Engineering and Technology Journal e-ISSN: 2456-3358
Volume 07 Issue 10 October-2022, Page No.-1597-1604
DOI: 10.47191/etj/v7i10.07, I.F. – 6.484
© 2022, ETJ

Designing Frameworks for Reliability in Deep Learning Systems

AYSE ARSLAN

ABSTRACT: There has been a great amount of progress in deep learning models in the last decade. Such models are most accurate when applied to test data drawn from the same distribution as their training set. However, in practice, the data confronting models in real-world settings rarely match the training distribution.
This study explores the use of co-design approaches for developing reliable design frameworks for deep learning systems. It aims to raise awareness on how to develop reliable ML models within the context of recommender systems. While much work needs to be done in this field, the study provides suggestions and practical tips for how to develop reliable ML models such as in the case of recommender systems.

INTRODUCTION

There has been a great amount of progress in deep learning models in the last decade. While the efforts of the AI (artificial intelligence) community to bring openness and transparency to AI research, the inherent challenges of the field remain unchanged. For instance, when it comes to testing and experimentation, the data confronting models in real-world settings rarely match the training distribution.

One problem that remains to be solved is how to make design judgments given the huge costs of running the models.
The study starts with a brief overview of AI and co-design approaches. Next, it sheds light onto model architecture, especially for ranking and recommender models as the practical tips and suggestions relate to the context of a recommender platform. By doing so, it aims to raise awareness on how to develop reliable ML models.
Although the study provides an architecture and model recommendations specifically for a recommender system, the main points should be relevant for any other ML-driven model as well.

Review of Existing Studies

Design involves making good judgments in pursuit of desirable design outcomes. Design judgments do not follow a formal linear or rule-based process, yet emerge depending on the designer's experiences and the contextual aspects of the design situation (Dunne, 1999; Nelson & Stolterman, 2014).
When it comes to making design judgments for machine learning (ML), the process can become more challenging. The generic definition of "an AI system" is a single unified software system that can satisfy the following criteria:

- To reliably pass the adversarial Turing test in which the human judges are instructed to ask interesting and difficult questions, designed to advantage human participants, and to successfully unmask the computer as an impostor.
- To have general robotic capabilities such as being able to autonomously satisfactorily assemble a sophisticated model when equipped with appropriate tools and when given human-readable instructions,
- To get top-1 strict accuracy of at least 90.0% on interview-level problems in a benchmark set as introduced by Dan Hendrycks, Steven Basart et al.

"Unified" means that the system is integrated enough that it can, for example, explain its reasoning on a Q&A task, or verbally report its progress and identify objects during model assembly.
It should also be noted that agents play a crucial role for any ML system. The central feature of agency is that agents are systems whose outputs are moved by reasons (Dennett, 1987). An agent chooses a particular action because it "expects it" to deliver a desirable certain outcome. Main characterizations of agents include:

• The intentional stance: An agent's behavior can be usefully understood as trying to optimize an objective (Dennett, 1987).
• Cybernetics: An agent's behavior adapts to achieve an objective (e.g. Ashby, 1956; Wiener, 1961).
• Decision theory / game theory / economics / AI: An agent selects a policy to optimize an objective.
• An agent is a system whose behavior can be compressed with respect to an objective function (Orseau et al., 2018).
• "An optimizing system is ... a part of the universe [that] moves predictably towards a small set of target configurations" (Flint, 2020).

• A goal-directed system has self-awareness, planning, consequentialism, scale, coherence, and flexibility (Ngo, 2020).
• Agents are ways for the future to influence the past (via the agent's model of the future) (Garrabrant, 2021; von Foerster et al., 1951).
Within this light of information, the system of the RL training process might be an agent, but the learnt RL policy itself, in general, won't be an agent as after training.

Despite their agency, ML models can still become prone to adversarial attacks, therefore there is a need to determine how to prevent adversarial attacks and to develop reliable models. Nelson and Stolterman (2014) called this prioritization appreciative judgment.
On the other hand, instrumental judgment (Nelson & Stolterman, 2014) includes determining when and the duration in which different tools are used during the design process (Gray & Boling, 2018). The aim is to bring elements of a design into a unified whole while connective judgments bind together different elements of a design situation (Gray et al., 2015; Nelson & Stolterman, 2014).

As design is highly dependent on context, navigational judgment assists designers in adjusting their approach to the changing situational realities of a design situation. Designers use navigational judgment as a tool to deal with unpredictable situations such as adversarial attacks in ML models (Nelson & Stolterman, 2014).
There are also quality judgments made based on the interplay between designer's personal preferences, external standards, and the uniqueness of the design context (Nelson & Stolterman, 2014). They can also include the stylistic feel of a designed artifact, including such ideas as "form, occurrence, essence, and excellence" (Nelson & Stolterman, 2014, p. 151).
When making design judgments on ML models, what would be needed to either complement or supplant example-driven trained neural network systems is the so called common sense, which refers to the ability to see things that to many human-beings seem obvious, to draw fast and simple, obvious conclusions (Brachmann, 2005).

Traditional AI models are trained to optimize decisions to achieve the optimal return. To achieve this within a large number of domains including recommender systems, the model should learn many domain-specific sub-tasks (e.g., filtering different kinds of noise or focusing on a specific detail), which can only be learned from a semantically diverse dataset. Therefore, it may not always be easy to evaluate a model. There are three reasons why model evaluation plays a crucial role:
1. To estimate the generalization accuracy, the predictive performance of a model on future (unseen) data.
2. To increase the predictive performance by tweaking the learning algorithm and selecting the best-performing model from a given hypothesis space.
3. To identify the machine learning algorithm that is best-suited for the problem at hand. Hence, we want to compare different algorithms, selecting the best-performing one as well as the best-performing model from the algorithm's hypothesis space.
One of the main techniques include the re-substitution evaluation and holdout method The holdout method is the simplest model evaluation technique; it can be summarized as follows.

- First, developers take a labeled dataset and split it into two parts: A training and a test set.
- Then, they fit a model to the training data and predict the labels of the test set. The learning algorithm builds a model from the training set of labeled observations.
- Then, they evaluate the predictive performance of the model on an independent test set that shall represent new, unseen data.
- The fraction of correct predictions, which can be computed by comparing the predicted labels to the ground truth labels of the test set, constitutes the estimate of the model's prediction accuracy. This helps them to deal with real world limitations such as limited access to new, labeled data for model evaluation.

Nevertheless, such model estimates may suffer from bias and variance. For instance, the re-substitution evaluation (fitting a model to a training set and using the same training set for model evaluation) is heavily optimistically biased.
To overcome such biases, there are also some more advanced techniques for model evaluation. The bootstrap method is a resampling technique for estimating a sampling distribution, and the uncertainty of a performance estimate – the prediction accuracy or error.
Another technique is the use of k-nearest neighbors algorithm. A k-nearest neighbors model literally stores or memorizes the training data and uses it only at prediction time..
In contrast to k-nearest neighbors, a simple example of a parametric method is logistic regression, a generalized linear model with a fixed number of model parameters: a weight coefficient for each feature variable in the dataset plus a bias (or intercept) unit.
The process of finding the best-performing model from a set of models that were produced by different hyperparameter settings is called model selection. While the learning algorithm optimizes an objective function on the training set the aim is to optimize a performance metric such as classification accuracy.
One of the probably most common technique for model evaluation is k-fold cross-validation. Here, the main idea behind cross-validation is that each sample in a dataset has the opportunity of being tested. k-fold cross-validation is a

special case of cross-validation where one iterates over a dataset set k times. In each round, one splits the dataset into k parts: one part is used for validation, and the remaining $k - 1$ parts are merged into a training subset for model evaluation.
In contrast to the repeated holdout method, test folds in k-fold cross-validation are not overlapping. In repeated holdout, the repeated use of samples for testing results in performance estimates that become dependent between rounds which can be problematic for statistical comparisons.

Although, developers may prefer simpler models for several reasons, Domingos made a good point regarding the performance of "complex" models. As he mentions in his article, "Ten Myths About Machine Learning:"
"Simpler models are preferable because they're easier to understand, remember, and reason with. Sometimes the simplest hypothesis consistent with the data is less accurate for prediction than a more complicated one."
Using these procedures, one has to bear in mind that the aim is then to not compare between models yet different algorithms that produce different models on the training folds. As Gael Varoquaux [Varoquaux, 2017] writes, cross-validation is not a silver bullet. However, it is the best tool available, because it is the only non-parametric method to test for model generalization.
One of the main optimization methods is hyperparameter optimization which aims at finding
theright hyperparameters for a machine learning task by keeping track of use patterns in a database usually consisting of optimization trajectories. A Transformer-based framework for hyperparameter tuning, based on large-scale optimization data using flexible text-based representations can be used.
An example of transformer neural networks would be large language models often spanning across billions of parameters and trained on gigabytes of text data which can be used for a wide range of tasks, including text generation, question answering, summarization, and more.

Another technique for resource optimization would be the use of restless-multi-armed bandits (RMABS). An RMAB consists of n arms where each arm (representing a beneficiary) is associated with a two-state Markov decision process (MDP).

In general, the focus in ML community has been so far on Markov reward functions; for instance, given a state space that is sufficient to form a task such as (x,y) pairs in a grid world, is there a reward function that only depends on this same state space that can capture the task?

Regardless of the techniques used, there should be an alignment between these models and human values. At a high-level, the main approach to alignment focuses on engineering a scalable training signal for very smart AI systems that is aligned with human intent. It has three main pillars:

- Training AI systems using human feedback: RL from human feedback is the main technique for aligning mostly deployed language models. The aim is to train a class of models derived from pre-trained language models so that they are trained to follow human intent: both explicit intent given by an instruction as well as implicit intent such as truthfulness, fairness, and safety.
- Training AI systems to assist human evaluation: In order to scale alignment, engineers prefer to use techniques like recursive reward modeling (RRM), debate, and iterated amplification. The ultimate aim is to train models to assist humans to distinguish correct from misleading or deceptive solutions.
- Training AI systems to do alignment research: As there is indefinitely no scalable solution a more pragmatic approach might be building and aligning a system that can make faster and better alignment research progress than human-beings can.

The ultimate goal is to train models to be so aligned that they can off-load almost all of the cognitive labor required for alignment research.
Importantly, there is a need for "narrower" AI systems that have human-level capabilities in the relevant domains to do as well as human-beings on alignment research.
There are various other techniques to update model parameters, yet given the scope of this study, the next section will explain related techniques for developing recommender systems.

RECOMMENDER SYSTEMS

The main goal of a recommender system is to produce features from both videos and text (i.e., the user question), jointly allowing their corresponding inputs to interact.
One main challenge for recommender systems relates to the use of language models for ML as they may not be inherently grounded in the physical world due to the lack of interaction during the training process.

Recently, researchers seek to combine advanced language models with learning algorithms while grounding the language model within a specific real-world context. This model determines the probability of a specific skill for completing the instruction by multiplying two probabilities: (1) task-grounding (i.e., a skill language description) and (2) world-grounding (i.e., skill feasibility in the current state).
Another challenge is embedding videos into deep learning models such as recommender systems which require more sophisticated solutions such as objects in a scene, as well as temporal information, e.g., how things move and interact, both of which must be taken in the context of a natural-language question that holds specific intent.

Online video sharing platforms need to understand perceptual video quality (i.e., a user's subjective perception of video quality) in order to better optimize and improve user experience.

To improve the quality, one can collect ground-truth labels from the platform dataset and categorize factors that affect quality perception into three high-level categories: (1) content, (2) distortions, and (3) compression.

Another approach is to train a model from scratch on existing quality datasets. However, this may not be feasible as there are limited quality datasets. To overcome this limitation, one can apply a self-supervised learning step to the model during training to learn comprehensive quality-related features, without ground-truth derived from raw videos.

When it comes to using videos, some ML-powered features may rely on leveraging neural network sparsity, a common solution that works across devices, from entry level computers to high-end workstations. However, mid-tier and high-end devices often have powerful GPUs that remain untapped for ML inference. Therefore, some related design decisions regarding ML-drive video platforms include, but are not limited to:

- Backbone: To compare several widely-used backbones for on-device networks to be a better fit for the GPU by removing inefficient blocks
- Decoder: To switch to a multi-layer perceptron (MLP) decoder consisting of 1x1 convolutions instead of using simple bilinear up-sampling
- Model size: To develop a larger model without sacrificing the real-time frame rate necessary for smooth video segmentation

One common challenge for web-based inference is that web technologies can incur a performance penalty when compared to apps running natively on-device.

Moreover, security and privacy threats of ML models should also be taken into account as adversaries can stage effective attacks against these systems and potentially obtain sensitive information used in training the models. Rewards cannot act as a prediction label as they don't represent any labels.

During training, ML models go through episodes, each of which consists of a trajectory or sequence of actions and states. The training and output trajectories can be used by an attack trainer that trains an ML classifier to detect input trajectories that were used in the target RL model's training.

Given these security challenges, developing a multi-domain/task model can be both tedious and challenging, therefore, the aim should be to:

1) achieve high accuracy efficiently (keeping the number of parameters low),

2) learn to enhance positive knowledge transfer while mitigating negative transfer, and

3) effectively optimize the joint model while handling various domain-specific difficulties.

In this way, a multi-path network is learned from neural architecture search by applying one reinforcement learning controller for each domain to select the best path in the super-network created from a search space.

A more effective way for automatically designing deep learning architectures could be to define a search space, made up of various potential building blocks that could be part of the final model. In this way, the search algorithm finds the best candidate architecture from the search space that optimizes the model objectives, e.g., classification accuracy.

Ranking is a core problem across a variety of domains, such as search engines, recommender systems, or question answering. Some designs can also introduce noise in the ranking scores which causes the loss to sample many different rankings that may incur a non-zero cost.

Within the light of this information, a model architecture can be developed in two stages:

- Search: In the search stage, to find an optimal path for each domain jointly, an individual reinforcement learning (RL) controller is created for each domain, which samples an end-to-end path (from input layer to output layer).

  At the end of the search stage, all the sub-networks are combined to build a heterogeneous architecture for the model.
- Training: For this to work, it is necessary to define a unified objective function for all the domains. An algorithm that adapts throughout the learning process should be designed such that losses are balanced across domains.

This is an efficient solution to build a heterogeneous network to address the data imbalance, domain diversity, negative transfer, domain scalability, and large search space of possible parameter sharing strategies for machine learning.

DESIGN METHODOLOGY

Using the co-design approach, this study explores how to understand the reliability of a model in novel scenarios. It suggests three general categories of requirements for designing reliable machine learning (ML) systems:

(1) They should accurately report uncertainty about their predictions ("know what they don't know");

(2) They should generalize robustly to new scenarios (distribution shift); and

(3) They should be able to efficiently adapt to new data (adaptation). Importantly, a reliable model should aim to do well in all of these areas simultaneously out-of-the-box, without requiring any customization for individual tasks.

Co-design generally refers to the collaboration between researchers and users to produce digital artefacts (Barbera et al., 2017; Cober et al., 2015). The co-design process differs from other methods of design in that it operates "bottom-up" with users being active participants in the design process, who provide critical insight into daily work practices and the existing context. Co-design is then defined as:

"a highly facilitated, team-based process in which users, researchers, and developers work together in defined roles to design an innovation, realize the design in one or more prototypes, and evaluate each prototype's significance for addressing a concrete need." (Roschelle et al., 2006)

Co-design has several advantages and is generally beneficial for both researchers and users. Researchers gain insight into the user contexts in which their tools will be deployed (Cober et al., 2015). This in turn helps to ensure that the environments or tools being designed to meet the needs and goals of the user and increases the likelihood of continuous implementation and use of the new technologies.

Despite its promise, problems emerge within the co-design process. For instance, a user may have difficulties expressing their ideas in a non-ambiguous, highly formalized way that computer scientists can convert into products.

The process model of a co-design session is divided into three phases: establishing context, design, and presentation.

Phase One: Establishing Context

The objective of this stage is to create a shared understanding of aspects such as content, resources, and challenges of the recommender system to be developed.

Phase Two: Design

Individual team members are encouraged to make proposals, adapt, or expand on ideas and ask questions. A common trend among both high- or low-structured approaches to co-design is the use of physical artifacts to support a shared understanding of the design (Barbera et al., 2017).

Phase Three: Presentation and Documentation

During this process, the researchers document, both visually and in text form, the design decisions that are agreed on by the group. At the end of stage three, an initial design has been created, agreed upon, and translated into a graphical format, which can be clearly interpreted by all parties involved.

RECOMMENDATIONS

When it comes to designing ML models such as recommender systems, the following suggestions can be taken into account:

1. Defining the goal:

Goals can include:

- user retention,
- increased revenue,
- cost reduction.

If the global task of a recommender system is to select a shortlist of content from a large catalog one choice might be to focus on the history of the user's interaction with the service. Yet, "good recommendations" from a user perspective and from a business perspective are not always the same thing.

2. FINDING THE OPTIMAL USER TOUCHPOINT:

When a decision has been made on the global goal, the best way to display recommendations need to be made. For instance, in the case of recommender systems, display of recommendations can occur:

- in the feed
- push notifications,
- email newsletter,
- the section with personalized offers in a personal account,
- or other sections on the site/application.

Many factors influence the choice of touchpoint such as push notifications or the complexity of integration with ML microservice.

As ML should ideally be implemented when it is seen by the maximum number of users,

using ML at this point would be impractical. The main rule here is to integrate ML where it will make the biggest increase in business metrics.

3. COLLECTING DIVERSE FEEDBACK:

Feedback is the actions a user can take to demonstrate how they feel about the content. To build a recommender system, there is a need to learn how to collect different types of feedback:

Explicit:

This can be a rating by any scale or a like/dislike.

I**mplicit**:

This can include;

- the amount of time a user spends on the content,
- the number of visits to the content page,
- the number of times one shares the content on social networks or sends it to friends.

Feedback should correlate with the business goals of the recommender system.

Some important technical points to consider are:

- Expanding user feedback channels: In addition to the time spent on the page, one can start collecting user comments and determining their tone.
- Keeping a history of user feedback: This helps one to identify insights in long-term users' behavior. Also, a large amount of historical data will allow one to compare models without running AB tests, in an offline format.
- Collecting data on all platforms

4. DEFINING BUSINESS METRICS:

ML experts got used to working with the metrics of ML algorithms: precision, recall, etc. businesses might be interested in other indicators such as:

- session depth,
- conversion to click//view,
- retention,

- average click per user
-

5. SEGMENTING USERS:

From a business perspective, the audience of the site can be very heterogeneous in various ways including;

- socio-demographic characteristics,
- activity on the service (number of feedback, frequency of visits),
- geo-positions,

The system should provide the ability to calculate metrics in different user sections, to notice the improvement (or deterioration) of metrics in each particular segment. For example, there can be two large segments:

- "high activity" — users visit the app frequently and watch a lot of content,
- "low activity" — users visit the app rarely.

6. DETERMINING THE RIGHT OFFLINE METRICS:

When the feedback data is collected and the business metrics are selected, there is a choice of offline metrics, which can optimize a model, such as precision@k or recall@k,

One can choose offline metrics that correlate with business metrics by calculating the correlation between offline and online metrics.

7. CREATING A BASELINE MODEL:

Rather than trying to use the most complex models to solve the problem one can start with simpler approaches instead of neural networks. This simple model is called a baseline.

For example, one can consider using a simple approach based on the k-Nearest neighborhoods algorithm to create a service for recommending content in push notifications, and only in the second iteration, move to a more complex boosting model.

8. Choosing the ML Algorithm and Discard the Worst Models:

The next step is to train more complex models. Recommender systems usually use both neural networks and classical ML algorithms:

- Matrix factorization,
- Logistic Regression,
- KNN (user-based, item-based),
- Boosting.

One can also prefer to count offline metrics already accumulated in the feedback system. In this way, one can distinguish very bad models from "not quite bad ones" in order to test the "not quite bad model".

9. RUN EVERYTHING THROUGH THE AB TESTING SYSTEM:

Any changes in the recommender algorithm, such as switching from a baseline to an advanced model, must go through a system of AB tests. Without good analytics, one either can't see the effect of a recommender system, or one can misinterpret the data, which can cause business metrics to deteriorate.

This is why AB tests need to measure both short-term and long-term effects. When conducting AB tests, one need to ensure that the samples in the test and control groups are representative.

10. REMEMBER THE CLASSIC PROBLEMS IN PRODUCTION:

When rolling out the algorithm "in production" it is necessary to provide a solution to a number of classic problems.

- Users' cold start: What to recommend to those who haven't left feedback? One can make lists of globally popular content and make them as diverse as possible to be more likely to "hook" the user.
- A feedback loop: One can show the content to the user, then collect feedback, and run the next learning cycle on that data. In this case, the system learns from the data it generates itself. To avoid this trap, one can usually allocate a small percentage of users who receive random output instead of recommendations — with this design, the system will be trained not only on its own data but also on users' interactions with randomly selected content.

Each of these suggestions might have their own advantages and dis-advantages. Based on the context and requirements of the model to be developed, relevant methods can be selected. It should be taken into account that there is no single one, best approach that would work for all models.

CONCLUSION

Design is a complex process and designing for ML models can certainly be much more complex. Despite the progress made in the field and the rise of large-scale pre-training models, the data confronting models in real-world settings rarely match the training distribution which can ultimately affect the reliability of a model.

This study explored the use of co-design approaches for developing reliable design frameworks for deep learning in the context of recommender systems. It aims to raise awareness on how to develop reliable ML models.

While much work needs to be done in this field, the study provides some practical steps to be followed when it comes to developing reliable models. Although the study provided an architecture and model recommendations for a recommender system, the main points should be relevant for any other ML-driven model as well.

REFERENCES

1. Assuncao, Marcos D., et al. "Big Data Computing and Clouds: Challenges, Solutions, and Future

Directions."Journal of Parallel and Distributed Computing, vol. 79–80, Dec. 2013, https://doi.org/10.1016/j.jpdc.2014.08.003.

2. Athanases, Steven Z., et al. "Fostering Data Literacy through Preservice Teacher Inquiry in English Language Arts." The Teacher Educator, vol. 48, no. 1, 2013, pp. 8–28.
3. Bargagliotti, Anna, et al. Pre-K-12 Guidelines for Assessment and Instruction in Statistics Education II (GAISE II): A Framework for Statistics and Data Science Education. American Statistical Association,2020,https://www.amstat.org/docs/default-source/amstat-documents/gaiseiiprek-12_full.pdf.
4. Bersin Insights Team. Insights from IMPACT 2018. Deloitte Development LLC, 2018, https://www2.deloitte.com/content/dam/Deloitte/ca/Documents/audit/ca-audit-abm-scotia-insightsfromimpact-2018.pdf.
5. Bhargava, Rahul, et al. "Beyond Data Literacy: Reinventing Community Engagement and Empowerment in the Age of Data." Data-Pop Alliance, Data-Pop Alliance, Nov. 2015, https://datapopalliance.org/item/beyonddata-literacy-reinventing-community-engagement-and-empowerment-in-the-age-of-data.
6. Cowell, Matt. "A Roadmap for Creating a Data Literacy Program - QuantHub." QuantHub, QuantHub, 18 June 2020, https://quanthub.com/data-literacy-program. DataLiteracy. "The Data Literacy Score | Data Literacy."
7. Data Literacy | Learn the Language of Data, Feb. 2021, https://dataliteracy.com/data-literacy-score. ---. "About This Project - DataLiteracy.Ca." DataLiteracy.Ca, Intyernet Archive, 22 Dec. 2021, https://web.archive.org/web/20211222131031/http:/dataliteracy.ca/about-this-data-literacy-project.
8. Deahl, Erica. Better the Data You Know: Developing Youth Data Literacy in Schools and Informal Learning Environments. Massachusetts Institute of Technology, 2007, https://cmsw.mit.edu/wp/wpcontent/uploads/2016/06/233823808-Erica-Deahl-Better-the-Data-You-Know-Developing-Youth-DataLiteracy-in-Schools-and-Informal-Learning-Environments.pdf.
9. Duncan, Alan D., Donna Medeiros, Aron Clarke, et al. "How to Measure the Value of Data Literacy." Gartner, Gartner, Apr. 2022, https://www.gartner.com/en/documents/4003941. ---. "Toolkit: Data Literacy Individual Assessment." Gartner, Gartner, Apr. 2020, https://www.gartner.com/en/documents/3983897.
10. Ebbeler, Johanna, et al. "Effects of a Data Use Intervention on Educators' Use of Knowledge and Skills." Studies in Educational Evaluation, vol. 48, Mar. 2016, pp. 19–31, https://doi.org/10.1016/j.stueduc.2015.11.002.
11. Kleitz, Lauren, and Joe Shelley. EDUCAUSE Data Literacy Institute. EDUCAUSE, 24 Jan. 2022, https://events.educause.edu/educause-institute/data-literacy-institute/2022/online-1.
12. Kolbe, Richard H., and Melissa S. Burnett. "Content-Analysis Research: An Examination of Applications with Directives for Improving Research Reliability and Objectivity." Journal of Consumer Research, vol. 18, no. 2, Sept. 1991, pp. 243–50, https://doi.org/10.1086/209256.
13. Koltay, Tibor. "Data Governance, Data Literacy and the Management of Data Quality." IFLA Journal, vol. 42, no. 4, Nov. 2016, pp. 303–12, https://doi.org/10.1177/0340035216672238.
14. Kumar, Vivekanandan, and David Boulanger. "Explainable Automated Essay Scoring: Deep Learning Really Has Pedagogical Value." Frontiers in Education, Oct. 2020, https://doi.org/10.3389/feduc.2020.572367.
15. Linn, Robert L., and David M. Miller. Measurement and Assessment in Teaching, 9th Edition. Pearson, 2005,https://www.pearson.com/us/highereducation/product/Linn-Measurement-and-Assessment-in-Teaching9th-Edition/9780131137721.html.
16. Lorente, Clara Llebot. LibGuides: Research Data Services: Data Types & File Formats. Oregon State University, June 2021, https://guides.library.oregonstate.edu/research-data-services/data-managementtypes-formats.
17. Means, Barbara, et al. Teachers' Ability to Use Data to Inform Instruction: Challenges and Supports. US Department of Education, Office of Planning, Evaluation and Policy Development, Feb. 2011, https://eric.ed.gov/?id=ED516494.
18. National Defence. The Department of National Defence and Canadian Armed Forces Data Strategy - Canada.Ca. Government of Canada, 3 Dec. 2019, https://www.canada.ca/en/departmentnationaldefence/corporate/reports-publications/data-strategy.html.
19. Nikou, Melpomeni, and Panagiotis Tziachris. "Prediction and Uncertainty Capabilities of Quantile Regression Forests in Estimating Spatial Distribution of Soil Organic Matter." ISPRS International Journal of GeoInformation, vol. 11, no. 2, Feb. 2022, p. 130, https://doi.org/10.3390/ijgi1102013

20. Salleh, Kahirol Mohd, et al. "Competency of Adult Learners in Learning: Application of the Iceberg Competency Model." Procedia - Social and Behavioral Sciences, vol. 204, Aug. 2015, pp. 326–34, https://doi.org/10.1016/j.sbspro.2015.08.160.
21. Samuel, A. L. "Some Studies in Machine Learning Using the Game of Checkers." IBM Journal of Research and Development, vol. 3, no. 3, July 1959, pp. 210–29, https://doi.org/10.1147/rd.33.0210.
22. Schield, Milo. "Information Literacy, Statistical Literacy and Data Literacy." IASSIST Quarterly, 2004, http://www.statlit.org/pdf/2004-Schield-IASSIST.pdf.
23. Walsh, John. Implementing DND/CAF Data Strategy. Government of Canada, 26 Sept. 2021, https://publicsectornetwork.co/wp-content/uploads/2021/09/John-Walsh-PDF.pdf.
24. Wells, Dave. Building a Data Literacy Program. 4 Jan. 2021, https://www.eckerson.com/articles/the-dataliteracy-imperative-part-i-building-a-data-literacy-program.
25. Williams, Paula G., et al. "On the Validity of Self-Report Assessment of Cognitive Abilities: Attentional Control Scale Associations With Cognitive Performance, Emotional Adjustment, and Personality." Psychological Assessment, vol. 29, no. 5, 2017, pp. 519–30.
26. Wolff, Annika, et al. "Creating an Understanding of Data Literacy for a Data-Driven Society." Journal of Community Informatics, vol. 12, no. 3, Aug. 2016, https://doi.org/10.15353/joci.v12i3.3275.
27. Yi Min Lim, Delia, et al. "Datastorming: Crafting Data into Design Materials for Design Students' Creative Data Literacy." C&C '21: Creativity and Cognition, Association for Computing Machinery, 2021, pp. 1–9, https://doi.org/10.1145/3450741.3465246