I have; all results are computed in the testing step.

algorithm	portfolio mean	portfolio std	mean error	std error
optimal (IP)	98.67	52.56	-	-
DDQN	53.03	60.01	-	-
DRQN	73.81	57.71	7.6	17.7
DRQN with PER	77.4	58.74	3.4	8.2
D3RQN with PER	76	53.43	9.23	13.75

As we can see from the above, the biggest improvement for the model comes from the LSTM layer. And the idea being that the RNN (LSTM in our case) will be able to retain information from states further back in time and incorporate that into predicting better Q values and thus performing better on games that require long term planning. One technical detail is that we choose to initialize the LSTM layers' hidden states as zero.

Also, there's a tiny improvement that comes from the Prioritized Experience Replay. The reason why I put so much attention on the PER is that given the dynamic of our model is an MJD with a high chance that the price would have a negative jump in the future, and when a jump happens, this experience might "surprise" the agent. The PER will put more weight on those unexpected experiences. I expect the PER has the functionality that captures the possible jumps and predicts an accurate Q value when the jump happens.

Since the Neural Network is a universal function approximation, the Dueling Network architecture gives a separate layer from predicting the action value, and this will yield a more complex NN. I hope this separate action value layer could further reduce the error that the NN makes, this is failed according to the table. However, during the testing step of the D3RQN algorithm, I did observe the trend of the error that the agent makes is going down, which indicate I might need to train this agent longer.

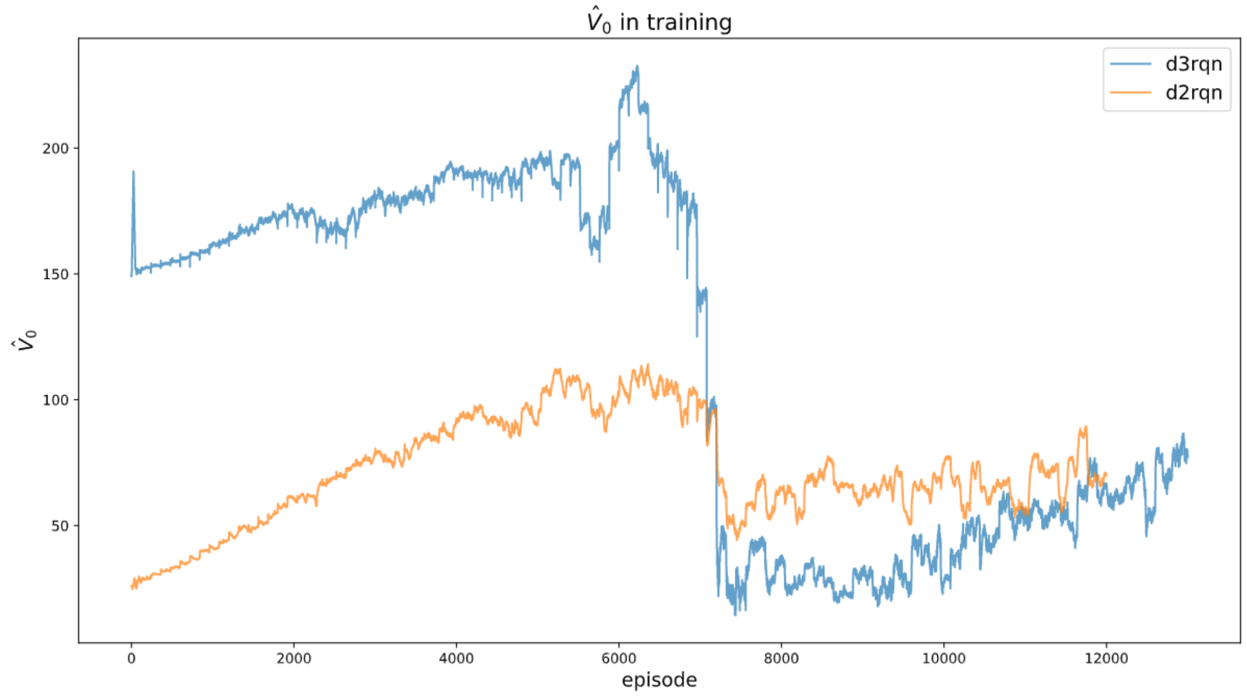
Furthermore, I think the Distributional Q learning would be helpful for our problem. Given a state s_t, this method predicts a distribution of returns for each action according to the points that the user fixed before the training. Given that all three parts of the reward are related to the price, and the log-ratio between future price and current price follows a normal distribution, I do think the distribution of returns is more proper for our problem. Amin [2] gives a result that, for a MJD, the increment between current price and the next period price is fixed, and with different probabilities that depending on the current price. It requires further research for using above result.

Learning curve experiment

We can notice that the learning curve presented in the previous section is different from a typical learning curve, where the Q value would monotonically increase in general. In our learning curve, there's a huge jump around 7500 episodes. The main topic in this section is to present a learning curve analysis and try to explain this phenomenon.

Before the experiment begins, let's first remember ourselves that we introduced an error (to make an infeasible action) that we want to control and reduce in the training step, and this error implies how much penalties p_{-} are applied. Since the penalty is directly applied to the reward, control the error will be an alternative choice to gain a larger reward in the training step. Therefore, we could achieve a larger state-action value if we control the error.

Next, we will discuss the experiment itself. We will use the Dueling Double Deep Recurrent Q Network (D3RQN) with PER to perform the experiment. Since we added the Dueling Network into the original NN, we trained 1000 more episodes. The learning curve of D3RQN with PER is as following, and I also present the DDRQN with PER learning curve for comparison purposes.



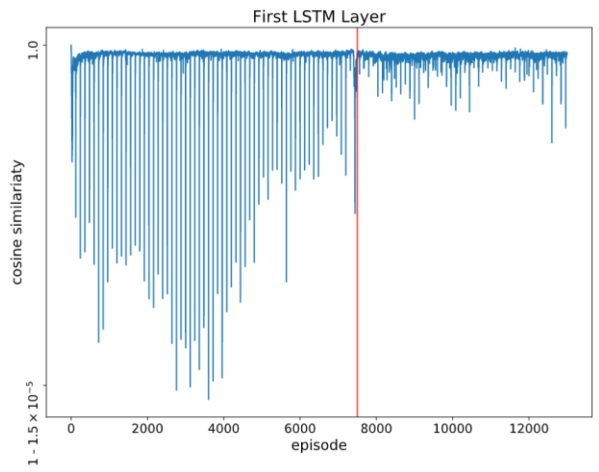
As we can see, both have a jump at around 7500 episodes, but with different jump size. And after the jump, the $\hat{V}_0$ keeps increasing and tends to be stabilized.

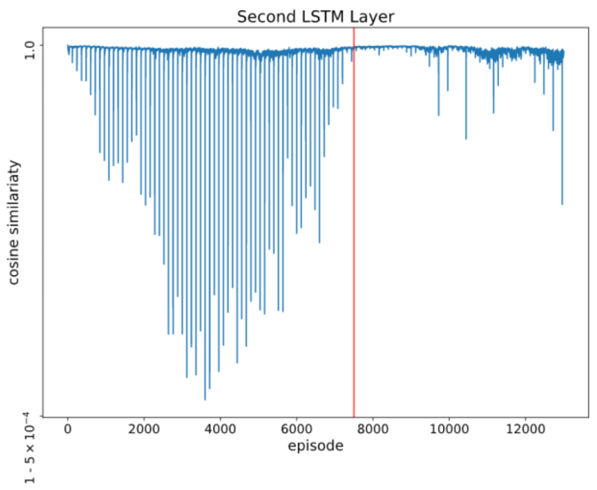
The hypothesis is that the LSTM layers are updated before 7500 episodes, during this period, the whole network also learns how to reduce the error; after 7500 episodes, the LSTM layers are stabilized, and the rest of the network has to re-learn and control the error. Furthermore, the sudden drop is not related to overfitting.

LSTM layers analysis in training

The agent we trained with D3RQN contains two LSTM layers; one returns a sequence, and one returns a single vector; follows by the *Dueling* fully connected layers. Motivated by the blog *DQN and DRQN in partially observation gridworlds* [13], we can observe a clear pattern in the LSTM hidden state updates when training a *DRQN* agent, this suggests we should take a look at our LSTM hidden states. When training, we store the hidden state return by both of the LSTM layers in replay buffer as experience, we measure the mean (with respect to the minibatch) cosine similarity between the hidden state before and after the training. And then we replace the hidden state stored in the replay buffer for the next comparison (if it exists).

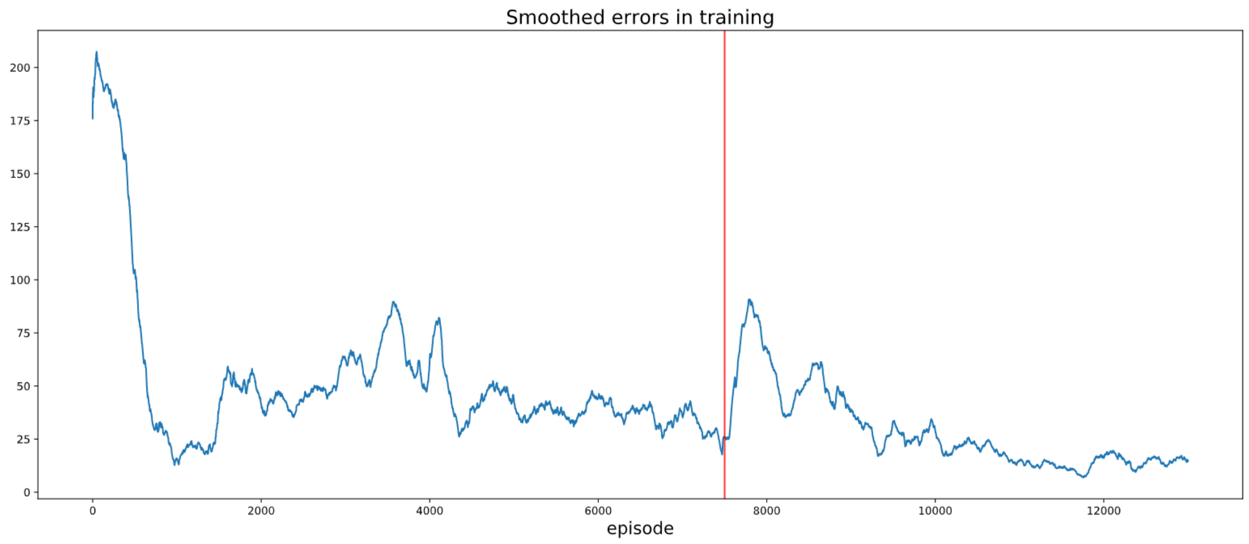
The mean (with respect to the time steps in one episode) cosine similarity of the hidden states is in the following, I use the red line to indicate where the 7500 episode is. First, we should notice that the mean cosine similarity is small, which suggests we should pay attention to the size of it but not the value itself. Second, we can observe that the majority of the hidden states update happen before 7500 episodes. This is actually the same as the first part of the hypothesis.





Error analysis in training

At each time step in each episode, I store the error made by the agent and measure the mean error made by the agent in one episode. The mean error made by the agent in the training step is at the following, I also use the red line to indicate where the 7500 is.

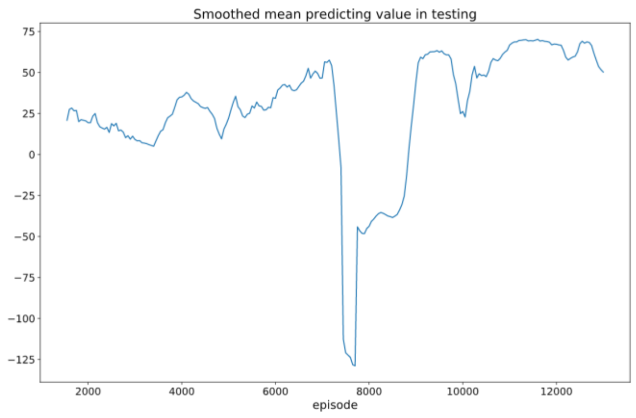


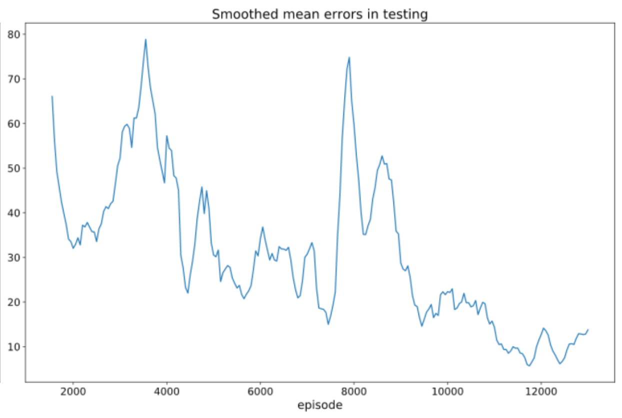
As we can observe, there is a clear trend for the mean error between 4000 to 7500 episodes. We would expect a lower mean error if the trend was continuing after 7500 episodes, however, there is a raise of the mean error at 7500 episodes where the majority of hidden states update finished. This is the second part of the hypothesis, after the hidden states update is finished, the rest of the neural network has to relearn to control the error it makes.

Portfolio value and mean error in testing

When training, after a fixed number of episodes, we store the weight learned by the Neural Network. This process is been performed after 1400 episodes since we use the epsilon greedy for exploring and it ends approximately at this time.

In the testing step, we use the weighted stored in training step to act as the trading agent, and we store the portfolio value and mean error as the agent's performance. We plot the smoothed mean predictions and mean errors.





From the above plot, we can see that some of the agents before 7500 episodes are doing pretty well. However, when we take a closer look at the smoothed mean error, we can see that even some of the agent before 7500 episodes has a high predicting value; they also have a high mean error. The agent after 7500 episodes can do better in predicting the portfolio value with a lower error make. This actually concludes our hypothesis, and the drop in the learning curve is not overfitting.

Appendix

Time in state: a simple example

In this subsection, I will introduce the reason why we think our problem is time inhomogeneous. Let's consider a simple example where we have 10 decision times in 10 years horizon. Under the swing option framework, let's assume that the maximum number of assets we can buy per transaction is 5, the minimum number of global constraints is 15 (this is also the number of assets we need to buy before maturity to exist without penalty), and the maximum number of global constraints is 20. Assume the market dynamic gives the following price trajectory in the next ten years (A Merton's Jump Diffusion process):

time	0	1	2	3	4	5	6	7	8	9	10
price	100	105	95	50	55	60	65	75	60	50	55

Furthermore, assume that the penalty for the swing option framework is 60 per asset shortage. Assume we had bought 10 before time 3, and we did not buy any during time 4 to time 8. If the time (or equivalently, number of decision times left) does not include in the state, then at time 3 and time 9, we are basically in the same state; that is we own 10 assets, and the price is 50. However, when we make decisions, we should make different decisions in these two cases:

- At *time 3*, the jump happens, we still have time for waiting.
- At *time 9*, the price is low, but we're facing the situation where we have to buy some assets; otherwise we will lose more.

To conclude, we include time in our state since time is sufficient relevant information when making a decision. For the same reason, we also include the number of assets we hold till decision time in the state as well as the exponential weighted average price. The price at time 5 and time 9 are both 50. However, a single value 50 is unable to describe the general price movement.

D3RQN with PER

```
D3RQN with PER
Initialize replay memory $\mathcal{D}$ to capacity $N$,minibatch size $k$, exponent $\alpha$ and $\beta$, small constant $\epsilon$
Initialize action-value function $Q(\cdot\ |\boldsymbol{\theta_M})$ with random weights. Pre-train $Q(\cdot\ |\boldsymbol{\theta_M})$ and make a copy $Q(\cdot\ |\boldsymbol{\theta_T})$ ---------------------------------------------------------------------------------- [4]
for episode = 1, $M$ do
    Initialize sequence $s_0$ and preprocessed sequenced $\phi_0 = \phi(s_0)$ ---------------------- [1]
    for $t$ = 0, $T$ do
        With probability $\epsilon$ select a random action $a_t$, otherwise select $a_t = \max_a Q^*(\phi(s_t),\ a_t; \boldsymbol{\theta_M})$ ------------------------------------------------------------ [2]
        Execute action $a_t$ in emulator and observe reward $r_t$ and next state $s_{t+1}$. Set $s_t = s_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$. Store transition $(\phi_t,\ a_t,\ r_t,\ \phi_{t+1})$ in $\mathcal{D}$
        Set $\Delta\ = 0$
        for j = 1 to k do
            sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
            Compute IS weight $w_j = \left(N \cdot P(j)\right)^{-\beta} / \max_i w_i$
            Compute TD-error $\delta_j \leftarrow r_{t_j}^j + [Q\left(\phi_{t_j+1}^j, argmax_a Q\left(\phi_{t_j+1}^j \middle| \boldsymbol{\theta_M}\right) \middle| \boldsymbol{\theta_T}\right) - Q(\phi_j, a_j | \boldsymbol{\theta_M})]$
            Update transition priority $p_j\ = |\delta_j|$
            Accumulate weight-change $\Delta += w_j \cdot \delta_j \cdot \nabla_{\boldsymbol{\theta_M}} Q\left(\phi_{t_j}^j, a_{t_j}^j\right)$ ------------ [3]
        end for
        update weights $\boldsymbol{\theta_M}$
    end for
    Decay $\epsilon = \tau\epsilon$
    After $\rho$ iterations update $\theta_M = \theta_T$
end for
```

The whole algorithm is based on Double Deep Q-Network, in which we use the target network for prediction next state's Q value, use the main network to select an action, and perform the gradient descent step. A few things should be noticed here:

[1] The ϕ step corresponds to the feature engineering. As our state is a mixture of continuous and discrete values that differ in scale, we may want to perform normalization, standardization, or one hot encoding before we pass the state into the neural network.

[2] The ϵ exists since we are using the ϵ-greedy for exploration. There is an alternative method in *Rainbow* for exploration called *NoisyNet*, in which we use a noisy fully connective layer as the final layer in the neural network.

[3] Depending on the loss function, we may have different $\nabla_{\boldsymbol{\theta_M}} Q\left(\phi_{t_j}^{j}, a_{t_j}^{j}\right)$. For example, if we are using mean squared error, then $\nabla_{\boldsymbol{\theta_M}} Q\left(\phi_{t_j}^{j}, a_{t_j}^{j}\right) = \nabla_{\boldsymbol{\theta_M}}\left(y_j - Q\left(\phi_{t_j}^{j}, a_{t_j}^{j}\right)\right)^2$, where $y_j = r_{t_j} + \gamma \cdot \max_{a'} Q\left(\phi_{t_j+1}^{j}, a'; \theta_T\right) \cdot 1_{t_j+1 \neq T}$.

[4] Since we are using the Dueling Network, we need to define value $V(s; \theta, \alpha)$ and advantage $A(a; \theta, \beta)$ layers after the LSTM layers. We could then define the output of the neural network to be either $Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \alpha) + \left[A(a; \theta, \beta) - \max_{\tilde{a}} A(\tilde{a}; \theta, \beta)\right]$ or $Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \alpha) + \left[A(a; \theta, \beta) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(a'; \theta, \beta)\right]$.

References

[1] Halperin, I., & Feldshteyn, I. (2018). Market Self-Learning of Signals, Impact and Optimal Trading: Invisible Hand Inference with Free Energy. ArXiv, abs/1805.06126.

[2] Amin, K. I. (1993). Jump Diffusion Option Valuation in Discrete Time. The Journal of Finance, 48(5), 1833–1863. doi: 10.1111/j.1540-6261.1993.tb05130.x

[3] Halperin, I., 2017. QLBS: Q-learner in the black-scholes(-merton) worlds. URL: https://arxiv.org/abs/ 1712.04609. arxiv: 1712.04609.

[4] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A., Veness, Joel, Bellemare, Marc G., Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K., Ostrovski, Georg, Petersen, Stig, Beattie, Charles, Sadik, Amir, Antonoglou, Ioannis, King, Helen, Kumaran, Dharshan, Wierstra, Daan, Legg, Shane and Hassabis, Demis. "Human-level control through deep reinforcement learning." Nature 518 , no. 7540 (2015): 529--533.

[5] van Hasselt, H., Guez, A. & Silver, D. (2015). Deep Reinforcement Learning with Double Q-learning. CoRR, abs/1509.06461.

[6] Schaul, T., Quan, J., Antonoglou, I. & Silver, D. (2015). Prioritized Experience Replay (cite arxiv:1511.05952Comment: Published at ICLR 2016)

[7] Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M. & de Freitas, N. (2016). Dueling Network Architectures for Deep Reinforcement Learning.. In M.-F. Balcan & K. Q. Weinberger (eds.), ICML (p./pp. 1995-2003), : JMLR.org.

[8] Sutton, R. S. (1987). Learning to Predict by the Methods of Temporal Differences (TR87-509. 1). GTE Labs .

[9] Bellemare, M. G., Dabney, W. & Munos, R. (2017). A Distributional Perspective on Reinforcement Learning.. CoRR, abs/1707.06887.

[10] Venancio, T. M., Balaji, S., Geetha, S. & Aravind, L. (2010). Robustness and evolvability in natural chemical resistance: identification of novel systems properties, biochemical mechanisms and regulatory interactions. Mol. BioSyst. doi: 10.1039/c002567b

[11] M. Hessel et al., "Rainbow: Combining Improvements in Deep Reinforcement Learning." arXiv preprint arXiv:1710.02298, 2017.

[12] D. L. Clare Chen, Vincent Ying, Deep Q-Learning with Recurrent Neural Networks, http://cs229.stanford.edu/proj2016/report/ChenYingLaird-DeepQLearningWithRecurrentNeuralNetwords-report.pdf (2016)

[13] DQN and DRQN in partially observable gridworlds. (2020, March 30). Retrieved November 30, 2020, from https://kam.al/blog/drqn/

[14] Zhang, J., & Zhao, H. (2006). Asset pricing under jump diffusion. Hong Kong: School of Economics & Finance, University of Hong Kong.

Appendix 4: Automatic airport x-ray baggage scanner via adversarial domain adaptation

Jahyun (Lucrece) Shin

Background

The aim of this research is to build an automatic airport security baggage scanner. Currently, human operators at airports are inspecting if travellers' baggage contains any harmful objects. It would be both time-effective and cost-effective to implement a "smart" scanner that can automatically perform this scanning process to detect any harmful objects.

Data

X-ray images

Incheon International Airport located in Incheon, South Korea provided us with x-ray images of scanned baggage containing harmful objects, as shown in Figure 1. The provided x-ray images were labelled with seven classes: gun, knife, hard disk, phone, battery, USB, and shuriken. During initial stages, we decided to consider only two classes, gun and knife.

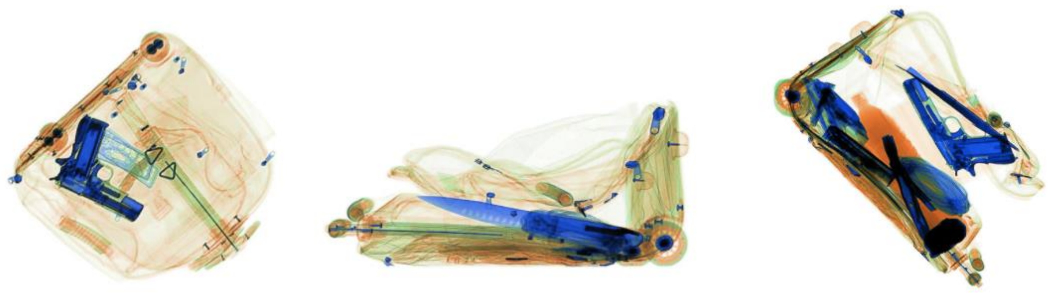


Figure 1. Examples of scanned x-ray images provided by Incheon International Airport

Removal of duplicates. The x-ray images contained many duplicates, in the form of different rotations of the same image. In order to calculate the number of unique images, all such duplicates were removed.

Removal of whitespace. Most gun and knife x-ray images came with a big white space that filled nearly three quarters of the entire image space. Since this is a waste of information for such high-dimensional data, they were cropped tightly to only contain the useful content of the image.

Removal of kitchen knife images. The given knife class is a mix of three different categories of knife: cutter knife, other knife, and kitchen knife. However, kitchen knife images are very clogged with a large overlap with other objects in the bag, as shown in Figure 2 (a) and (b). Although the final goal will be to detect the object in such clogged environments as well, for initial stages of model development, these images were considered as a source of noise, since the images of other categories of knife at least have a clear and isolated shape of knife, as shown in Figure 2 (c) and (d). Thus, kitchen knife images were removed from the dataset.

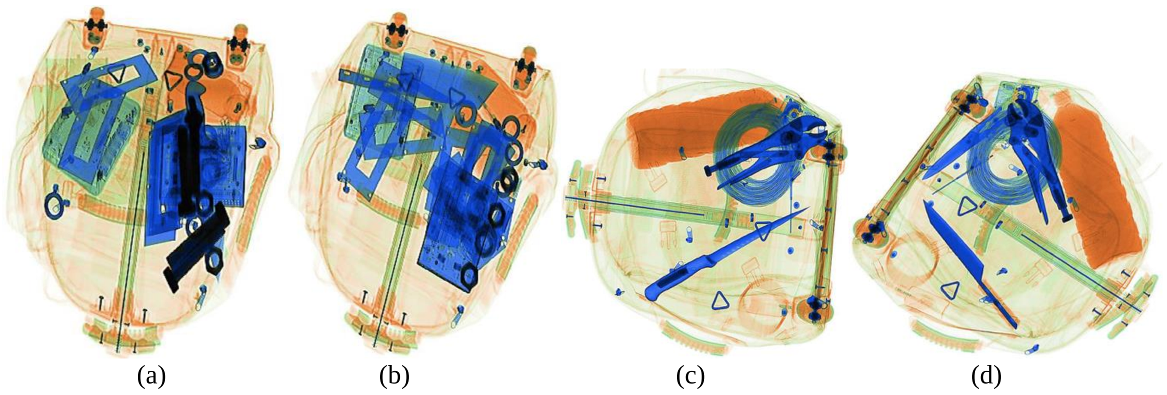


Figure 2. X-ray images of knife class (a), (b); kitchen knife images, (c); (d) other knife images

Google images

The biggest issue faced was that there were not a sufficient number of x-ray images to build a robust model without overfitting. The suggested solution was to scrape Google images of the same objects, as shown in Figure 3, which are openly available in tens of thousands of quantities from the Internet. When a model is trained with ample Google images, we can develop a technique to adapt the model to perform well for x-ray images as well.

Table 1 summarizes the number of Google images and x-ray images; it shows there are many more unique and labeled Google images than x-ray images.

Class	No. of Google images	No. of x-ray images (original)	No. of x-ray images (no duplicates or kitchen knife)
Knife	**1111**	550	**83**
Gun	**1045**	1050	**111**

Table 1. Number of Google and x-ray images.

Figure 3. Examples of scrapped Google images of gun and knife

Methodology

ResNext-50 backbone

In this problem, we take the image classification approach, where we will classify the whole image an object. For the main backbone model for image classification, we use Resnext-50 [3] for its relatively few number of parameters than other models such as vgg or AlexNet. It also has lower top-1 and top-5 error than the regular ResNet-50 or -101 [5].

"Benign" class

At first, a ResNext-50 model was trained with Google images of two classes (gun and knife). However, when the same model was tested with Google or x-ray images that do not contain gun or knife, it classified nearly all of them as knife, with 90 to 100% confidence. This result was alarming, since most baggage at the airport should not contain a gun or knife, and having such a high false alarm rate would be very inefficient.

Consequently, a third "benign" class was introduced, which represents all objects in the world that are not guns or knives. For this class, Google images of any random objects not related to gun and knife were scrapped. For x-ray images, given images of classes other than gun and knife (battery, USB, hard disk, and phone) were used.

Adversarial discriminative domain adaptation (ADDA)

Since there are not enough x-ray images to build a robust model, we use only the ample Google images with rich annotations as training data. We use the small number of x-ray images only for testing the model. Consequently, the specific problem is to reduce the difference in the distributions of Google images and x-ray images, so that the same model will work well on both types. Since x-ray images have inherently much different characteristics than Google images, a model trained only on the Google images might provide disappointing outcomes when tested on x-ray images [2]. This issue is known as *domain shift* [2].

To accommodate for domain shift, we introduce Adversarial Discriminative Domain Adaptation (ADDA) [1]. An overview of the model is shown in Figure 4. During training stage, the model first pre-trains a source encoder and a source classifier on the *class labels* of labeled source images. Next, a target encoder, which has the same structure with the source encoder, is initialized with the pre-trained weights of the source encoder from the first step. It then gets trained while the source encoder weights are frozen.

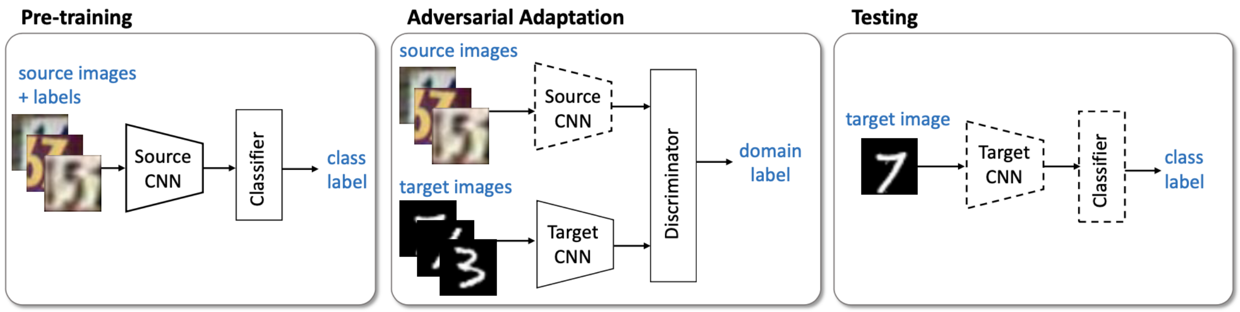


Figure 4. Overview of adversarial discriminative domain adaptation (ADDA)

The ADDA model assumes that the target domain is unlabeled, so it does not optimize for the *class labels* of the target domain. Instead, it introduces a discriminator model, a series of fully connected layers that tries to classify the *domain label* of each image. It receives both source encoder and target encoder's encoded features, and tries to classify which feature belongs to which domain. The goal is to confuse the discriminator so that it will not be able to tell if the encoded feature is from source domain or target domain. This way, the target encoder can be trained to map target images to a similar distribution with the source images.

Lastly, during the testing stage, we test the model with target images, using the target encoder and source classifier, to get the class labels for the target images.

Modifications. A few modifications are made to the original ADDA model. Instead of two separate encoders for the source and target domains, a single encoder is used. This eliminates the pre-training stage of the source encoder, and instead trains a single encoder with both source and target images together. Secondly, in the original ADDA, the target encoder was trained with the loss of domain classification only, since the source encoder weights were frozen. In the modified model, the single encoder is trained with source classification loss (only using the source images), and is simultaneously trained with domain classification loss with both source and target images. With such modifications, instead of separating the classification and domain adaptation tasks, they can be done at the same time.

Multi-label approach

Most x-ray images contain many more objects than the object of interest, as shown in Figures 1 and 2, while most Google images have a distinctive, isolated presence of the object of interest. Thus, this presence of many other objects in x-ray images can be quite confusing to the model that was trained only with Google images. To account for this point, instead of making the model to classify the image as one of the three classes, a more flexible approach was taken by making the model predict three things for an image: whether or not the image is benign, whether or not it is a gun, and whether or not it is a knife.

If using a single-label approach, the label of a single image would be a single number (i.e. 0, 1, or 2) for each class. For a multi-label model, a label for an image is composed of three binary numbers (0 and 1), where 1 means the label is present in the image and 0 means it is not. This way, a different label can be given for an image that contains more than one classes.

It is also found to be beneficial if a *soft label* was used for benign class, due to the model's tendency to always predict the benign class with more confidence. Thus, the label for only the benign class is changed to 0.5, while the other ones are left as 1.

All Google images' labels are re-formulated to distinguish between images that are with or without benign objects. For example, in Figure 5, images that look like (a) were given a label of [0, 0, 1] for [isBenign, isGun, isKnife] with only isKnife = 1. Images that look like (b) with a knife and benign objects both present were given a label of [0.5, 0, 1] with isBenign = 0.5 (soft label) and isKnife = 1.

(a) Google image with knife only

(b) Google image with knife and benign objects

Figure 5. Google images of knife class

Experiments

For experiments, we use ResNext-50 architecture provided in Pytorch. When training ADDA, the adversarial discriminator consists of 2 fully connected layers with 1024 and 2048 hidden nodes. Each layer uses a ReLU activation function. Optimization uses Adam optimizer for 50 epochs with a learning rate of 2e-6, and a batch size of 20 images. All training images are rescaled to 224x224 pixels. We used mean square error (MSE) loss was used for class classification by the encoder (in order to compensate for the soft labels of the benign class), and cross-entropy loss was used for domain classification by the discriminator.

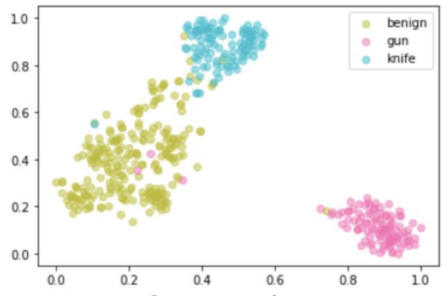
Results

The result of the experiment is presented in Table 2. It can be observed that using a multi-label approach with ADDA achieved the best result with both benign and gun classes. The largest increase in recall is for gun class, where it increased by nearly 75% from using a single-label approach without domain adaptation. For knife class, using a multi-label approach with ADDA only increased the accuracy by 5% than a single-label approach without ADDA, while staying low at only 12% recall.

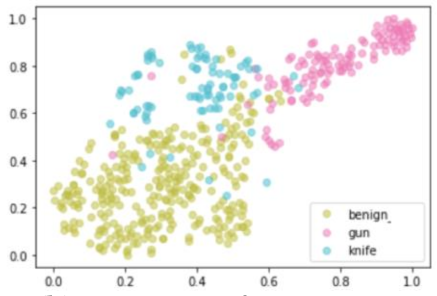
For *class-wise* qualitative analysis, Figure 6 visualizes with t-SNE [6] the feature representations of the multi-label ADDA model with different class labels. In (a) where Google image features are plotted, there is some area of overlap between the benign class (green) and knife class (blue), while gun class (pink) is separated far away. In (b) for x-ray image features, although the knife class (blue) seems to be a little bit separated from the benign class (green), the model seems to be classifying the whole area of both the benign and the knife class together as benign class (as observed in Table 2). It is thus not able to detect a clear decision boundary between benign and knife classes. On the other hand, the gun class (pink) is fairly separated from the two, which explains the high recall for gun class.

Method	Benign	Gun	Knife	Avg. (for gun and knife)
Source-only, single-label	0.997	0.137	0.072	0.209
Source-only, multi-label	0.997	0.436	**0.145**	0.291
ADDA, single-label	**1.000**	0.479	0.084	0.282
ADDA, multi-label	**1.000**	**0.889**	0.121	**0.505**

Table 2. Domain adaptation performance compared to source-only model



(a) Google image features



(b) Xray image features

Figure 6. t-SNE of class labels. t-SNE visualizations of ADDA features of Google and x-ray images of three different labels

For *domain-wise* qualitative analysis, Figure 7 visualizes with t-SNE [6] the feature representations of the ADDA model with the two different domain labels (Google as target domain- green and x-ray as source domain- pink). Near the beginning at epoch 5 of ADDA training, the two domains look quite separated. As training proceeds, however, the two become closer. At epoch 25, clusters of the two domains appear much closer to each other.

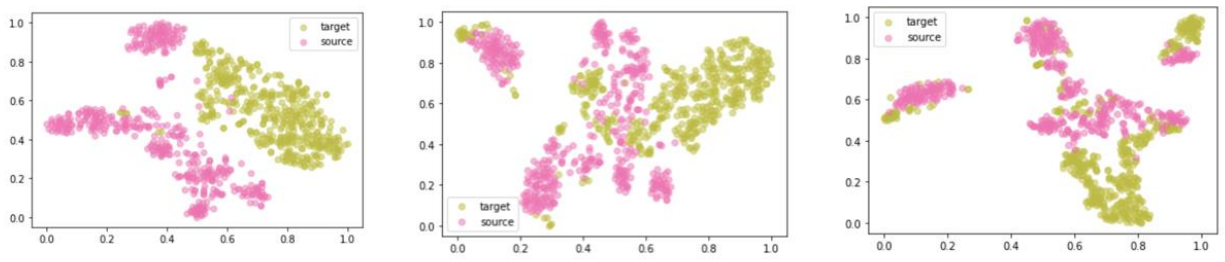


Figure 7. t-SNE of domain labels. t-SNE visualizations of ADDA features of Google and x-ray images at epoch 5, 15, and 25 of ADDA training

Conclusion

The ADDA model with multi-label approach was shown to be a simple yet effective method for domain adaptation between Google images and scanned x-ray images, despite large domain shifts. The effectiveness of the proposed method was demonstrated through higher recall for both gun and knife classes than using a single-label, source-only model without domain adaptation.

Since this is still an initial model development stage of the research, more effective and intuitive approaches will be considered in the future.

References

[1] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.
[2] Seungmin Lee, Dongwan Kim, Namil Kim, and Seong-Gyun Jeong. Drop to Adapt: Learning Discriminative Features for Unsupervised Domain Adaptation. In *IEEE CVPR*, 2019.
[3] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. In *CVPR*, 2017.
[4] Hui Tang and Kui Jia. Discriminative Adversarial Domain Adaptation. In AAAI, 2020.
[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
[6] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 2008.

Appendix 5: Incremental few-shot learning

Kuilin Chen, Chi-Guhn Lee

Introduction

Incremental learning is a learning paradigm that allows the model to continually learn new tasks on novel data, without forgetting how to perform previously learned tasks (Cauwenberghs & Poggio, 2001; Kuzborskij et al., 2013; Mensink et al., 2013). The capability of incremental learning becomes more important in real-world applications, in which the deployed models are exposed to possible out-of-sample data. Typically, hundreds of thousands of labelled samples in new tasks are required to re-train or fine-tune the model (Rebuffi et al., 2017). Unfortunately, it is impractical to gather sufficient samples of new tasks in real applications. In contrast, humans can learn new concepts from just one or a few examples, without losing old knowledge. Therefore, it is desirable to develop algorithms to support incremental learning from very few samples.

While a natural approach for incremental few-shot learning is to fine-tune part of the base model using novel training data (Donahue et al., 2014; Girshick et al., 2014), the model could suffer from severe over-fitting on new tasks due to a limited number of training samples. Moreover, simple fine-tuning also leads to significant performance drop on previously learned tasks, termed as catastrophic forgetting (Goodfellow et al., 2014). Recent attempts to mitigate the catastrophic forgetting are generally categorized into two streams: memory relay of old training samples (Rebuffi et al., 2017; Shin et al., 2017; Kemker & Kanan, 2018) and regularization on important model parameters (Kirkpatrick et al., 2017; Zenke et al., 2017). However, those incremental learning approaches are developed and tested on unrealistic scenarios where sufficient training samples are available in novel tasks. They may not work well when the training samples in novel tasks are few (Tao et al., 2020b).

To the best of our knowledge, the majority of incremental learning methodologies focus on classification problems and they cannot be extended to regression problems easily. In class-incremental learning, the model has to expand output dimensions to learn N0 novel classes while keeping the knowledge of existing N classes. Parametric models estimate additional classification weights for novel classes, while nonparametric methods compute the class centroids for novel classes. In comparison, output dimensions in regression problems do not change in incremental learning as neither additional weights nor class centroids are applicable to regression problems.

Besides, we find that catastrophic forgetting in incremental few-shot classification can be attributed to three reasons. First, the model is biased towards new classes and forgets old classes because the model is fine-tuned on new data only (Hou et al., 2019; Zhao et al., 2020). Meanwhile, the prediction accuracy on novel classes is not good due to over-fitting on few-shot training samples. Second, features of novel samples could overlap with those of old classes in the feature space, leading to ambiguity among classes in the feature space. Finally, features of old classes and classification weights are no longer compatible after the model is fine-tuned with new data.

In this paper, we investigate the problem of incremental few-shot learning, where only a few training samples are available in new tasks. A unified model is learned sequentially to jointly recognize all classes or regression targets that have been encountered in previous tasks (Rebuffi et al., 2017; Wu et al., 2019). To tackle aforementioned problems, we propose a nonparametric method to handle incremental few-shot learning based on learning vector quantization (LVQ) (Sato & Yamada, 1996) in deep embedded space. As such, the adverse effects of imbalanced weights in a parametric classifier can be completely avoided. Our contributions are three fold. First, a unified framework is developed, termed as incremental deep learning vector quantization (IDLVQ), to handle both incremental classification (IDLVQ-C) and regression (IDLVQ-R) problems. Second, we develop intra-class variance regularization, less forgetting constraints and calibration factors to mitigate catastrophic forgetting in class-incremental learning. Finally, the proposed methods achieve state-of-the-art performance on incremental few-shot classification and regression datasets.

Methodology

The general framework of IDLVQ for both classification and regression can be derived from a Gaussian mixture perspective (Ghahramani & Jordan, 1994), with a simplified covariance structure and supervised deep representation learning. In the base dataset (t = 1), a raw input x is projected into a feature space by a deep neural network f_{θ^1}, where θ^1 denotes the parameters in neural networks. In addition, N1 reference vectors $M^1 = \{m_1^1, \ldots, m_{N^1}^1\}$ are placed in the feature space, which can be learned to capture the representation of the base dataset. More reference vectors will be added incrementally while learning novel tasks.

The marginal distribution $p(f_{\theta^1}(x))$ of feature vector can be described by a Gaussian mixture model $p(f_{\theta^1}(x)) = \sum_{i=1}^{N^1} p(i)p(f_{\theta^1}(x)|i)$ of N1 components, where the prior $p(i) = 1/N^1$ and the component distribution $p(f_{\theta^1}(x)|i)$ is Gaussian. By assuming that each component distribution is isotropic Gaussian centred at m_i^1 with the same covariance, the posterior distribution of a component given the input is

$$p^1(i|x) = \frac{\kappa(f_{\theta^1}(x), m_i^1)}{\sum_{j=1}^{N^1} \kappa(f_{\theta^1}(x), m_j^1)}$$

where $\kappa(f_{\theta^1}(x), m_i^1) = \exp(-|f_{\theta^1}(x) - m_i^1|^2/\gamma)$ is a Gaussian kernel and γ is a scale factor. The conditional expectation of the output from a Gaussian mixture is $\hat{y} = \sum_{i=1}^{N^1} p^1(i|x)q_i^1$, where q_i^1 is the reference target associated with reference vector m_i^1. In classification problems, q_i^1 is either 0 or 1 indicating whether m_i^1 and x have the same label. Since each reference vector is assigned to a class at initialization, q_i^1 is fixed and does not require learning. Meanwhile, q_i^1 in regression problems is real-valued and has to be learned. The weights in neural networks θ^1, reference vectors M^1, reference targets q_i^1 (in regression problems only) and the scale factor γ are learned concurrently by minimizing a loss function between the true label y and the predicted label $\hat{y}$. The proposed IDLVQ is a nonparametric method as it makes prediction based on similarity to reference vectors, instead of using any regression or classification weights. The capacity of the model grows naturally by adding more reference vectors to learn novel tasks, while the old knowledge is preserved in existing reference vectors.

Experiments

We empirically evaluate the performance of IDLVQ-C on incremental few-shot classification on CUB200-2011 (Welinder et al., 2010) and miniImageNet datasets (Vinyals et al., 2016). The dataset is split into base classes and multiple groups of novel classes. We apply standard data augmentation, including random crop, horizontal flip and color jitter, on all training images. After each training session, the model performance is evaluated on a test set, which contains all classes that the model has been trained on.

CUB dataset is composed of 200 fine-grained bird species with 11,788 images. We split the dataset into 5894 training images, 2947 validation images and 2947 test images. All images are resized to 224 X 224. In addition, the first 100 classes are chosen as base classes, where all training samples in base classes are used to train the base model. The remaining 100 classes are treated as novel categories and split into 10 incremental learning sessions. Each incremental learning session contains 10 novel classes and 5 randomly selected training samples per class (10-way 5-shot).

miniImageNet dataset is a 100-class subset of the original ImageNet dataset (Deng et al., 2009). Each class contains 500 training images, 50 validation images, and 50 test images. The images are in RGB format of the size 84 X 84. We choose 60 and 40 classes for base and novel classes, respectfully. The 40 novel classes are divided into 8 sessions and each session contains 5 novel classes with 5 randomly selected training samples per class (5-way 5-shot).

ResNet18 (He et al., 2016) is used as the feature extractor for incremental classification problems. The learning process for each dataset is repeated 10 times and average test accuracy is reported.

The proposed method is compared to six methods for few-shot class-incremental learning: finetuning using Dt, joint training using the entire training set from all encountered classes, iCaRL (Rebuffi et al., 2017), Rebalancing (Hou et al., 2019), ProtoNet (Snell et al., 2017), incremental learning vector quantization (ILVQ) (Xu et al., 2012), SDC (Yu et al., 2020), and Imprint (Qi et al., 2018). Note that ILVQ is applied to the features extracted by neural networks in our experiment.

Method	sessions										
	1	2	3	4	5	6	7	8	9	10	11
Fine-tune	77.30	46.23	34.71	25.35	23.16	20.65	16.21	13.32	11.98	11.17	10.76
Joint train	77.30	73.28	68.80	65.34	63.75	62.00	60.81	59.71	**59.06**	**58.69**	**58.23**
iCaRL	77.30	57.18	54.67	48.11	40.76	36.85	33.12	30.42	28.22	26.84	25.23
Rebalancing	77.30	64.53	56.14	47.29	38.92	34.39	31.04	27.93	27.12	24.46	23.61
ProtoNet	77.30	69.76	66.01	62.29	59.58	57.10	55.13	54.09	52.40	51.65	50.36
ILVQ	77.30	71.50	66.79	62.71	60.20	57.84	55.27	55.06	52.42	51.72	50.47
SDC	77.34	74.45	69.45	65.27	61.81	58.26	56.14	55.71	53.31	52.79	51.52
Imprint	77.02	73.39	69.50	65.61	62.81	60.74	59.39	58.61	56.85	55.93	54.82
IDLVQ-C	**77.37**	**74.72**	**70.28**	**67.13**	**65.34**	**63.52**	**62.10**	**61.54**	**59.04**	**58.68**	**57.81**

Table 1. Prediction accuracy on CUB all classes using 10-way 5-shot incremental setting

Method	sessions								
	1	2	3	4	5	6	7	8	9
Fine-tune	64.25	30.11	18.53	6.31	2.86	2.68	1.87	1.56	1.42
Joint train	64.25	58.80	55.26	52.38	49.71	**48.37**	**45.91**	**44.68**	**43.38**
iCaRL	64.25	48.04	43.13	38.28	30.01	24.46	21.85	19.84	17.76
Rebalancing	64.25	49.21	44.17	37.71	30.11	22.92	19.99	17.96	16.25
ProtoNet	64.25	55.12	51.67	48.91	46.52	44.25	41.91	40.07	38.42
ILVQ	64.25	56.01	52.43	49.31	46.98	44.37	42.06	40.11	38.43
SDC	64.62	59.63	55.39	50.92	48.30	45.28	42.97	42.51	41.24
Imprint	64.71	59.85	55.71	52.47	**49.90**	47.31	44.57	42.57	41.26
IDLVQ-C	**64.77**	**59.87**	**55.93**	**52.62**	49.88	**47.55**	**44.83**	**43.14**	**41.84**

Table 2. Prediction accuracy on miniImageNet all classes using 5-way 5-shot incremental setting

The incremental few-shot learning results on CUB and miniImageNet are shown in Tables 1 and 2, respectively. Our method outperforms fine-tuning, iCaRL (Rebuffi et al., 2017), and ProtoNet (Snell et al., 2017) by a large margin. Simply fine-tuning the weights in classifier with few-shot training samples for novel classes significantly deteriorates the prediction accuracy. Although iCaRL alleviates catastrophic forgetting by tuning the model with a mix of old exemplars and novel few-shot data, the prediction accuracy still drops quickly because iCaRL requires sufficient samples per class to achieve satisfactory performance.

The ProtoNet relies on distance to prototypes (the mean of features within a class) to make classification but the fixed feature extractor may not be able to well separate novel classes. ILVQ is slightly better than ProtoNet because prototypes can be learned adaptively when more classes are available in incremental learning sessions. Some prototypes in ILVQ are close to the border of a class, which are more effective than class centroids in ProtoNet. However, ILVQ does not achieve the best performance because the feature extractor is fixed and cannot be learned along with the prototypes. IDLVQ-C has a small gain in the first couple of incremental few-shot learning sessions compared with SDC (Yu et al., 2020) and Imprint (Qi et al., 2018). Similar to ProtoNet, SDC also relies on prototypes to make classification. The performance of SDC is better than that of ProtoNet because SDC fine-tunes the feature extractor with novel dataset and compensates the drift in prototypes. However, the compensation for the drift of old-class prototypes can be less accurate in SDC because it is approximated by samples in novel classes. In parallel, the imprint method directly computes the normalized classification weights from the average of normalized features within a novel class. The imprint method avoids imbalanced classification weights and circumvents the overfitting in few-shot class-incremental learning through weight normalization. Nevertheless, the fixed feature extractor in the imprint method may not be well suited for novel classes. In contrast, IDLVQ-C updates the feature extractor only when necessary and compensates the shift of old reference vectors more accurately using exemplars from old classes. That is why the gain of IDLVQ-C increases with more incremental few-shot learning sessions. The performance of SDC, Imprint and IDLVQ-C is better than offline joint training in early sessions of incremental few-shot learning. Offline joint training may not result in oracle performance due to extremely imbalanced samples between base classes and novel classes.

Conclusions

A new incremental few-shot learning approach is developed to harmonize old knowledge preserving and new knowledge adaptation through quantized vector in deep embedded space. Prediction is made in a nonparametric way using similarity to learned reference vectors, which circumvents biased weights in a parametric classification layer during incremental few-shot learning. For classification problems, additional mechanisms are developed to mitigate the forgetting in old classes and improve representation learning for few-shot novel classes. For regression problems, the proposed approach has been reinterpreted as a kernel smoother to predict real-valued target over novel domain.

References

Michael Biehl, Anarta Ghosh, and Barbara Hammer. Dynamics and generalization ability of lvq algorithms. Journal of Machine Learning Research, 8(Feb):323–360, 2007.

FranciscoMCastro, Manuel J Mar´ın-Jim´enez, Nicol´as Guil, Cordelia Schmid, and Karteek Alahari.

End-to-end incremental learning. In Proceedings of the European conference on computer vision(ECCV), pp. 233–248, 2018.

Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In Advances in neural information processing systems, pp. 409–415, 2001.

Harm De Vries, Roland Memisevic, and Aaron C Courville. Deep learning vector quantization. In ESANN, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255. Ieee, 2009.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In International conference on machine learning, pp. 647–655, 2014.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 1126–1135. JMLR. org, 2017.

Bernd Fritzke. A growing neural gas network learns topologies. In Advances in neural information processing systems, pp. 625–632, 1995.

Zoubin Ghahramani and Michael I Jordan. Supervised learning from incomplete data via an em approach. In Advances in neural information processing systems, pp. 120–127, 1994.

Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4367–4375, 2018.

Spyros Gidaris and Nikos Komodakis. Generating classification weights with gnn denoising autoencoders for few-shot learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 21–30, 2019.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580–587, 2014.

Ian J Goodfellow, Mehdi Mirza, Aaron Courville Da Xiao, and Yoshua Bengio. An empirical investigation of catastrophic forgeting in gradient based neural networks. In In Proceedings of International Conference on Learning Representations ICLR, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

Saihui Hou, Xinyu Pan, Chen Change Loy, ZileiWang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 831–839, 2019.

Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13):3521–3526, 2017.

Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. From n to n+ 1: Multiclass transfer incremental learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3358–3365, 2013.

Mengjun Leng, Panagiotis Moutafis, and Ioannis A Kakadiaris. Joint prototype and metric learning for set-to-set matching: Application to biometrics. In 2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS), pp. 1–8. IEEE, 2015.

Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. IEEE transactions on pattern analysis and machine intelligence, 35(11):2624–2637, 2013.

Elizbar A Nadaraya. On estimating regression. Theory of Probability & Its Applications, 9(1): 141–142, 1964.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In European Conference on Computer Vision, pp. 524–540. Springer, 2020.

Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5822–5830, 2018.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In 5th International Conference on Learning Representations, ICLR 2017, 2017.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 2001–2010, 2017.

Mengye Ren, Renjie Liao, Ethan Fetaya, and Richard Zemel. Incremental few-shot learning with attention attractor networks. In Advances in Neural Information Processing Systems, pp. 5275– 5285, 2019a.

Zhenwen Ren, BinWu, Quansen Sun, and MingnaWu. Simultaneous learning of reduced prototypes and local metric for image set classification. Expert Systems with Applications, 134:102–111, 2019b.

Sascha Saralajew, Lars Holdijk, Maike Rees, and Thomas Villmann. Prototype-based neural network layers: incorporating vector quantization. In Advances in neural information processing systems - workshop poster, 2018.

Atsushi Sato and Keiji Yamada. Generalized learning vector quantization. In Advances in neural information processing systems, pp. 423–429, 1996.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 815–823, 2015.

Sambu Seo and Klaus Obermayer. Soft learning vector quantization. Neural computation, 15(7): 1589–1604, 2003.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In Advances in Neural Information Processing Systems, pp. 2990–2999, 2017.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In Advances in Neural Information Processing Systems, pp. 4077–4087, 2017.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1199–1208, 2018.

Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In European Conference on Computer Vision. Springer, 2020a.

Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Fewshot class-incremental learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12183–12192, 2020b.

Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. In Advances in Neural Information Processing Systems - Continual Learning workshop, 2019.

Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. Nature Communications, 11:4069, 2020.

Ehsan Variani, Erik McDermott, and Georg Heigold. A gaussian mixture model layer jointly optimized with discriminative features within a deep neural network architecture. In 2015 IEEE

International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4270–4274. IEEE, 2015.

Thomas Villmann, Michael Biehl, Andrea Villmann, and Sascha Saralajew. Fusion of deep learning architectures, multilayer feedforward networks and learning vector quantizers for deep classification learning. In 2017 12th international workshop on self-organizing maps and learning vector quantization, clustering and data visualization (WSOM), pp. 1–8. IEEE, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, DaanWierstra, et al. Matching networks for one shot learning. In Advances in neural information processing systems, pp. 3630–3638, 2016.

Wen Wang, Ruiping Wang, Shiguang Shan, and Xilin Chen. Prototype discriminative learning for face image set classification. In Asian Conference on Computer Vision, pp. 344–360. Springer, 2016.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

YandongWen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In European conference on computer vision, pp. 499–515. Springer, 2016.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 374–382, 2019.

Ye Xu, Furao Shen, and Jinxi Zhao. An incremental learning vector quantization algorithm for pattern classification. Neural Computing and Applications, 21(6):1205–1215, 2012.

Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, KaiWang, Yongmei Cheng, Shangling Jui, and Joost van deWeijer. Semantic drift compensation for class-incremental learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6982–6991, 2020.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 3987–3995, 2017.

Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13208–1.

Appendix 6: Security scan detection

Shashank Saurav, Anmol Garg

1.0 Introduction

Security concerns have made air travel an invasive affair. It is critical to the safety of all who fly that all luggage, hand luggage and each passenger's belongings are checked by scanners to ensure compliance with restrictions. Hence, X-ray scanning and analysis of items in the baggage is an essential aspect of airport security.

Current systems are designed to manually check for each scanned item using X-ray scanning equipment. This baggage check is usually required at airports or other transportation hubs such as bus or train stations where security of passengers is of high importance The project aims to utilize deep learning methodologies in Object Detection paradigm to automate baggage scanning and improve the security condition. This will further help save costs directed towards the manual checking and minimize any human mistakes.

1.1 Literature Review

One of the first Object Detection algorithms (1) repurposed classifiers to perform detections. To detect an object, these systems take a classifier and evaluate it at various locations and scales in an image using sliding windows approach. It involves taking multiple crops of the input image and then each crop is fed to a deep Convolutional Neural Network (CNN) to make a classification decision. The crop dimensions, shape would be an issue as the object may appear at any location with any shape and size. This limitation was alleviated by applying several crop sizes and dimensions and running them all through the CNN but this posed a time and computation cost issue.

In one of the other papers (2) Regional – Convolutional Neural Network (R-CNN) region proposal networks were used to capture blobby (feature-rich) regions which spits out a finite number of boxes where an object can be potentially present. These limited region proposals are then fed to a CNN for object classification. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene. However, this still needed finite number of boxes to be fed to CNN for processing rendering redundant computations. Variants of R-CNN such as Fast R-CNN & Faster R-CNN (3) have come up since then.

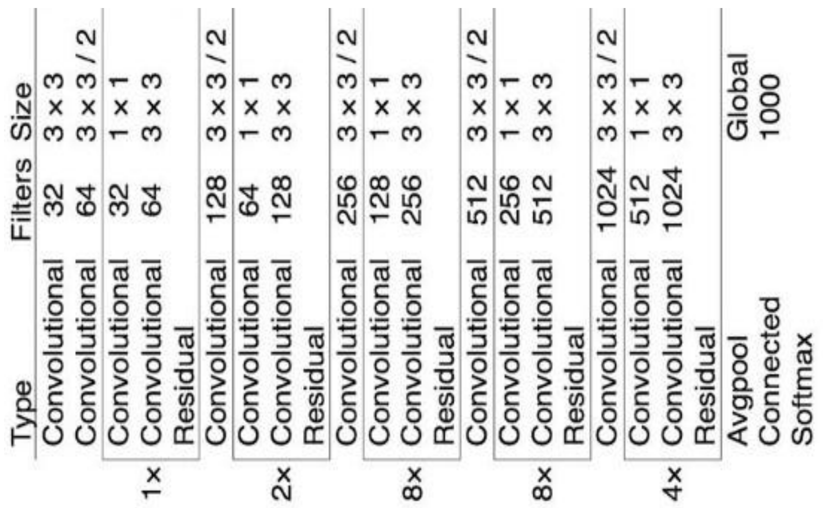
The current state-of-the-art algorithm You Only Look Once (YOLO) (4) entered the domain in 2018 and dominated the Object Detection paradigm since then with incremental improvements in different versions. Instead of doing multiple forward propagations of proposed regions on deep CNN, YOLO makes grid cells of the image and then makes proposed predictions on each grid cell. This way, only a single forward propagation of the image through a deep Neural Network is needed (thus, the name, You Only Look Once) which saves computation cost and time.

2.0 You Only Look Once (YOLO V3)

You Only Look Once (YOLO V3) is an improvement over previous YOLO detection networks. Compared to prior versions, it features multi-scale detection, stronger feature extractor network, and some changes in the loss function. As a result, this network can now detect many more targets from big to small. And, of course, just like other single-shot detectors, YOLO V3 also runs quite fast and makes real-time inference possible.

2.1 Architecture

YOLO V3 network makes use of only convolutional layers, making it a fully convolutional network (FCN).



	Type	Filters	Size
	Convolutional	32	3 × 3
	Convolutional	64	3 × 3 / 2
1×	Convolutional	32	1 × 1
	Convolutional	64	3 × 3
	Residual		
	Convolutional	128	3 × 3 / 2
2×	Convolutional	64	1 × 1
	Convolutional	128	3 × 3
	Residual		
	Convolutional	256	3 × 3 / 2
8×	Convolutional	128	1 × 1
	Convolutional	256	3 × 3
	Residual		
	Convolutional	512	3 × 3 / 2
8×	Convolutional	256	1 × 1
	Convolutional	512	3 × 3
	Residual		
	Convolutional	1024	3 × 3 / 2
4×	Convolutional	512	1 × 1
	Convolutional	1024	3 × 3
	Residual		
	Avgpool		Global
	Connected		1000
	Softmax		

Fig. 1: YOLO V3 Architecture (5)

The whole network is a chain of multiple blocks with some strides 2 convolution layers in between to reduce dimension. Inside a block, there's just a bottleneck structure (1x1 followed by 3x3) plus a skip connection. No form of pooling is used.

2.2 Working

The whole system can be divided into two major components: Feature Extractor and Detector; both are multi-scale. When a new image comes in, it goes through the feature extractor first so that we can obtain feature embeddings at three (or more) different scales. Then, these features are fed into three (or more) branches of the detector to get bounding boxes and class information.

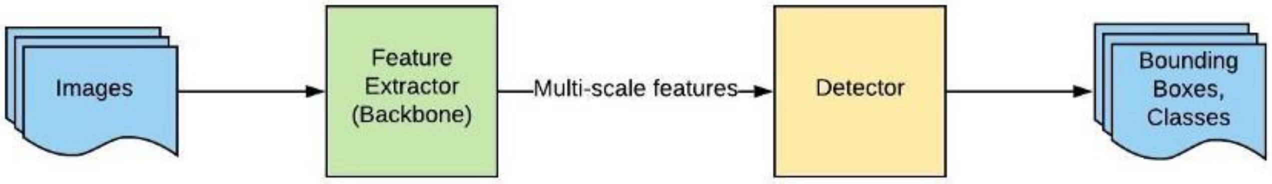


Fig. 2: Overall Working (5)

2.2.1 Multi-Scale Feature Extractor

If the goal is to do multi-class classification, an average pooling and a 1000 ways fully connected layers plus softmax activation will be added. However, in the case of object detection, detection head instead of classification head is added to the feature extractor. And since YOLO V3 is designed to be a multi-scaled detector, features from last three residual blocks are all used in the later detection. In the diagram below, assuming the input is 416x416, so three scale feature vectors would be of size 52x52, 26x26, and 13x13.

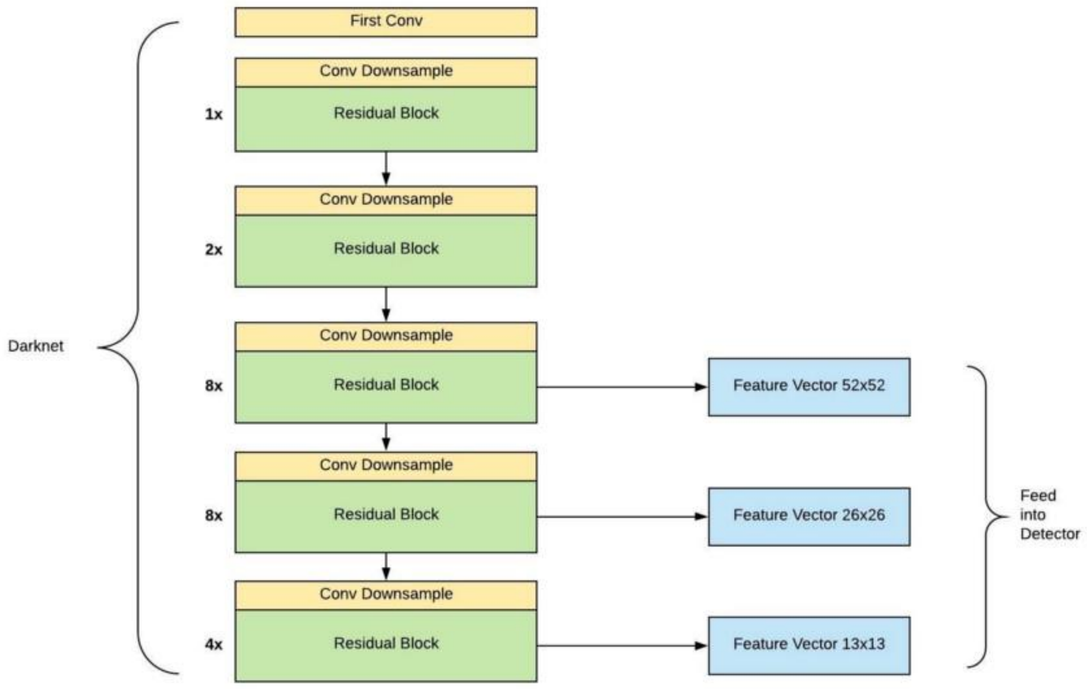


Fig. 3: Multi-Scale Feature Extractor (5)

The network down samples the input image until the first detection layer, where a detection is made using feature maps of size 13x13. Further, layers are up sampled by a factor of 2 and concatenated with feature maps of a previous layers having identical feature map sizes. Another detection is now made using feature maps of size 26x26. The same up sampling procedure is repeated, and a final detection is made using feature maps of size 52x52.

2.2.2 Multi-Scale Detector

Once we have three features vectors, we can now feed them into the detector. Multiple 1x1 and 3x3 convolution layers are used before a final 1x1 convolution layer to form the final output. For medium and small scale, it also concatenates features from the previous scale.

Assuming the input image is (416, 416, 3), the final output of the detectors will be in shape of *[(52, 52, 3, (4 + 1 + num_classes)), (26, 26, 3, (4 + 1 + num_classes)), (13, 13, 3, (4 + 1 + num_classes*))] where num_classes is the total number of classes that a detected item, in an image, can belong to. The three items in the list represent detections for three scales.

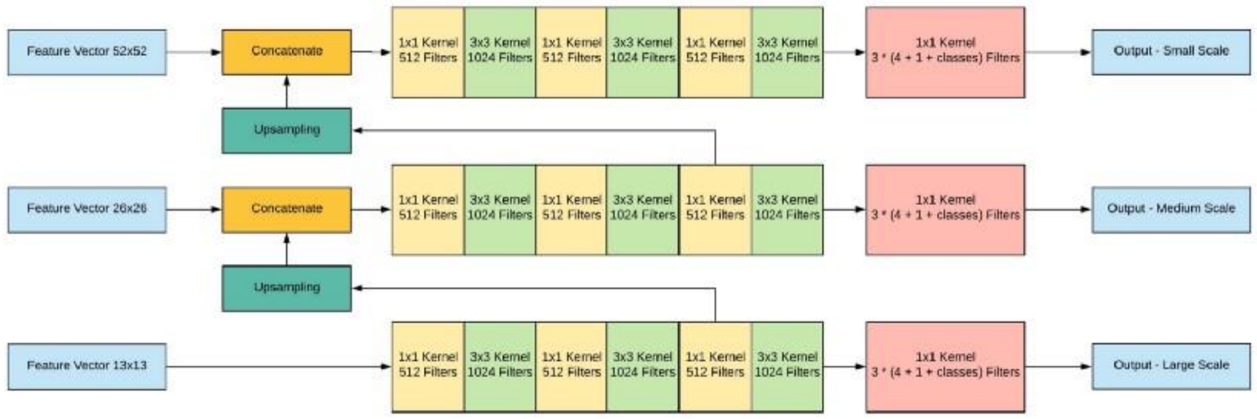


Fig. 4: Multi-Scale Detector (5)

2.2.3 Interpreting Output

The goal of object detection is to get a bounding box and its class.

Anchor box is a prior box that have pre-defined aspect ratios. These aspect ratios are determined before training by running K-means on the h, w of ground truth bounding boxes of training dataset.

Since the convolution outputs a square matrix of feature values (like 13x13, 26x26, and 52x52 in YOLO), we define this matrix as a grid and assign anchor boxes to each cell of the grid. In other words, anchor boxes anchor to the grid cells, and they share the same centroid. And once we defined those anchors, we can determine how much does the ground truth box overlap with the anchor box and pick the one with the best Intersection Over Union (IOU) area and couple them together.

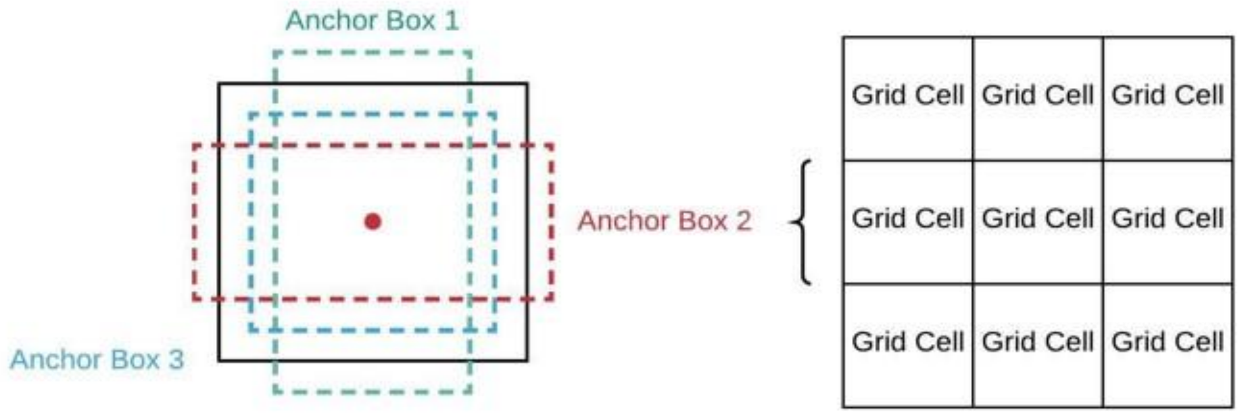


Fig. 5: Anchor Box (5)

In YOLO v3, we have three anchor boxes per grid cell. And we have three scales of grids. Therefore, we will have 52x52x3, 26x26x3 and 13x13x3 anchor boxes for three different scales. For each anchor box, network predicts 3 things:

1. The location offset against the anchor box: t_x, t_y, t_w, t_h. This has 4 values.
2. The objectness score to indicate if this box contains an object. This has 1 value.
3. The class scores to tell us which class this box belongs to. Number of values is equal to number of classes.

In summary, we are predicting 4 + 1 + num_classes values for one anchor box.

The following log transformations describe how the network output is transformed to obtain bounding box predictions.

x = sigmoid(tx) + Cx
y = sigmoid(tx) + Cy
w = exp(tw) * pw
h = exp(th) * ph
where,
x, y = center coordinates of the predicted bounding box
w, h = width and height of the predicted bounding box respectively
Cx, Cy = Top left coordinates of the grid cell
pw, ph = Dimensions of anchor boxes for the grid cell

The objectness score is passed through a sigmoid, to interpret as a probability (P_0). The class scores are passed through softmax to represent the probabilities of the detected object belonging to a particular class (P_C^j)

2.3 Training

YOLO V3 makes prediction across 3 different scales. This means, with an input image of size 416 x 416, we make detections on scales 13 x 13, 26 x 26 and 52 x 52. And, therefore we have 52x52x3, 26x26x3 and 13x13x3 bounding boxes predictions for the three scales respectively.

Out of many bounding box predictions per scale, only one needs to be chosen to compare against the ground truth bounding box and further calculate loss. The cell (on the input image) containing the center of the ground truth box is chosen to be the one responsible for predicting the object. Now, this cell can predict three bounding boxes. The bounding box responsible for detecting the object (final predicted bounding box) will be the one whose anchor has the highest IOU area with the ground truth box. Finally, loss is calculated using that final predicted bounding box and the true label.

The above steps are applied to all the three scale of image separately to get one prediction vector at each scale. Loss for each scale are summed together to get total loss for that image.

With the final predicted bounding box, we can now calculate the loss against the ground truth labels.

The loss function consists of four parts:

- L-2 Regression Loss on boundary box co-ordinates and dimensions
- Cross Entropy Loss on predicted class probabilities (P_C^j)
- Binary Cross Entropy Loss on objectness probability (P_0) which indicates how likely is there an object in the final predicted bounding box
- Binary Cross Entropy Loss on no-objectness probability to penalize false positives proposals.

$$Loss = -\,t\ln P_0^* - (1-t)\ln P_0^* - \sum_{i=1}^{N_b} \ln\left(1-P_0^i\right) + \left|\left|x-x_t\right|\right|_2^{\,2} + \left|\left|y-y_t\right|\right|_2^{\,2} + \left|\left|w-w_t\right|\right|_2^{\,2} + \left|\left|h-h_t\right|\right|_2^{\,2} - \sum_{j=1}^{N_c} C_j * \ln(P_C^j)$$

where,
t = 1 if object is present, else t = 0
N_b : bounding boxes from all the grid cells except the responsible grid cell
P_0^i : P_0 corresponding to N_b
P_0^*: P_0 corresponding to final predicted bounding box
$x_t,\ y_t, w_t,\ h_t$: true bounding box coordinates and dimensions
x, y, w, h: box coordinates and dimensions corresponding to final predicted bounding box
C_j = 1 if true class = j, else C_j = 0

2.4 Testing

At test phase, for a given input image of size 416 x 416, YOLO predicts ((52 x 52) + (26 x 26) + 13 x 13)) x 3 = 10647 bounding boxes. In order filter one, out of 10647 predicted bounding boxes, against each ground truth bounding box, **Non-Max Suppression (NMS)** is applied on the predictions.

For each input image:

- Drop all predicted bounding boxes with objectness probability less than a certain threshold (confidence threshold)

- For each class of object detected, sort the remaining predictions is descending order of objectness probability
    - Compare intersection-over-union (IOU) areas between the top prediction and the rest, and drop whose areas is above a certain threshold (NMS threshold)
    - Repeat the above step for the next top prediction.

3.0 Methodology

The methodology followed is discussed using the different aspects. The overall approach is summarized below:

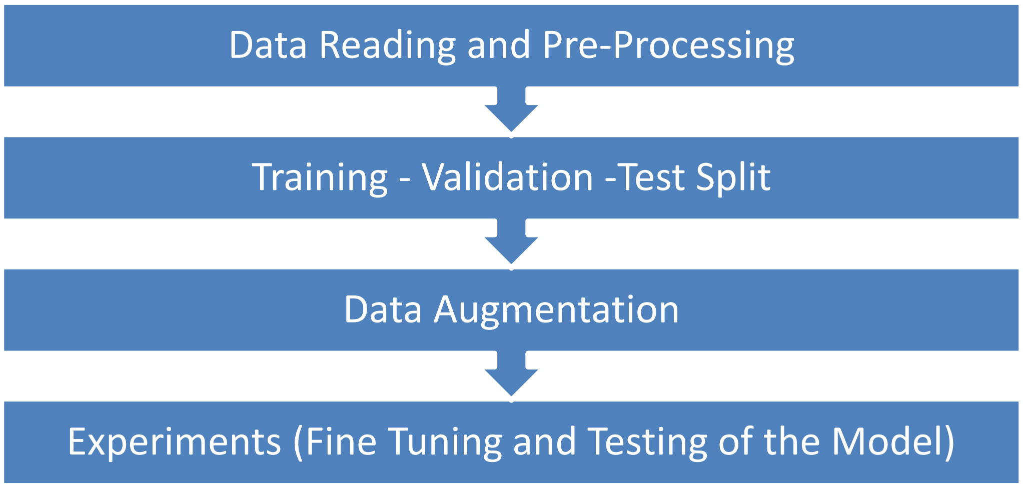

Data Reading and Pre-Processing
Training - Validation -Test Split
Data Augmentation
Experiments (Fine Tuning and Testing of the Model)

Fig. 6: Methodology Pipeline

3.1 Data Reading and Pre-Processing

In the current work, a dataset containing 2,050 images is used containing different class of items for detection. The breakdown of the dataset is as follows:

Class Description	Class ID	Original Count of Images
Handgun	0	549
Kitchen Knife	1	50
Cutter Knife	2	50
Hard Disk	3	50
USB	4	50
Hard Disk – 2	5	50
Kitchen Knife – 2	6	50
Shuriken	7	50
Battery	8	50
Battery – 2	9	50
Phone	10	50

Table 1. Dataset breakdown

For each image, ground truth bounding boxes' dimensions were provided for each banned item present in the image in the following format:

[class id of the banned item, x, y, w, h] where, x, y: centre coordinates of the bounding box
w, h: width and height of the bounding box

To ease the detection, different types of hard disk, kitchen knife and battery were combined, giving a total of 7 classes. Since all the input images had different dimensions, therefore the images were resized and padded to dimension of 416 which ensures that the grid scales at which detection is performed comes out to be 13, 26 & 52 respectively. This resizing operation is performed while maintaining the aspect ratio of the image.

Metrics: The metrics we considered to evaluate the model were precision and recall.

- $\text{Recall} = \frac{\text{Number of banned items detected}}{\text{Total Number of banned items Actually Present}}$
- $\text{Precision} = \frac{\text{Number of banned items in the predictions}}{\text{Total Number of items detected as banned}}$

More focus was given on recall to avoid False Negatives

3.2 Training-Validation-Test Split

The breakdown of the dataset split is as follows:

Class	Class ID	Original Count	Train Count	Valid Count	Test Count
Handgun	0	549	524	15	10
Knife	1	150	125	15	10
Phone	2	50	25	15	10
Hard Disk	3	100	75	15	10
USB	4	50	25	15	10
Shuriken	5	50	25	15	10
Battery	6	100	75	15	10
Total		1049	674	105	70

Table 2. Dataset split

3.3 Data Augmentation

As the count of images for most of the classes were less, the images were augmented by flipping and rotation. Accordingly, the ground truth bounding boxes' coordinates and dimensions were changed.

- **Image Flipping**: The images were flipped horizontally, vertically, and both horizontally & vertically i.e. dual flipping. The center coordinates of the ground truth bounding boxes were calculated accordingly. The width & height of the boundary box remained the same.

Horizontal flip:

$x_{new} = 1 - x_{orig}$
$y_{new} = y_{orig}$

Vertical flip:

$x_{new} = x_{orig}$
$y_{new} = 1 - y_{orig}$

Both Horizontal & Verical (Dual) flip:

$x_{new} = 1 - x_{orig}$
$y_{new} = 1 - y_{orig}$

- **Image Rotation:** Each of the flipped images were rotated clockwise by 90, 180 & 270 degrees. The center coordinate as well as height and width were adjusted accordingly.

90-degree clockwise rotation:

$x_{new} = 1 - y_{orig}$
$y_{new} = x_{orig}$
$w_{new} = h_{orig}$
$h_{new} = w_{orig}$

180-degree clockwise rotation:

$x_{new} = 1 - x_{orig}$
$y_{new} = 1 - y_{orig}$
$w_{new} = w_{orig}$
$h_{new} = h_{orig}$

270-degree clockwise rotation:

$x_{new} = y_{orig}$
$y_{new} = 1 - x_{orig}$
$w_{new} = h_{orig}$
$h_{new} = w_{orig}$

The count of images after augmentation is summarized below:

Class	Class ID	Original Count	Train Count (Augmented)	Valid Count	Test Count
Handgun	0	549	8384	15	10
Knife	1	150	2400	15	10
Phone	2	50	800	15	10
Hard Disk	3	100	1600	15	10
USB	4	50	800	15	10
Shuriken	5	50	800	15	10
Battery	6	100	1600	15	10
Total		1049	16384	105	70

Table 3. Dataset split after data augmentation

3.4 Experiments (Training and Testing)

For training, we used pre-trained weights of YOLO v3 model (trained on COCO dataset). Since none of the classes were common in scanned baggage and COCO dataset and domain of both dataset were very different (visibly), all layers of YOLO v3 network were fine tuned.

3.4.1 Experiment - 1

1. We weighted the no objectness part of the loss function (discussed in 2.2.4) to heavily penalize false positives. Following is the loss function that was used for training:

$$Loss = -t\ln P_0^* - (1-t)\ln P_0^* - 5 * \sum_{i=1}^{N_b} \ln\left(1 - P_0^i\right) + \left\|x - x_t\right\|_2^2 + \left\|y - y_t\right\|_2^2 + \left\|w - w_t\right\|_2^2 + \left\|h - h_t\right\|_2^2 - \sum_{j=1}^{N_c} C_j * \ln(P_C^j)$$

2. Oversampled images of minority class to deal with class imbalance.

Class Description	Training Count of Images (Oversampled)	Validation Count of Images	Test Count of Images
Handgun	8384	15	10
Knife	8000	15	10
Hard Disk	7200	15	10
USB	8000	15	10
Shuriken	8000	15	10
Battery	7200	15	10
Phone	8000	15	10

Table 4. Oversampled training set

3. Hyperparameters:

- Learning Rate = 0.01
- Batch Size = 18
- Optimizer = Adam
- Confidence Threshold = 0.9
- IOU threshold = 0.5
- NMS Threshold = 0.4
- Training Epochs = 32

Results

a) Model Dynamics

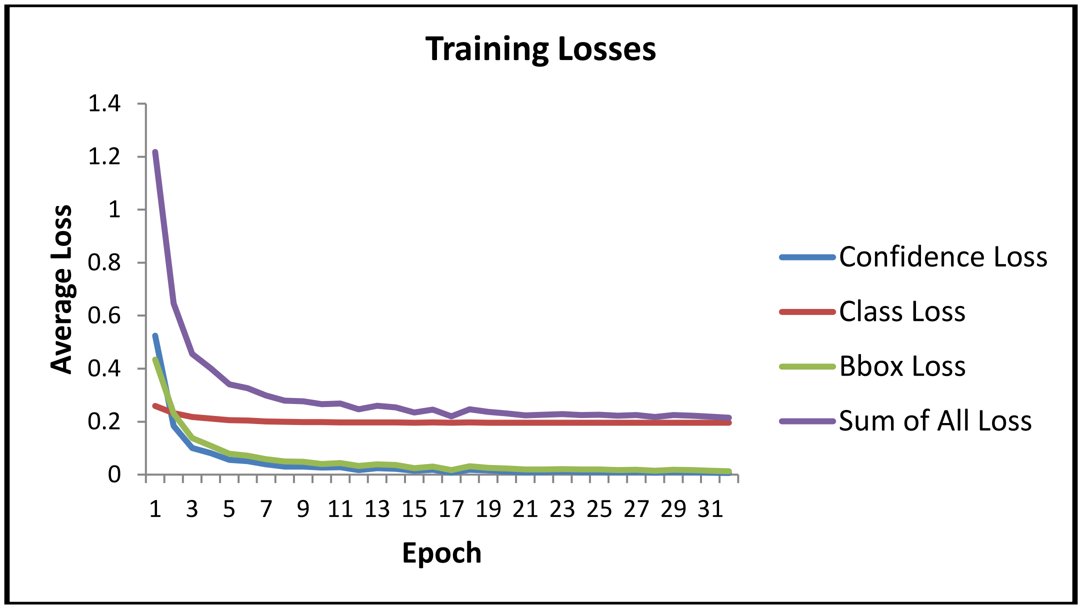


Fig. 7: Training Dynamics

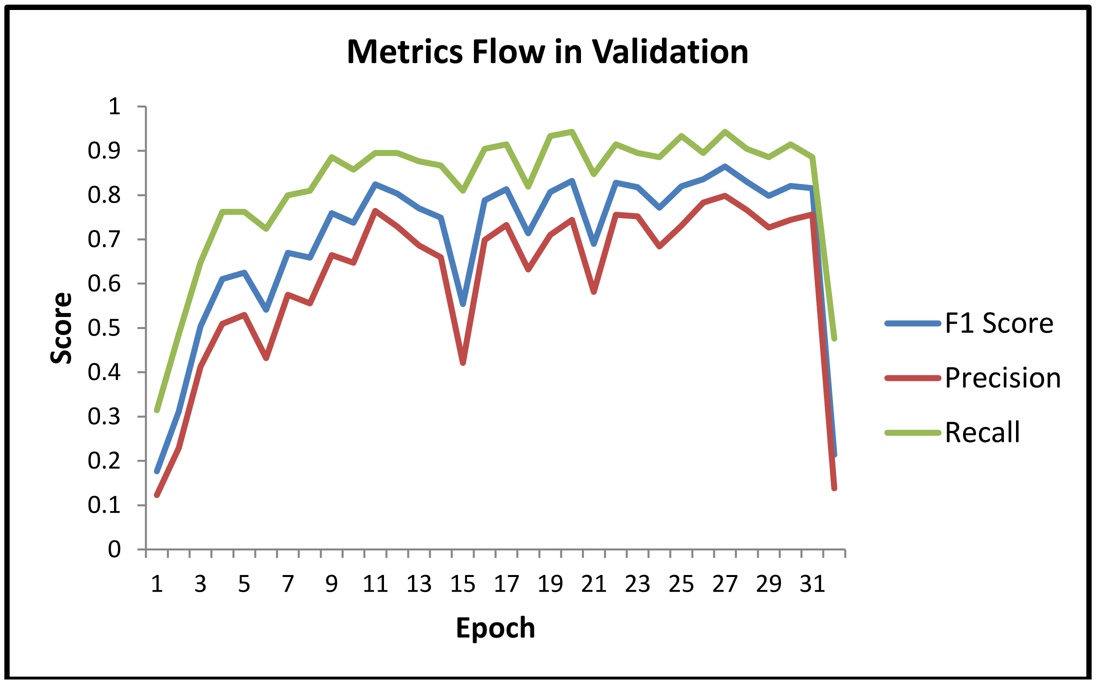


Fig. 8: Validation Metrics Flow

b) Scores

Best Recall at Epoch no 20

Best Precision at Epoch no 27

After 27 Epochs: Best Validation Precision

	Training			Validation			Test		
Class Description	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %
Handgun	99.98	100	99.99	100	100	100	100	100	100
Knife	99.60	100	99.80	100	93.33	96.55	90.91	100	95.24
Hard Disk	96.62	100	98.28	77.78	93.33	84.85	90.91	100	95.24
USB	99.75	100	99.87	81.82	60.0	69.23	77.78	**70.0**	73.68
Shuriken	50.0	100	66.67	50.0	100	66.67	**50.0**	90.0	64.28
Battery	99.50	100	99.75	100	100	100	90.91	100	95.24
Phone	98.76	100	99.38	83.33	100	90.91	69.23	90.0	78.26
Average	**96.79**	**100**	**98.37**	**80.16**	**91.43**	**85.84**	**78.31**	**92.86**	**84.97**

Table 5. Result on best precision epoch

After 20 Epochs: Best Validation Recall

	Training			Validation			Test		
Class Description	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %
Handgun	99.69	100	99.84	100	100	100	100	100	100
Knife	98.72	100	99.35	75.0	100	88.71	76.92	100	86.96
Hard Disk	94.71	100	97.28	65.22	100	78.95	90.91	100	95.24
USB	99.75	100	99.87	57.55	66.67	60.61	75.0	**60**	66.67
Shuriken	49.94	100	66.61	50.0	100	66.67	**47.37**	90	62.07
Battery	98.28	100	99.13	100	100	100	90.91	100	95.24
Phone	97.80	100	98.89	88.23	100	93.75	69.23	90	78.26
Average	**96.21**	**100**	**98.07**	**72.46**	**95.24**	**82.30**	**75.29**	**91.43**	**82.58**

Table 6. Result on best recall epoch

3.4.2 Experiment - 2

1. We weighted the no objectness part of the loss function (discussed in 2.2.4) to heavily penalize false positives. The following is the loss function used for training:

$Loss = -t\ln P_0^* - (1-t)\ln P_0^* - 5 * \sum_{i=1}^{N_b} \ln\left(1 - P_0^i\right) + \left|\left|x - x_t\right|\right|_2^{\ 2} + \left|\left|y - y_t\right|\right|_2^{\ 2} + \left|\left|w - w_t\right|\right|_2^{\ 2} + \left|\left|h - h_t\right|\right|_2^{\ 2} - \sum_{j=1}^{N_c} C_j * \ln(P_C^j)$

2. To deal with class imbalance, instead of oversampling, balanced batch was sent at every iteration. Single batch comprised of 3 images of each of Phone, Knife, USB, Shuriken and 2 images for each of Handgun, Hard Disk and Battery

3. Hyperparameters:

- Learning Rate = 0.01
- Batch Size = 18
- Optimizer = Adam
- Confidence Threshold = 0.9
- IOU threshold = 0.5
- NMS Threshold = 0.4
- Training Epochs = 42

Results

a) Model Dynamics

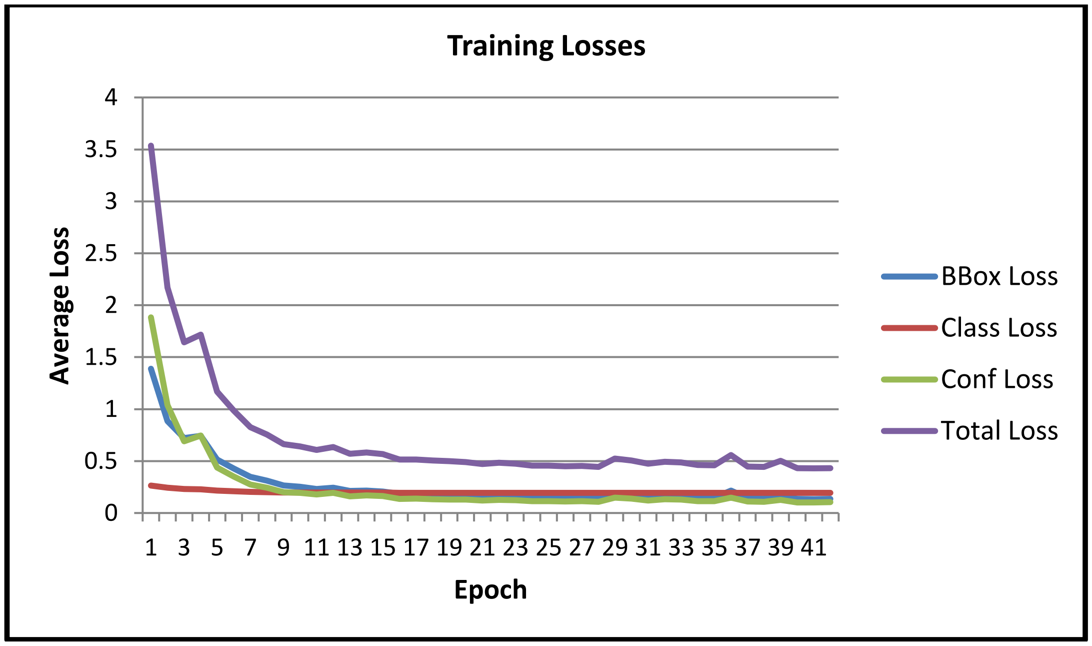


Fig. 9: Training Dynamics

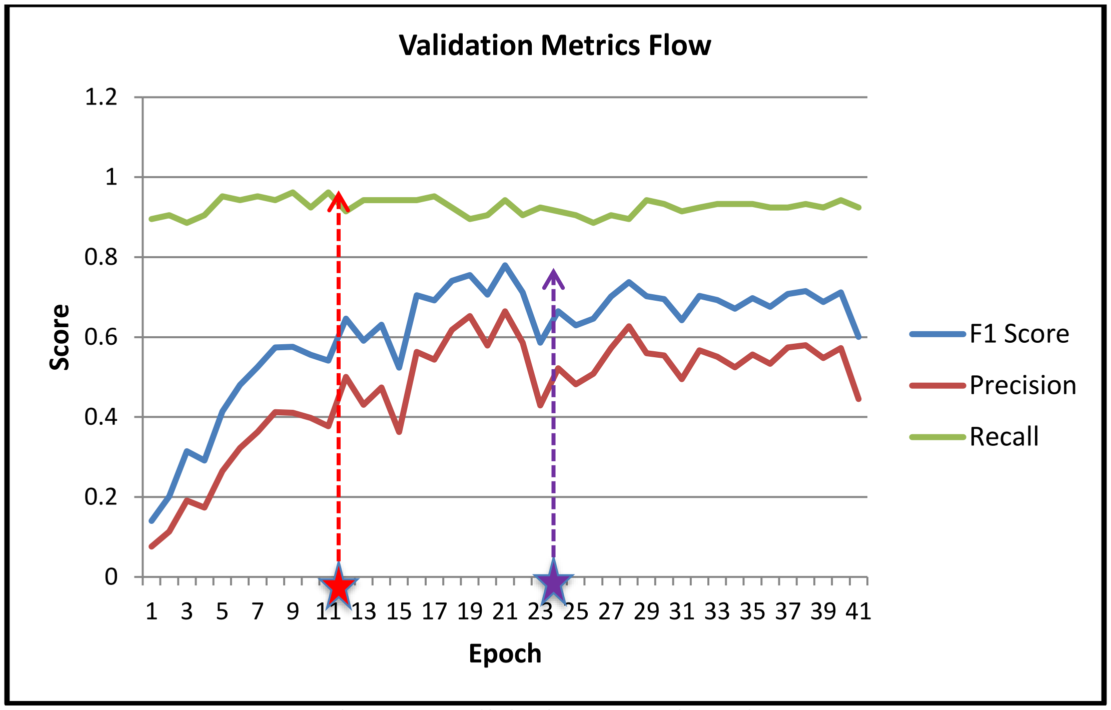


Fig. 10: Validation Metrics Flow

b) Scores

★ Best Overall Precision and F1 at Epoch no 21

★ Best Overall Recall at Epoch no 9

<u>After 21 Epochs: Best Validation Precision</u>

	Training			Validation			Test		
Class Description	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %
Handgun	83.88	99.98	91.23	93.75	100	96.77	100	100	100
Knife	96.01	100	97.97	93.75	100	96.77	100	100	100
Hard Disk	94.48	100	97.16	83.33	100	90.91	83.33	100	90.91
USB	51.41	100	67.91	45	60	51.43	70	70	70
Shuriken	93.9	100	96.85	100	100	100	83.33	100	90.91
Battery	91.4	100	95.5	88.23	100	93.75	83.33	100	90.91
Phone	98.76	100	99.37	88.23	100	93.75	64.29	90	75
Average	**85.95**	**99.99**	**92.44**	**83.19**	**94.28**	**88.39**	**82.5**	**94.28**	**88**

Table 7. Result on best precision epoch

After 9 Epochs: Best Validation Recall

	Training			Validation			Test		
Class Description	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %
Handgun	71.38	99.98	83.3	68.18	100	81.08	71.43	100	83.33
Knife	81.72	99.95	89.92	88.23	100	93.75	100	100	100
Hard Disk	67.72	100	80.75	50	100	66.67	45.45	100	62.5
USB	44.29	100	61.4	33.33	73.33	45.83	43.75	70	53.84
Shuriken	83.16	100	90.8	88.24	100	93.75	83.33	100	90.91
Battery	68.03	100	80.97	60	100	75	78.43	100	83.33
Phone	80	100	88.88	57.69	100	73.17	50	90	64.28
Average	**71.3**	**99.98**	**83.24**	**59.41**	**96.19**	**73.45**	**62.26**	**94.28**	**75**

Table 8. Result on best recall epoch

Note: NMS Threshold reduced to 0.0005 from 0.4 for testing

3.4.3 Experiment - 3

1. We weighted the no objectness part of the loss function (discussed in 2.2.4) to heavily penalize false positives. Following is the loss function that was used for training:

$$Loss = -t\ln P_0^* - (1-t)\ln P_0^* - \mathbf{5} * \sum_{i=1}^{N_b} \ln\left(1 - P_0^i\right) + \left\|x - x_t\right\|_2^2 + \left\|y - y_t\right\|_2^2 + \left\|w - w_t\right\|_2^2 + \left\|h - h_t\right\|_2^2 - \sum_{j=1}^{N_c} C_j * \ln\left(P_C^j\right)$$

2. To deal with class imbalance, instead of oversampling, balanced batch was sent at every iteration. Single batch comprised of 3 images of each of Phone, Knife, USB, Shuriken and 2 images for each of Handgun, Hard Disk and Battery.

3. Anchor Box dimensions were obtained by running K-Means on scanned baggage images. Custom anchor box dimensions obtained were (19, 19), (88, 65), (67, 87), (124, 95), (95, 124), (68, 192), (192,70), (166, 126), (125, 169).

4. Hyperparameters:

- Learning Rate = 0.01
- Batch Size = 18
- Optimizer = Adam
- Confidence Threshold = 0.9
- IOU threshold = 0.5
- NMS Threshold = 0.4
- Training Epochs = 42

Results

a) Model Dynamics

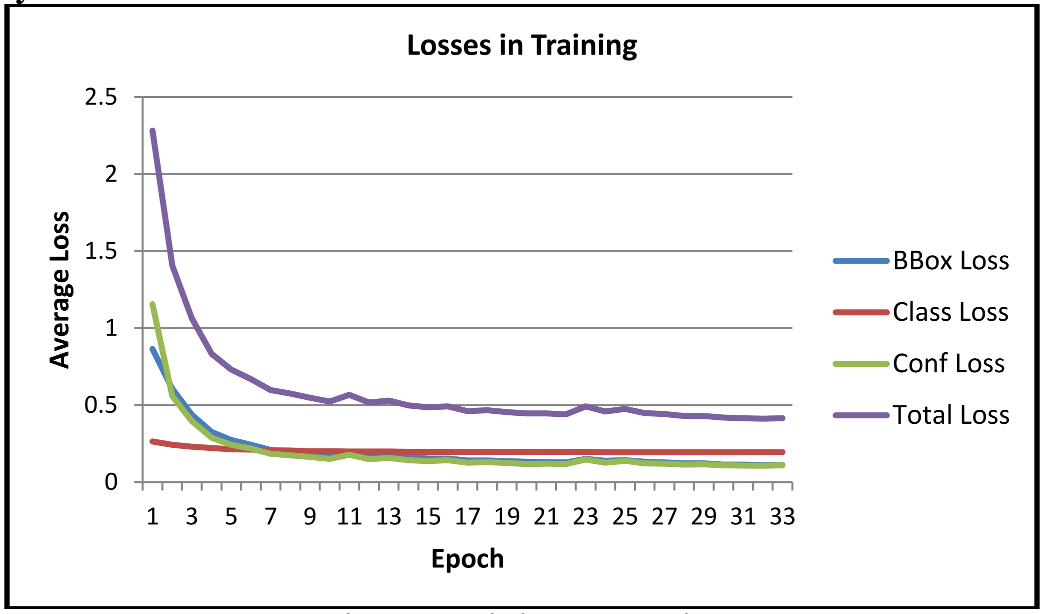


Fig. 11: Training Dynamics

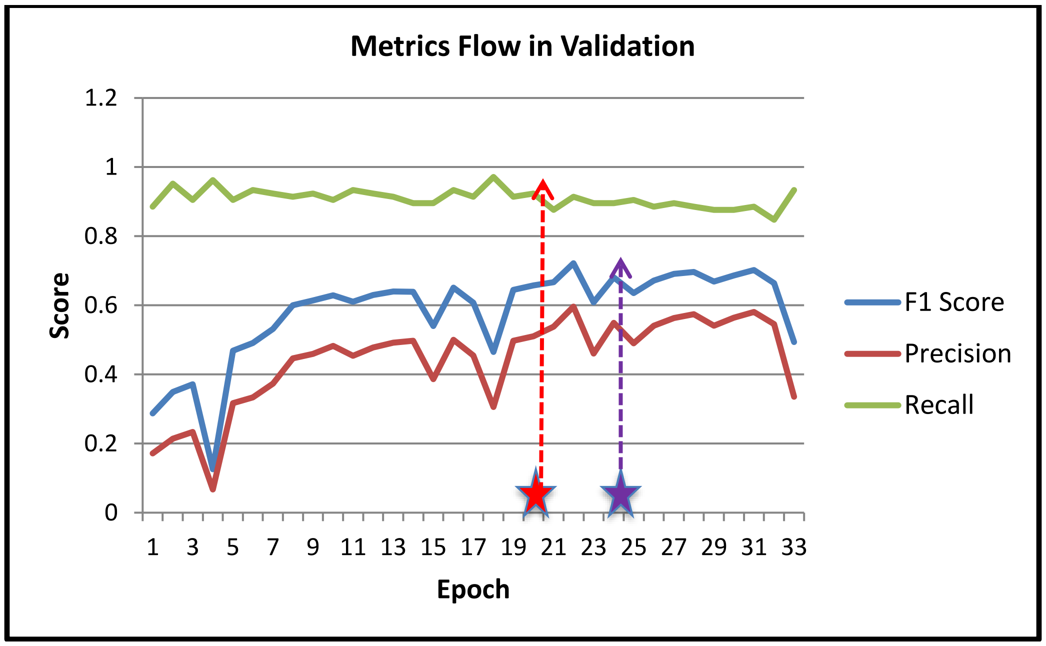


Fig. 12: Validation Metrics Flow

b) Scores

★ Best Recall Overall at Epoch no 18

★ Best Precision Overall at Epoch no 22

After 18 Epochs: Best Validation Recall

	Training			Validation			Test		
Class Description	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %
Handgun	52.61	99.85	68.91	40.54	100	57.69	62.5	100	76.92
Knife	53.98	100	70.11	68.18	100	81.08	62.5	100	76.92
Hard Disk	35.16	100	52.04	34.09	100	50.84	35.71	100	52.63
USB	43.38	100	60.51	38.71	80	52.17	50	80	61.53
Shuriken	56.65	100	72.33	65.21	100	78.95	50	80	61.53
Battery	62.72	100	77.09	65.21	100	78.95	76.92	100	86.95
Phone	52.7	100	69.02	37.5	100	54.54	43.48	100	60.6
Average	51.12	99.91	67.63	46.36	97.14	62.76	51.56	94.28	66.67

Table 9. Result on best recall epoch

After 22 Epochs: Best Validation Precision

	Training			Validation			Test		
Class Description	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %
Handgun	96.95	100	98.45	100	100	100	100	100	100
Knife	95.55	100	97.72	93.75	100	96.77	100	90	94.74
Hard Disk	98.19	100	99.09	75	100	85.71	100	100	100
USB	52.98	100	69.26	70	46.67	56	80	40	53.33
Shuriken	90.1	100	94.78	82.35	93.33	87.5	57.14	80	66.67
Battery	91.88	100	95.77	78.95	100	88.23	76.92	100	86.96
Phone	98.76	100	99.37	78.95	100	88.23	66.67	100	80
Average	94.02	100	96.92	82.75	91.42	86.87	80.26	87.14	83.56

Table 10. Result on best precision epoch

Note: NMS Threshold reduced to 0.0005 from 0.4

Sample Output Images

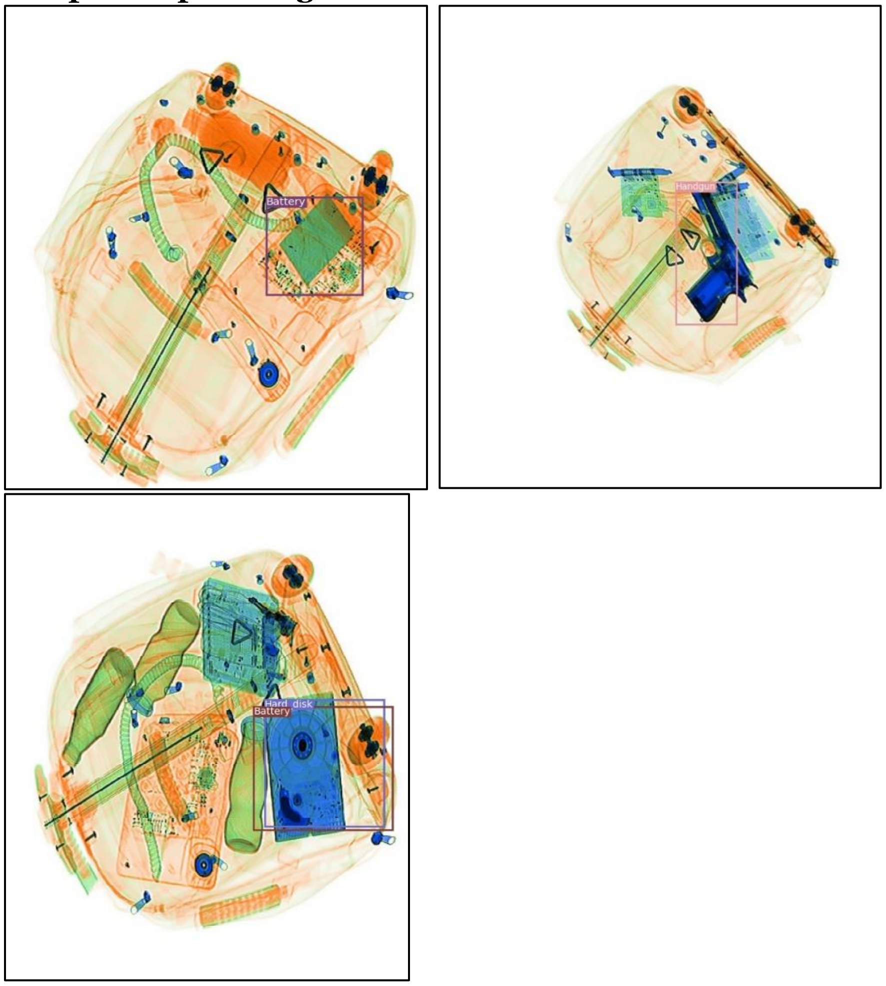


Fig. 13: Sample Output Images

Appendix 7: A novel GRU-driven stochastic degradation process for battery forecasting

Zihan Zhang

Background

In recent decades, there has been significant growth in the development of rechargeable battery-powered devices, such as electric vehicles, leading to a huge demand for batteries with high reliability and quality. End of life (EoL) is a critical indicator of battery health; it can be estimated by either adaptive stochastic processes or advanced machine learning techniques. However, these approaches either follow the degradation path having a specific form or lack stochastic interpretation due to its black-box nature. To address these challenges, a GRU-driven stochastic degradation process is proposed that can formulate battery degradation, in which drift fluctuation is controlled by a recursive Gaussian distribution with its mean learnt from a GRU-driven degradation pattern. Due to the non-Markovian state transitions, a sampling-based expectation maximization algorithm is developed to estimate model parameters based on historical observations. To validate the superiority of the proposed methods, a case study of NASA battery data was implemented. The results show a better performance with respect to EoL accuracy than that achieved with traditional methods.

Introduction

Due to their high energy, high power density, and long cycle life, Li-ion batteries are widely used, for example, in electric vehicles (EVs) and hybrid EVs. Because battery reliability is essential to the safe and reliable operation of a vehicle, increasing efforts have been invested in enhancing battery health management. Typically, battery states, as described by state-of-health (SoH) series, degrade over increasing numbers of charge and discharge cycles. However, as battery ages, battery SoH forecasting becomes challenging, owing to complicated degradation mechanisms and varying operating conditions.

Literature Review

Model-based and data-driven forecasting methods are the conventional approaches in battery prognostic problems. Model-based capacity forecasting method usually refers to electrochemical model, equivalent circuit model, empirical model and stochastic model. Electrochemical model and equivalent circuit model can reveal the physical or chemical dynamics of the battery properties by exploring complex aging mechanisms, leading to rapid advances in theoretical research and battery design. However, the high cost of measuring the internal degradation parameters makes their practical application infeasible. In contrast, empirical models may be useful, because the mathematical forms of the degradation trajectory can be mined from historical battery degradation processes, such as linear model, exponential model, polynomial model and Verhulst model. Similarly, stochastic models can capture the degradation uncertainty based on nonlinear Wiener processes with specific drift terms, such as linear and exponential. Furthermore, advanced filtering techniques, such as Kalman filter, extended Kalman filter (EKF), unscented Kalman filter (UKF) and particle filter (PF), can help either empirical or stochastic models reduce impacts of measurement noise and external disturbance to further increase forecasting accuracy. However, without prior knowledge of the underlying degradation mechanisms and limited by specific expressions of degradation path (i.e., mathematical form in empirical models and drift terms in stochastic models), it is difficult to generalize them to batteries of diverse types.

Some alternative methods to model-based approaches have been investigated, but none can reliably predict battery capacity degradation. Data-driven capacity forecasting methods rely only on historical measurements to establish a degradation model, without prior knowledge of inherent degradation behavior, which is more suitable for complicated prognostics issues. Optimization methods for data-driven models integrating kernel techniques are popular for improving the prediction efficiency, such as support vector machine (SVM), relevant vector machine (RVM) and Gaussian process regression (GPR). But it is difficult to generalize them due to their sensitive kernel parameters. To overcome this drawback, neural network-based methods have been implemented, such as autoregressive model, Elman neural network and temporal convolutional network. But their accuracy is limited to short-data windows. Recurrent neural networks (RNNs) can model long-range dependencies, but it is hard to implement back propagation in practice because repeatedly applying the squashing nonlinear activation function causes the model to predict that capacity will decay exponentially over time.

More practically, Zhang et al. adopted long short-term memory (LSTM) recurrent neural network to capture long-term degradation trends and predict battery end of life (EoL), in which an extra memory cell is introduced to solve the vanishing gradient problem, and he used a dropout method to mitigate model overfitting. Besides, gated recurrent unit (GRU) achieves faster operation for real-time prediction with less training parameters than LSTM. Although GRU/LSTM exhibit better performance in learning long-term dependence than RNN, there is no accurate and general battery degradation model that not only describes stochastic degradation characteristics but also tracks underlying degradation evolution via mining long-term dependencies among capacity series.

Contributions

To close the gap, a GRU-driven stochastic degradation process is proposed for tracking battery degradation with age. Specifically, the hidden state of the GRU is introduced into the Wiener degradation process as a pattern term for capturing long dependencies in historical drifts. Then, a differential function maps the pattern term to the mean of the drift space. Thus, drift is updated based on the entire degradation history instead of only the last time step. However, in parameter estimation, using the traditional expectation maximization (EM) algorithm becomes infeasible due to the non-Markovian drift transition. Thus, a sampling-based EM (SEM) algorithm is developed to handle the no closed form drift transition. A sampling step using Markov Chain Monte Carlo (MCMC) is added between the conventional E- and M-steps to obtain posterior samples of drifts. Finally, a forecasting algorithm is proposed to estimate battery SoH and EoL following the generative process of predicting degradation trajectory.

To summarize, this project makes the following contributions to current battery prognostics:

1) A GRU-driven nonlinear degradation pattern is designed to control the evolution of battery degradation;
2) Drift is updated recursively from a Gaussian distribution whose mean is mapped from a pattern term rather than the value in the last times-step.
3) A novel GRU-driven stochastic degradation model is proposed to capture battery degradation path, which combines interpretability of stochastic process and flexibility of neural networks.
4) A sampling-based expectation maximization algorithm is developed to estimate parameters of the proposed model by learning posterior drift samples drawn via the MCMC algorithm.