Deep learning challenge: Lessons from the field
Artificial intelligence (AI) is front and center of the technology news at present. In particular, machine learning (ML) and deep learning (DL) are receiving lots of attention because of numerous successful applications rolled out by technology titans such as Google, Facebook and Apple. Just a few examples include Google Translate, DeepFace from Facebook and Apple's virtual personal assistant, Siri.

The remarkable momentum that deep learning has achieved is a result of increased computing power, the availability of large datasets and recent advances in artificial neural networks. Increasingly, it is not only the large technology companies, but also smaller companies in other areas that have identified the potential, not just of traditional machine learning, but specifically of deep learning, to make them competitive and successful.

01/ Introduction

This whitepaper is for business stakeholders wishing to gain a sound understanding of the key challenges between conventional software engineering projects and those involving machine learning or deep learning. The paper will highlight 12 potential challenges that may affect AI projects within an organization.

In this paper, case studies are used to illustrate the potential of deep learning as a mainstream tool for a spectrum of businesses. The case studies describe the experiences of Peltarion's data scientists working in the field on customer projects. As real-life examples, they are used to illustrate the unique challenges of deep learning projects. In sharing them, we seek to use our prior experience to enable businesses to take advantage of deep learning technologies without falling victim to project pitfalls. Deep learning is still a nascent technology and organizations seeking to use it should be informed of these challenges in advance.

This paper is based on a recent academic publication, authored by some of the Peltarion team, and extended for a business audience.

---

02 / Machine learning and deep learning differences

It is worth taking a moment to note the difference between machine learning and deep learning before we compare them with traditional software engineering.

Machine learning (ML) uses algorithms to parse data and learn from it, then makes predictions based on that learning. A significant amount of manual effort is needed for “feature engineering,” i.e., to manually program how to transform and represent the data.

Deep learning (DL) is a subfield of machine learning, and is effectively an evolution of it: the machine can make decisions, thereby reducing the need for feature engineering by a programmer. Instead, it abstracts layers of algorithms to build an artificial neural network that learns and makes predictions automatically.

ML and DL are highly data driven. Both use data from the external world to identify patterns, in comparison to conventional software, which follows hard-coded rules. While common software engineering practices have shown the value of abstraction boundaries such as encapsulation and modular design, ML and DL systems can erode the boundaries because of their dependency on the external world.

Given the dependence of ML and DL on data, it seems logical that the data should be tested just as well as code, but there is a lack of best practices for how to do so today. In traditional software engineering terms, it may be possible to write tests that verify the functionality of code, but in deep learning, a code test will not identify “data bugs.”

As with any software project, ML and DL systems have the capacity to incur technical debt: the long-term costs of fast-moving software processes. These systems are also subject to data dependencies (sometimes known as “scientific debt”), which are difficult to evaluate and analyze, and impose an additional (maintenance) overhead.

In summary, a ML or DL system is fundamentally different from a traditional software project in that there is a high dependency on external data, and you leave much of the responsibility for finding a working solution to the model. These differences can severely impact predictability, transparency and reproducibility, which in turn can affect business value. The traditional tools and practices for reviewing and writing test code, and maintaining it against technical and scientific debt are not always applicable or sufficient for building production-ready ML or DL systems.
03 / The challenges of AI projects

In this section, we identify 12 key challenges specifically related to the intersection of software engineering practices and AI applications. The challenges are, in part, based on existing research and validated by Peltarion’s experiences in the field and have been identified and categorized into three areas: development, production and organization.

<table>
<thead>
<tr>
<th>Technology</th>
<th>People</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited transparency</td>
<td>Cultural differences</td>
<td>Effort estimation</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>Privacy and safety</td>
<td>Dependency management</td>
</tr>
<tr>
<td>Resource limitations</td>
<td>Unintentional feedback loops</td>
<td>Monitoring and logging</td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td>Glue code and supporting systems</td>
</tr>
<tr>
<td>Development challenges</td>
<td></td>
<td>Experiment management</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each challenge is assigned a rating in terms of the potential impact severity it may have on the business value of a project (for example, in terms of delivery schedule, overall quality and cost overheads). The challenge difficulty is also rated to indicate how easy it would be to solve for a typical software team with limited experience in data science/AI. While we recognize that none of these challenges are trivial, the difficulty rating is assigned low if we consider the challenge to be one a traditional software organization is already equipped to solve. The difficulty is indicated to be high when a high level of domain expertise in deep learning is required. At the time of writing, expert knowledge of this type is generally to be found only within a limited number of specialist organizations.

The ratings for these challenges should be treated as indicative and simply used to help organizations think and plan ahead. Each and every business faces its own particular circumstances, just as each and every AI project may have a range of specific and even unique circumstances, meaning that the impact of a challenge may vary accordingly.
As we have already noted, the nature of developing AI/DL projects is different compared to traditional software. To achieve industrial-grade quality and reliability requires a new generation of development tools and best practices.

During the development of DL models, a large number of experiments are usually performed to identify the optimal model. Version control is therefore important in order to be able to compare and reproduce experiments scientifically. However, versioning can be problematic because of the number of variables, or branches, in the system. For example:
- Hardware (e.g., GPU models)
- Platform (e.g., operating system and installed packages)
- Source code (e.g., model training and pre-processing)
- Configuration (e.g., model configuration and pre-processing settings)
- Data sources (e.g., input signals and target values)
- Training state (e.g., versions of trained model)

Furthermore, different versions of every model are created during training, each with different parameters and metrics that need to be properly measured and tracked.

**Potential impact severity:** High  
**Challenge difficulty:** Medium

A core principle of good software engineering is to reduce a complex system into smaller, simpler blocks. While deep learning systems essentially do that automatically, their complexity can make it very difficult to explain how the results are reached. This can cause problems in heavily regulated businesses, such as banking or healthcare, where transparency is very important, and it may lead to less efficient models being used because they are easier to explain.

**Potential impact severity:** Medium  
**Challenge difficulty:** Medium/High
Case study

Predicting Oil and Gas Recovery Potential
A study was initiated with Peltarion by a U.S. oil and gas exploration company. The project used deep learning and geological information to predict the estimated ultimate recovery (EUR) of oil or gas to be extracted from a well.

The data consisted of high-resolution geological maps combined with a small number of oil/gas wells that had been in production for a while (where the EUR was accurately known). The project resulted in a decision support tool that could, given a coordinate, predict the EUR.

However, the transparency challenge was evident for this project. A borehole costs > $1M so the results needed justification before very expensive engineering decisions could be made. The exploration engineers wanted to know how the neural network had made its decisions, but no direct simple explanation could be provided.

Troubleshooting

The complex inner workings of a neural network makes it difficult to debug in a traditional way. In addition, the use of distributed computing, external libraries such as TensorFlow or frameworks such as Apache Spark, make it difficult to debug problems that arise. Even if it was possible to step through source code, manual evaluation is practically impossible because it involves the inspection of millions of parameters. Thus, compared to traditional software engineering, a small bug may not be detected at compile time or run time.

Potential impact severity: Low/Medium
Challenge difficulty: Medium/High

Testing

As we have discussed, a key difference between AI systems and traditional software is that data partly replaces code and a learning algorithm is used to automatically identify patterns in the data. Ideally, to test the correctness of a model, we would compare the distribution of input data over time with the resultant output predictions. This is difficult in part because of the non-deterministic nature of many training algorithms and also because of the dependency on the external world, which can make it difficult to reproduce previous results and compare one distribution to another.

Given the high data dependency, it is also important that the data is tested. However, few data testing tools exist today, especially when compared to software testing.
Before deploying a production-ready AI system, a significant amount of "plumbing" code is needed to support it and the infrastructure it runs upon. A mature system may end up with 95 percent of the code being plumbing and "glue code," and this software also needs to be tested.

**Potential impact severity:** Low/Medium  
**Challenge difficulty:** Medium

### Case study: Weather forecasting

Weather forecasts today rely on physical modeling and are solved primarily using finite element methods. They require very expensive supercomputers that are slow and, for many applications, inaccurate. Peltarion, in collaboration with a national meteorological agency, aimed to use deep learning for weather forecasting in order to improve wind turbine generator predictions.

### Production challenges

The challenges described in this section are relevant to the maintenance of AI and DL systems once they enter production.

### Dependency management

Use of the newest hardware available is recommended, particularly for DL systems, to take advantage of advances in processing capabilities. However, this comes with associated engineering costs in terms of updating software to accommodate the hardware changes and then manage dependencies.

A machine learning system is, by definition, always open ended, as it is driven by external data. No matter how carefully you construct the model and test the code that defines it, its performance will always be heavily dependent on the external data, and if the method for generating incoming data changes, it can impact the model. Over time, data dependencies can add up to the same level of technical debt as code dependencies.

**Potential impact severity:** Low/Medium  
**Challenge difficulty:** High
Monitoring and logging

As the behavior of the external world changes, drifts may occur in the incoming data, causing the behavior of a live AI system to suddenly change. It may not be sufficient to monitor accuracy metrics of validation and test datasets. Metrics such as the distribution of the predictions and the input data from upstream data providers should also be monitored.

**Potential impact severity:** Medium  
**Challenge difficulty:** Medium

Unintentional feedback loops

In models deployed in a big data context (as AI systems often are), there is a risk of creating unintended feedback loops where, over time, your real-world system adapts to your model rather than the other way around. Consider a widely used real estate price prediction system which becomes popular. It is possible that, over time, the price predictions can easily become a self-fulfilling prophecy.

**Potential impact severity:** Medium/High  
**Challenge difficulty:** Medium

Glue code and supporting systems

As described in the section covering development challenges, there is a significant amount of "glue code" that interacts with supporting infrastructure, including code that interacts with external systems such as cloud services. Keeping the code up to date and aligned with changes introduced by external providers can introduce unexpected challenges in production.

**Potential impact severity:** Medium/High  
**Challenge difficulty:** High

Organizational challenges

The organizational approach to putting AI or DL models into production is somewhat different from that of a traditional software engineering project.
Effort estimation

It will always be hard to estimate the effort needed in an AI project. Although the goals of an AI project can be well defined, there is no way of guaranteeing when a model will achieve the desired goal, and an unknown number of iterations will be needed before results reach acceptable levels. In addition, it is normally not possible to either adjust/decrease scope or run the project to a predefined delivery date.

**Potential impact severity:** Medium/High

**Challenge difficulty:** Medium/High

---

Case study: Poker bot Identification

Online poker grew very quickly in the early 2000s, and automating play using software was attempted by many and successful for some. Those running online poker sites need to keep games free from “bots,” and this project was initiated to detect them.

Despite promising intermediate results, the project was cancelled before it was completed because the model didn’t reach expected levels in the first few iterations. Although the goals of an ML project can be well defined, there is no way of guaranteeing when a satisfactory model will be reached, and it is not until that point that you can claim to have accomplished anything of value for the client.

---

Privacy and safety

The lack of understanding of the internal workings of a large neural network can have implications for privacy and safety. The knowledge in a neural network is stored in a distributed way across the weights of the network. Although we know that specialization occurs in specific regions of the neural network, its exact mechanism is poorly understood. Thus, it is very difficult for designers to control where and how information is stored. To protect sensitive data, it can be obscured, but this can impact the efficiency of data exploration, model development and troubleshooting. Additional research is still needed to guarantee the methods used to secure the data.

To guard against unintended or negative actions, where models are involved in business-critical or “human”-critical inference, a process should be defined to ensure that any actions taken that are based on the model are appropriate.

**Potential impact severity:** High

**Challenge difficulty:** Medium/High
Building a production-ready AI system usually involves collaboration between people with different roles and potentially very different expectations on outcomes. An initial prototype may be built by data scientists who focus mainly on model performance and building a working proof of concept. Transforming a prototype into a production-ready system that also interacts with existing backend and frontend systems usually requires a significantly larger effort. This normally includes collaboration with, for example, teams in the organization like backend engineers, UX designers and product owners. While a data scientist might be pragmatic about their code as long as it achieves desired results in a controlled environment, members of the engineering team care a lot more about nonfunctional areas like maintainability and stability, which are critical to live systems.

Potential impact severity: Medium
Challenge difficulty: High

The figure below illustrates the 12 challenges described in terms of their potential to impact the successful rollout of an AI/DL project, and the ease in which a general organization – with limited experience in this kind of project – can find a solution.

As noted previously, the ratings for these challenges should be treated as indicative and simply used to help organizations think and plan ahead. Each and every business faces its own particular circumstances, just as each and every AI project may have a range of specific and even unique circumstances, meaning that the impact of a challenge may vary accordingly.
Those challenges sitting in the high impact and high difficulty quadrant, such as effort estimation and resource limitations, should be of immediate concern because of the need for specialist solutions before the power of deep learning technology can be leveraged most effectively. Having noted that every organization and each AI project is somewhat unique, there are some options to assist with these challenges.

There are a range of platforms and products available to provide an on-demand model to make projects easier to deploy and scale. Potentially, running a "platform-based model" makes it simpler to get started from scratch and create bespoke AI models, and this may be something to consider to mitigate these challenges.

For example on effort estimation, the organization can plan ahead recognizing that there may be uncertainty in the timings. One mitigating technique is to facilitate the project to get to each stage faster. Organizations need to generate and get access to data in the right formats for an AI project which can cause delay. There are both learnings and steps to ensure this data access step is repeatable for the following AI projects. In addition, getting models running, trained and evaluated quickly and in faster experimentation cycles will get the team to a point where although it may not get rid of all uncertainty, milestones can be hit earlier and quicker.

Resource limitations such as data science talent gaps are one of the most significant constraints for AI projects. Software can play a role by simplifying the tool set in use and making the whole cycle simpler, intuitive, transparent and auditable. If more junior data scientists and competent engineers can be assigned to some of these AI projects, this can also have an impact on solving the challenge by increasing the resources pool available.

There are some reasons why AI projects have issues as they migrate from development to production. Some are inherent to the nature of AI, and others stem from the relative immaturity of the sector. The latter will self-correct as tools and platforms evolve and as AI becomes more mainstream within the enterprise production environment.

Privacy and safety are global issues for AI and businesses should assess each project in this light. As regulators and government increasingly raise their requirements on organizations to manage privacy, it is imperative that AI projects are on the right side of regulation. Similarly, the potential for AI to do good (as well as impact the business) is very significant. Safeguarding is necessary against malicious and unintended consequences.
Challenges sitting in the high impact but low difficulty quadrant are those that a competent software organization should be proficient at tackling. When a project starts, these issues need to be defined and prioritized early, and good practice from the software development domain should be applied to avoid lost business value as development proceeds. Monitoring and logging and experiment management are currently two significant challenges for AI projects. Better processes and improved tooling hold the answer to solving these two areas.

The challenges that are low impact but high difficulty, such as dependency management and troubleshooting, are areas an organization should consider to determine if they are of concern. The level of experience required to solve these kinds of challenges is frequently beyond the knowledge set held within more traditional engineering teams; therefore, experts from the AI/DL domain should be consulted in order to put effective measures into place.

To date, some large companies have begun to use AI and deep learning routinely in their products and services. The true potential for AI will only become a reality when more organizations embed AI into their standard operating model. Operationalizing AI through faster modeling and experimentation cycles would enable more AI projects, which could result in increased innovation and greater business value. As described in this paper, there are a number of challenges to designing and implementing AI systems. Peltarion’s experience in deep learning projects reveals that these challenges are not always obvious at project inception and can be more difficult to resolve if a business has not considered how to navigate through an AI project and manage or mitigate the inherent challenges that arise.

Deep learning is still a nascent technology and organizations seeking to use it should be informed of these challenges in advance. This whitepaper has set out to summarize them so that business stakeholders considering AI projects can be cognizant of some of the pitfalls of working with AI, and steer their projects to avoid running into difficulty.

About Peltarion

Peltarion’s operational AI platform is a collaborative, graphical cloud platform for building, managing and deploying machine and deep learning models at scale. Peltarion makes AI technology more attainable and affordable by eliminating the software engineering overhead related to AI. Peltarion helps companies to focus on the actual value driving side of machine learning, instead of the infrastructure. Cutting down time to market and significantly improving the value of AI projects.