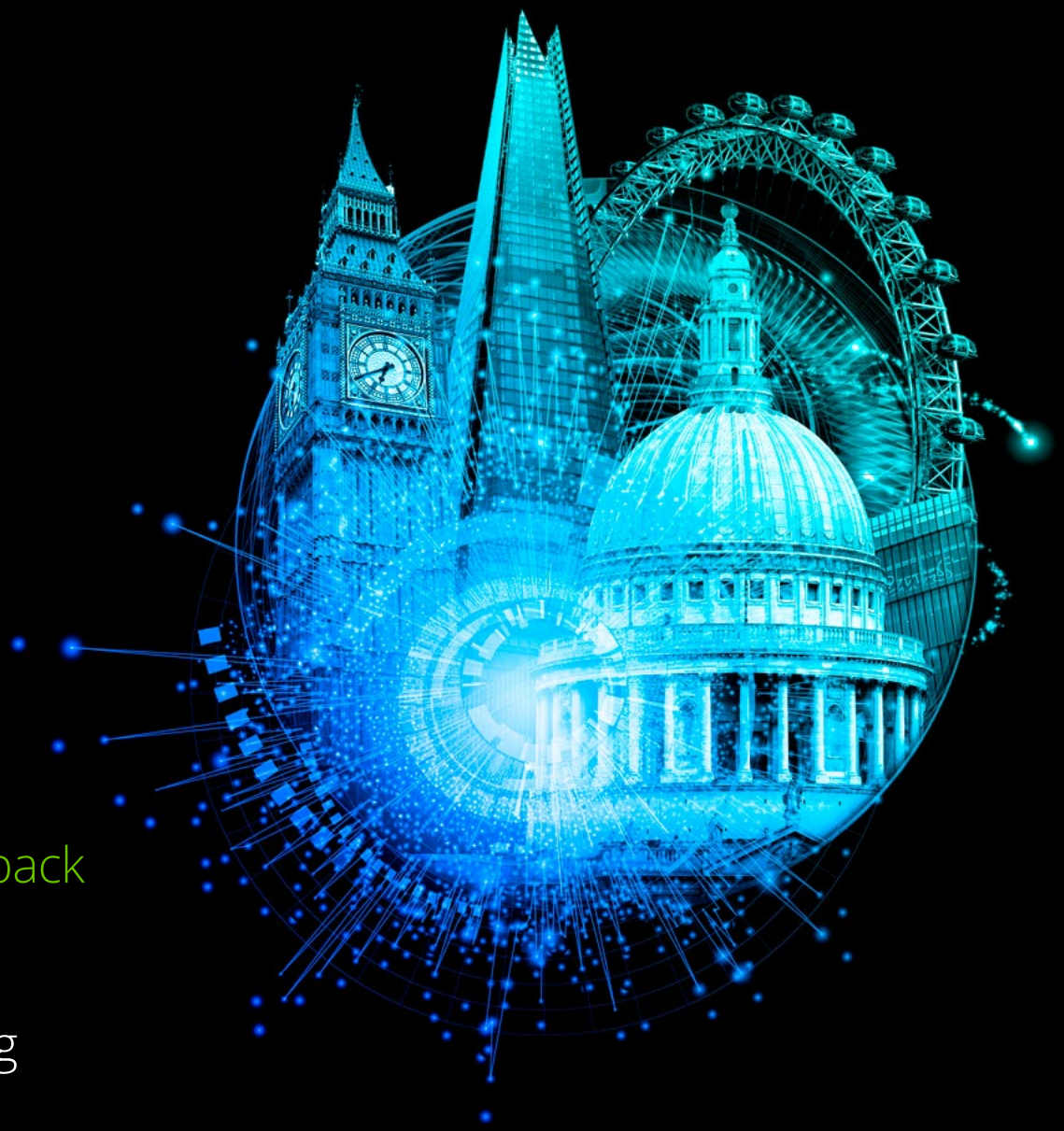
Deloitte.



Leveraging digital feedback for customer centricity

A practical review of Reinforcement Learning

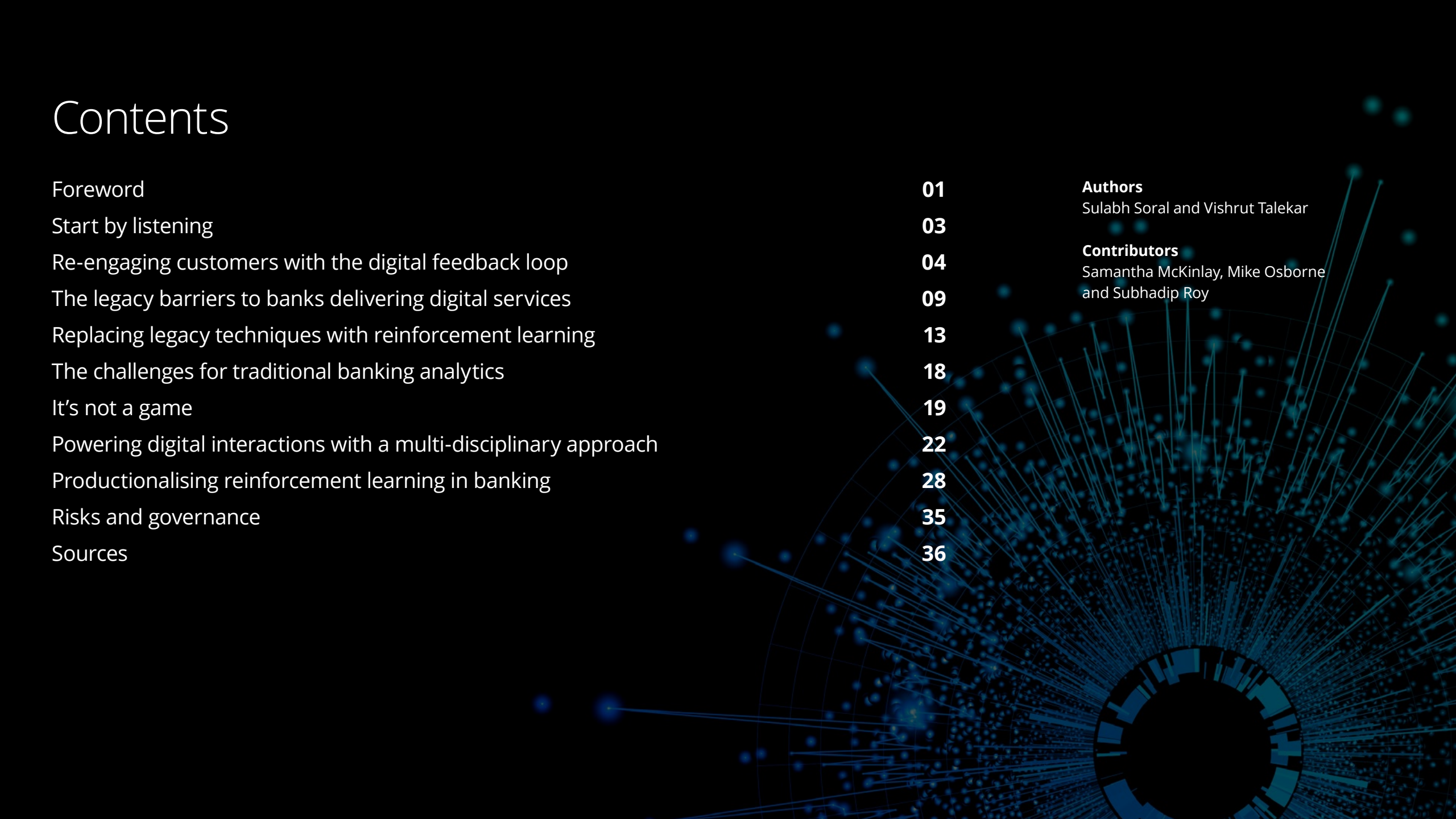
by Deloitte AI Institute, UK

MAKING AN IMPACT THAT MATTERS since 1845

Contents

Authors
Sulabh Soral and Vishrut Talekar

Contributors
Samantha McKinlay, Mike Osborne and Subhadip Roy



Foreword

Digitisation offers an array of new opportunities for organisations and requires a fresh approach to conducting business and servicing customers. Amongst other goals, it enables a firm to be customer-centric at scale.

However, when firms do digitise, customers expect a higher quality experience. Studies have shown that even one bad digital experience can prove very costly[1].

As the pandemic accelerates digitisation, organisations will also need to accelerate their servicing of these expectations. According to a study conducted by Deloitte UK, the crisis has significantly increased the use of digital channels across industries such as banking, retail, entertainment and health[2].

This creates potential challenges for organisations as they continue to digitise – particularly for banks. Research conducted in US, Canada and Australia suggests that customer satisfaction declines as retail banks move from branch to digital-only channels[3].

In this paper, we use retail banking as an example to explore the implication of digitisation for organisations.We see how a 'continuous learning' approach can shape customer centricity.

We examine how interaction data generated through digital channel feedback between a bank and customer can be utilised to generate continuous learning with appropriate AI technology – like Reinforcement Learning.

And we consider the technology choices required to support such a strategy, as well as the resulting operational and risk implications.

The rewards of getting it right are great. But, as you will see, this is no walk in the park. Success will require investment – in every sense of the word.

With that in mind, our aim is to present a comprehensive overview of the extensive requirements, challenges and potential solutions that arise from applying this technology for readers who are exploring the topic.

Indeed, although we use the example of retail banking throughout this paper, we believe that the information within it is applicable to any organisation servicing customers via digital channels and exploring how Reinforcement Learning can improve customer experience.

"The rewards of getting it right are great. But, as you will see, this is no walk in the park."

"If you're not listening, you're not learning."

Lyndon B Johnson

Start by listening

To become customer-centric, organisations need to actively listen to their customers. Historically, banks have 'listened' to their customers through channels such as branches and contact centres.

Although, these channels provide qualitative customer information, they do not usually reach all customers. In fact, many of these engagements between the bank and its customers are highly transactional and generic in nature. At times, this has been supplemented with customer focus groups and ethnographic research, both of which give a lot of insight on a sample of customers, but do not include all customers.

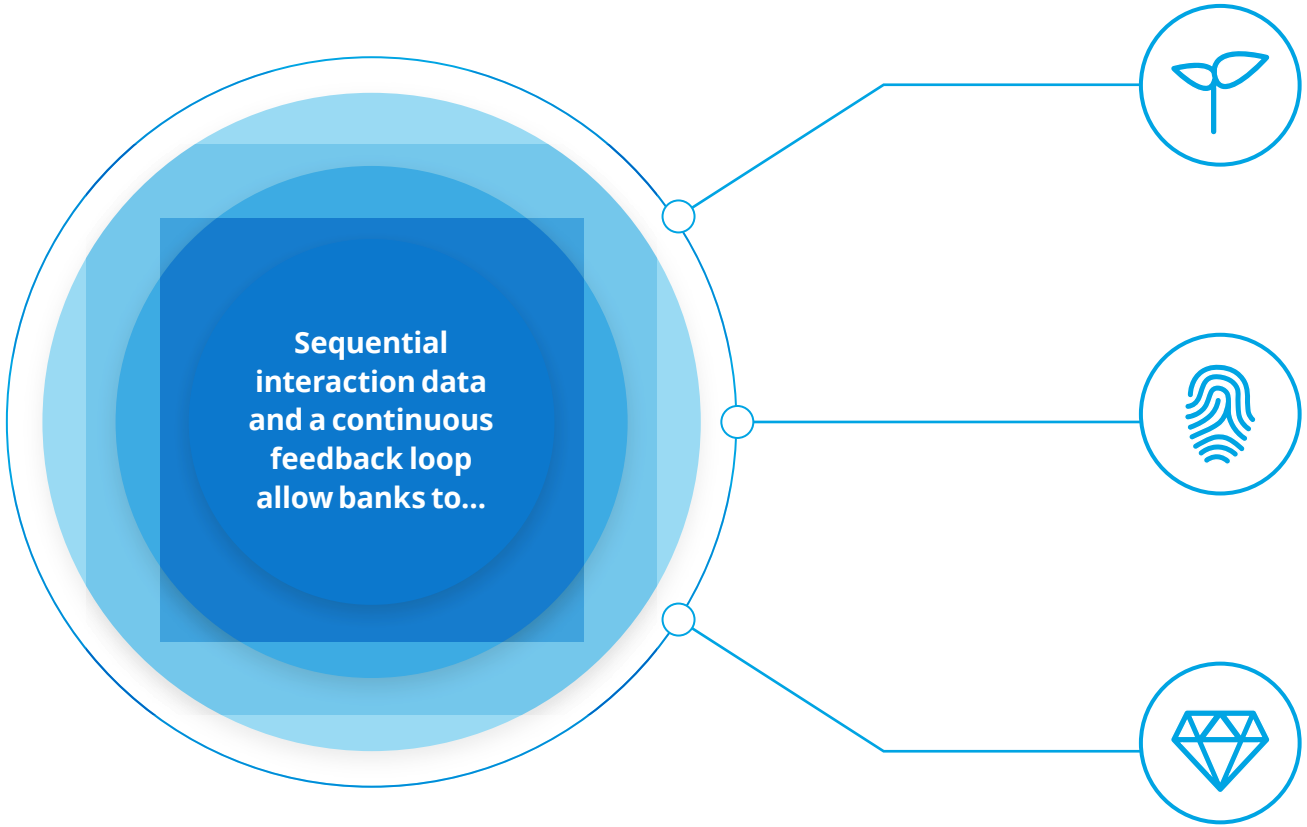
Indeed, due to the reduction in face-to-face contact and with the increasing penetration of analytics, the data collected on the customer at different stages of their lifecycle by the bank or through third parties has been utilised as a proxy for listening to the customer. This 'proxy listening' has – by necessity – become the primary source of customer understanding, but such data cannot replace the deeper understanding of customer needs gained from actual engagement.[4]

Re-engaging customers with the digital feedback loop

The availability of digital channels provides organisations with an opportunity to re-engage with millions of customers at scale, something traditional channels were unable to do. Digitised organisations can now interact, gain feedback and learn from their customers by capturing and acting on sequential interaction data.

In a 2020 report, WEF states that "near-term value will be found in use cases that combat data fragmentation and accelerate feedback loops which anticipate customer moments of need"[5].

How the digital feedback loop delivers customer centricity



focus on Customer Financial Wellness and have a long-term outlook on the customer relationship

Customer centricity has increasingly come to mean more than just understanding which products customers want. Specifically, retail banks are focusing on being guardians of customer financial health. After the credit crisis and more so during the pandemic this focus has further widened to include financial inclusion and financial sustainability.

establish a multi-channel Hyper-personalised Experience to build deep customer relationships and trust

To deliver personalisation, banks need to have an ongoing dialogue with their customers. A continuous digital feedback loop helps build an incrementally better view of evolving customer needs and enables highly personalised view of the customer determined by individual requirements, as opposed to the traditional discrete product centric view.[5]

increase their utility to the customer through Frequent Value Delivery

Here the feedback loop works like a pipeline through which the bank gets notified as customer needs arise, this could be in real time. However, this feedback loop only works if the customer is engaged and communicating with the bank – else there is no feedback to "listen" to and act upon. So it turns out that for customer centricity to work, we firstly need to think about how to engage customers in a meaningful and sustainable dialogue.

Investing in customer financial wellness

Setting customer financial health as an objective would mean that the bank will need to have a long-term outlook on the customer relationship. In practice this means accepting that rewards must be realised over a longer-term – sometimes foregoing near-term revenue for the greater benefit of the full customer lifecycle.

What is a hyper-personalised experience?

There are many ways to define hyper-personalisation, however, for the purposes of this paper we will utilise the definition included in Deloitte's paper which explains that "Hyper-personalisation can be defined as harnessing real-time data to generate insights by using behavioural science and data science to deliver services, products and pricing that are context-specific and relevant to customers' manifest and latent needs"[6].

The paper also observes that organisations need to focus on hyper-personalisation due to fast-evolving customer expectations driven by the rapid growth in CX.

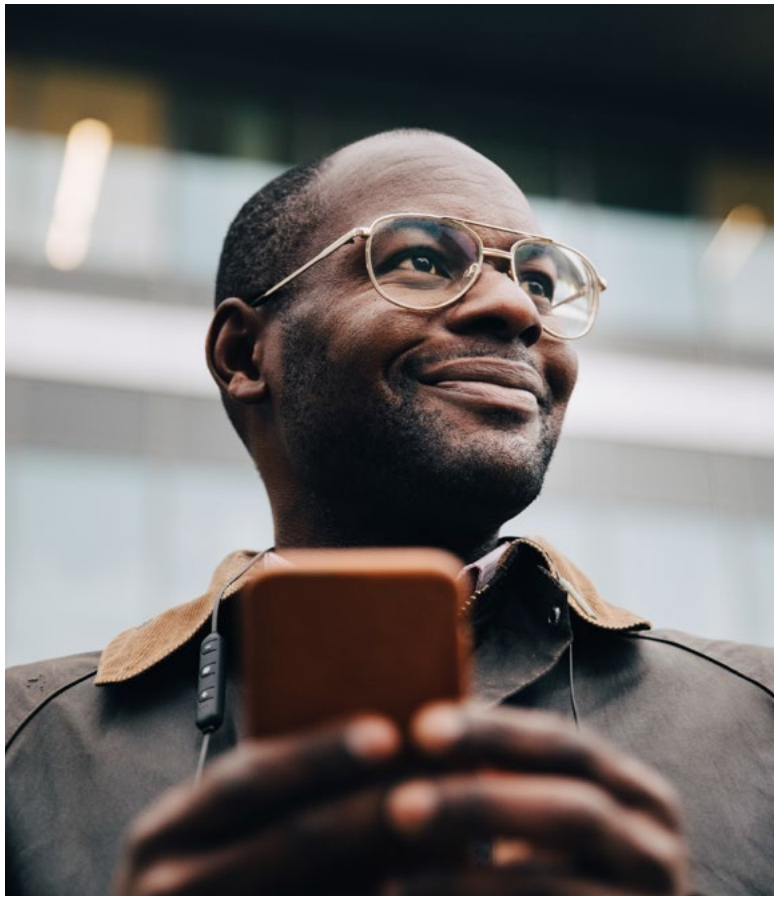
And with banking regulations pushing for greater financial inclusion, Hyper-personalisation is seen to help them evolve from a discrete offerings paradigm to something that is centred around individual customer needs.

Indeed, according to an HSBC report, "Customers will increasingly be able to expect a highly-personalised service determined by their individual requirements, instead of based around a set of savings, borrowing and investment products, each with their own sales and servicing characteristics"[6,7]

However, the current consensus is that most customers do not think that they are being delivered personalised services by their banks. In fact, according to the Deloitte report "94% of banks cannot deliver on the hyper-personalisation potential"[6]

So, how can this be put right?

> "Customers will increasingly be able to expect a highly personalised service determined by their individual requirements."[6,7]



Become more 'useful' with frequent value delivery

Keeping customers engaged is never easy but in financial services it is particularly challenging. In this product-centric world the potential conversation touchpoints that a bank can have with a customer are limited. There are only so many products and services that banks can offer.

Consequently, for banks to really become customer-centric and keep customers engaged, they need to increase their utility to individual customers through an expanding set of features. By features we do not necessarily mean services and products in a traditional banking sense. They can also include non-banking things that the customer finds useful, such as...

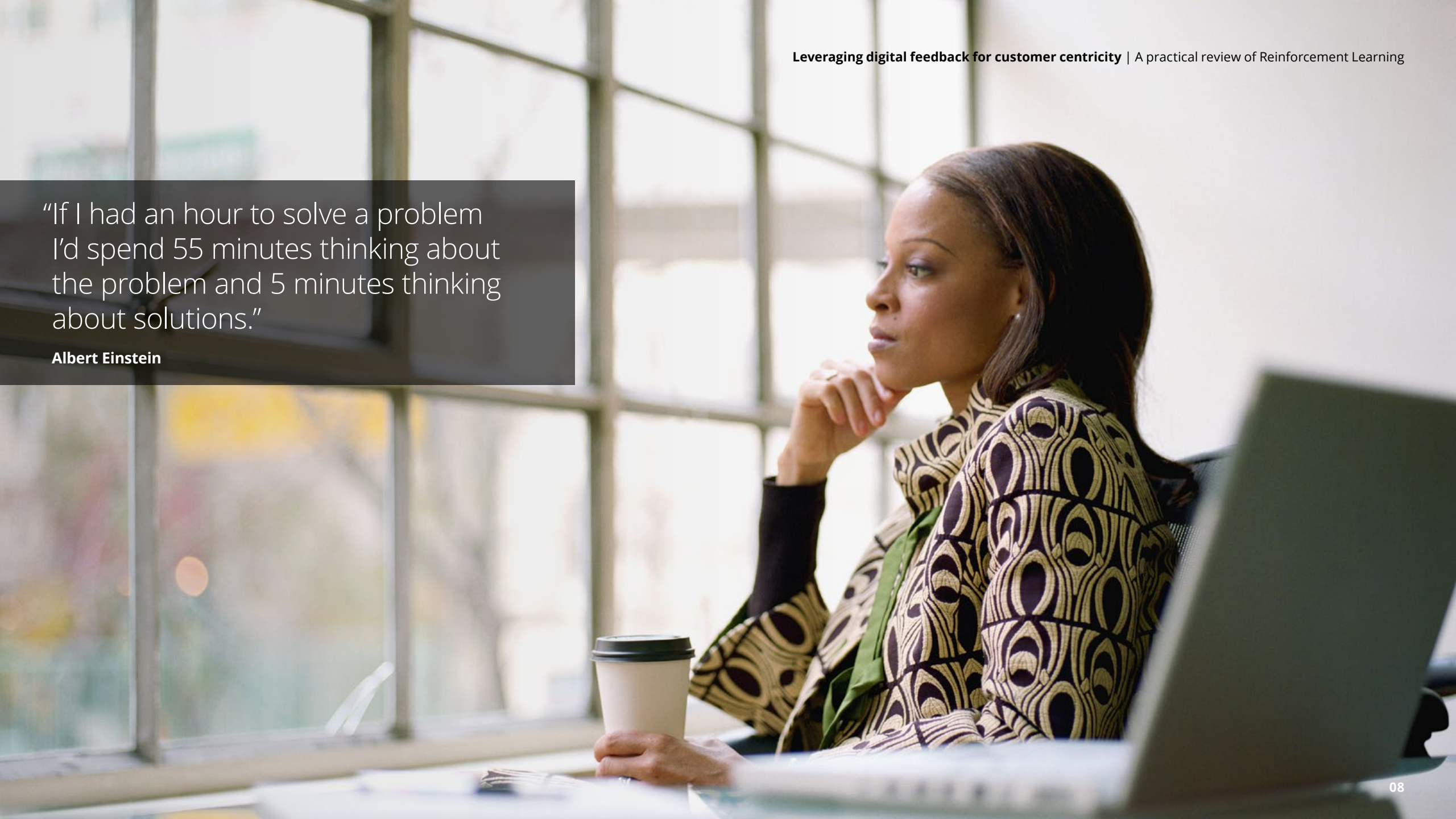
- **Being a financial guardian** – give customers insights about their spending; reminders about upcoming payments; encouragement to create a budget, etc.
- **Providing financial offers** – cite areas for potential savings or gains via third party vendor switch.
- **Delivering non-financial added-value insights** – provide information related to interests such as the weather (if the customer is expected to travel based on a recent travel merchant transaction); availability of retail offers (based on purchase history).

Thanks to the virtuous data cycle of the continuous feedback loop, banks can become more customer-centric – better serving customers by staying current with their needs through personalised and frequent engagement.

Due care is required to maintain appropriate and useful levels of engagement – always keeping the customer's long-term financial health at the heart of the interaction.

That's why it's important to understand exactly which features really make a difference to a customer by asking the right questions:

- What value do they bring?
- How does this value get accentuated as different features interact with one another?
- And how do these features deliver on longer term bank-customer relationship objectives?



"If I had an hour to solve a problem I'd spend 55 minutes thinking about the problem and 5 minutes thinking about solutions."

Albert Einstein

The legacy barriers to banks delivering digital services

If we compare the requirement of integrating continuous customer feedback and longer-term reward maximisation to the current customer analytics methodologies at most retail banks, we observe a mismatch and/or inefficiency.

Legacy customer analytics methodologies such as traditional recommender systems focus on the short term and are myopic.[8] They evaluate whether a customer is likely to purchase a product based on life stage, lookalikes and past consumption behaviour as a proxy for need.

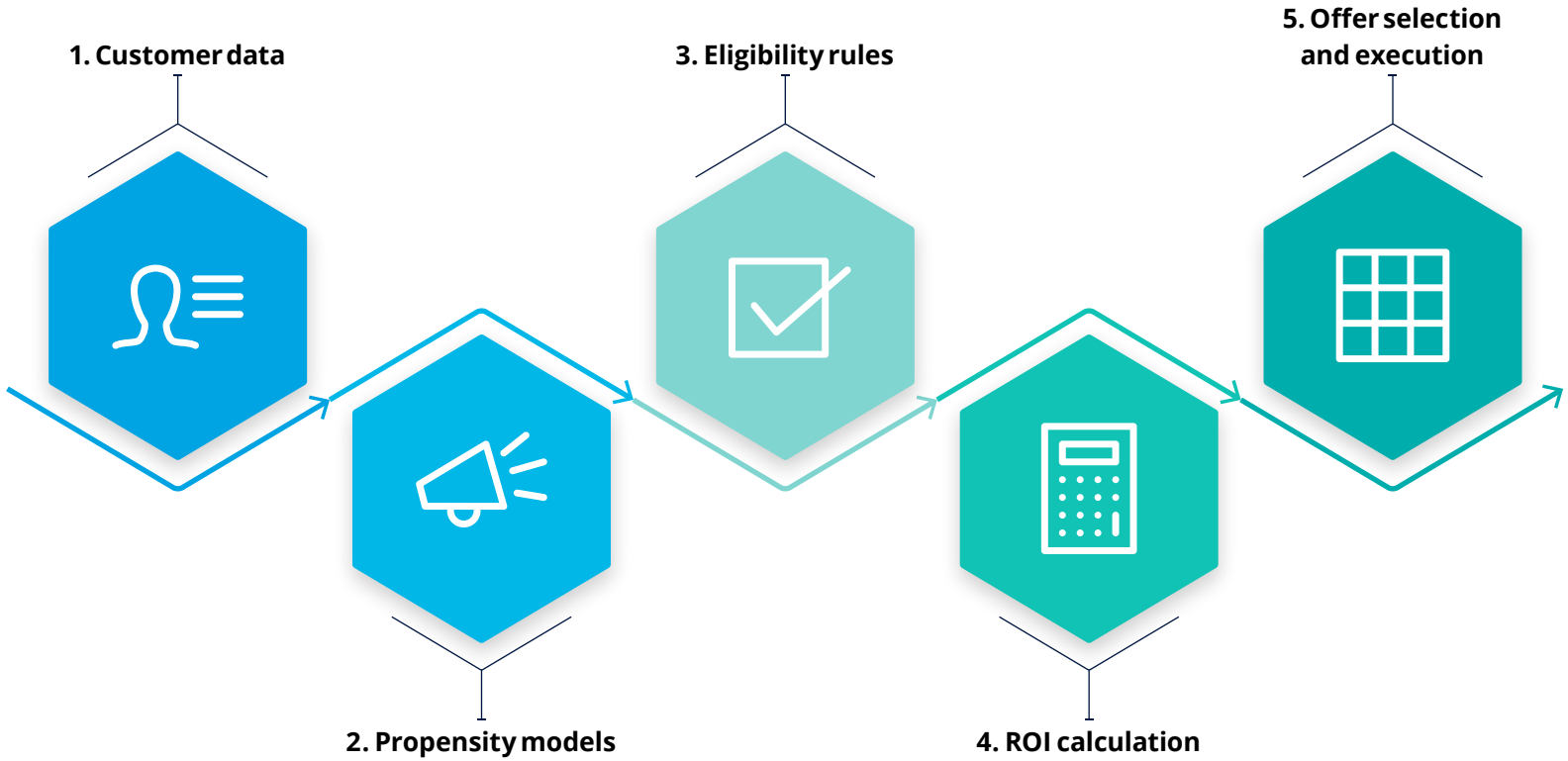
Unfit for purpose

These methodologies neither consider the longer-term impacts of their recommendations nor the efficacy of these recommendations in the context of all other possible interactions that the bank has with the customer. This is not to say that banks have not considered longer term strategies, but that even in those cases, similar models were stitched together to support execution.

In short, these methodologies are fit for purpose in a legacy product-centric view but not in a customer-centric world. This challenge has been widely covered in many academic papers referencing industries such as retail and others.[9, 10]

Let's look at this through the example of typical marketing analytics services which are very much product- and offer-centric. You have a set of offers/products and a customer base to reach out to. Analytics is being used to maximise the impact of the campaign/offers and optimise marketing ROI.

Five key components drive this process



1. Customer Data

Typically, the biggest asset and driver behind the analytics framework is access to rich customer information.

a. **Customer profile/demographics** – Captured at a point in time in the past. Often, these data attributes are not updated to reflect the current customer demography. (e.g., change in salary/job, change in lifestyle, marriage, number of dependents, etc.). This is a missed opportunity.

b. **Transaction data** – Transactions that the customers made in the past. Making the most of these limited transaction data is key to understanding the customer profile. Most financial institutes are using analytics to extract valuable information from simple debit/credit card transactions. Identifying the merchants, location, time of the day, channel and value helps you to create a customer pen portrait.

However, individual banks may never get a view on the full transaction behaviour of a customer because they use multiple cards and accounts. Also, a substantial number of transactions may not reveal much about purchase or other behaviours – such as when a customer transfers money to a friend without mentioning the purpose.

c. **Contact history** – Data captured during all customer contacts helps banks to assess the impact of any contact with the customer. Customer response to an outgoing call/email/notification with an offer can be captured with click-through rate, mail opening, and subsequent applications. Data captured during an inbound call can help to understand customer concerns, channel preference, while opening a cross-sell/up-sell avenue.

On the downside, these interactions (e.g., selling a product or handling complaints) are irregular customer engagements. Additionally, the rate of customer feedback to product offers, whether positive or negative, is low. And even when feedback is received, it may not be instantaneous.

d. **Third party data** – These are typically paid data attributes sourced from vendors to enhance information about your customers, such as risk score, customer segments. Extracting other information through social media, news, and other open-source data helps you enrich the customer data profile and make offers more contextual.

"The biggest asset and driver behind the analytics framework is access to rich customer information."

2. Propensity models

After the bank has sourced the relevant data, the objective is to develop one or a set of models that calculates the probability of response to any offer or product at a customer level through a statistical propensity model.

Based on previous customer responses to similar offers, the model is trained on historical data using a supervised algorithm (logistic regression, random forest, decision tree, etc.). The algorithm will generate a point estimate that can be interpreted as the likelihood of a response from the customer in the event of a positive contact made through the preferred channel.

In most circumstances, these models don't consider current customer mindset, feedback and other interactions that can influence their long-term affiliation with the brand and subsequent engagement campaigns.

3. Eligibility rules

Once the bank has a model that predicts the probability of response to a campaign at a customer level, the next step is to identify the target population. Applying certain heuristic business eligibility rules and marketing consent, the target population is selected for offer roll-out. In many cases, marketers lose a large percentage of their customer base due to these eligibility rules, making many customers uncontactable.

4. ROI calculation

Every marketing campaign is restricted by a marketing budget. A mathematical ROI calculator helps you to identify the most profitable customer pool based on the expected ROI. Typically, the return is calculated as 'Expected Value x Response Probability'.

5. Offer selection and execution

After the offer/campaign level ROI has been calculated, the top offers for the customers are ranked for execution. Based on marketing objective and budget, the offers are rolled out using a different channel.

As we can see from the above description of a common analytics workflow, the purpose of these models is to predict if a customer will respond to a particular product rather than directly determining what the customer needs. Also, in most cases, the customer interactions have been one-sided, and the analytical model doesn't capture customer feedback or analyse the sequence of interactions, making little impact on what subsequent actions will really help the customer.

"Most of human learning is unsupervised learning. If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake."

Yann LeCun (On true AI) Carnegie Mellon University

Replacing legacy techniques with reinforcement learning (rl)

Reinforcement learning (RL) can model sequential interaction data – something traditional analytics cannot do. In fact, according to researchers from Google Research and University of Texas at Austin, the next generation of recommenders will increasingly focus on modelling sequential user interaction and optimizing users' long-term engagement and overall satisfaction. They further state that RL lends itself well to power these Collaborative interaction recommenders.[11]

A brief introduction to Reinforcement Learning

Reinforcement Learning sits as one of three major groupings for machine learning models, the other two being 'supervised learning' and 'unsupervised learning':

- **Supervised learning** – Given a dataset and a 'correct' answer for each element of that dataset, we use a model to predict the correct answer for new elements from a previously unseen dataset or data record.
- **Unsupervised learning** – Given a dataset but no specific 'correct' answer to look for, we use a model to find interesting patterns or groupings for elements of the dataset.

A scenario exists, however, where some answers are better than others, but we don't know what those are.

All we know is how well our previous answers performed and we use that to decide on what to do when similar scenario appears. This is the stand-out feature of RL.

Imagine an experiment where a mouse is rewarded with cheese for escaping a maze. By repeating the experiment, we hope that the mouse will learn that escape is desirable. We could then manipulate the experiment by rewarding the mouse for using shorter routes to escape by varying the quantity of cheese, or even removing the cheese for previously found routes.

By doing this we can get the 'agent', our mouse, to find better and better solutions to a problem where we have not specified the best solution ahead of time, simply by rewarding certain outcomes and penalising others.

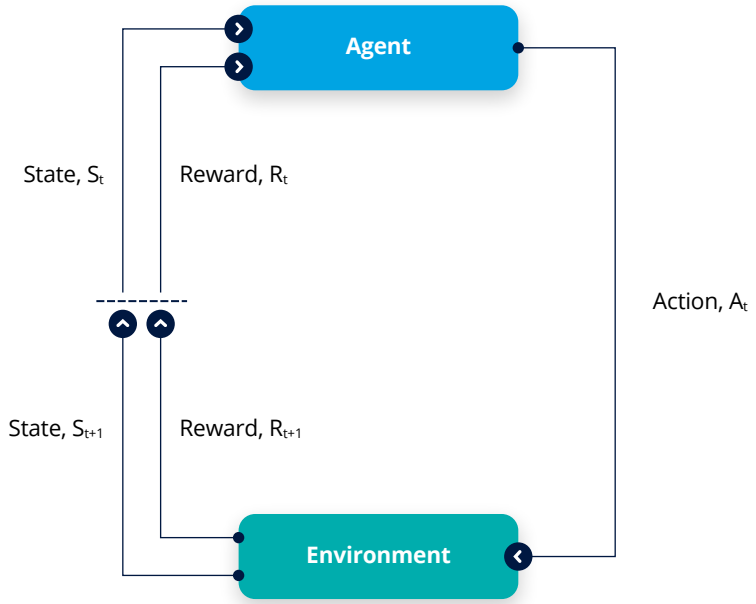
Source: https://www.vitalherbs.co.uk/wp-content/uploads/Lions-mane-mouse-maze.jpg

This experiment encapsulates the RL process and introduces us to some of the key components of a more structured approach. These are...

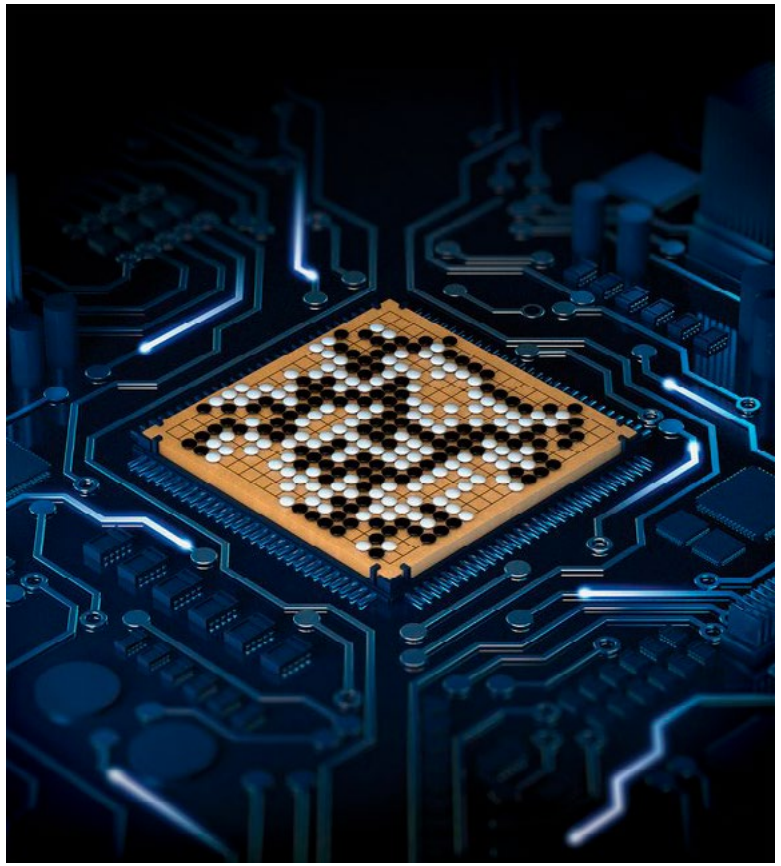
1. **Agent –** The model we want to train: the mouse.
2. **Environment –** The real-world scenario in which the agent exists: the maze.
3. **State –** The agent's imperfect view of the environment: what the mouse sees and remembers.
4. **Action –** The moves the agent can make: the mouse moves forward, backward, turns left, etc.
5. **Reward –** The feedback the agent receives for its selected action: the presence and size of the cheese.

"All we know is how well our previous answers performed and we use that to decide on what to do when similar scenario appears."

These five components are traditionally displayed in the following diagram[12]



Here, a continuous clockwise flow represents the process of an agent being in a state (their view of the world), taking an action based on that state which has knock-on effects on the environment. The agent is then rewarded for their action, while the changed environment results in a new state.



Source: https://storage.googleapis.com/deepmind-live cms/images/AlphaGoBreaker.width-600_ZAupxf4.jpg

Although this approach does allow a solution to all RL problems, given that environments and states become larger and more complex, it is increasingly difficult to create an agent that knows what to do in each unique state.

The modern iteration of RL has revolved around solving this with increased computational power and modern techniques, including neural networks, to tackle real world problems.

The most famous example of the presence of both requirements is 'AlphaGo' where Google's DeepMind trained an agent to play the game of Go, beating the human world champion.[13]

Now, let's look at a user journey in the maze of retail banking...

How to engage a bank customer

Let's say a customer has set up a current account with the bank and uses the bank's app as one of the channels to interact with it. Perhaps they like to check their balance once in a while, or perhaps they like to keep a tab on weekly spending. In either case, they are an intermittent user of the app and consequently not engaged with the bank. Maybe the app has too few functionalities, does not cater to their personal banking needs or they do not know how to find their way around the app?

How do we engage with this customer better? How do we find out what their requirements are and how they are evolving? How does the customer like to be communicated with? Which channels? In what tone of voice?

In a traditional set-up, banks would have to commission multiple researchers and tests to answer those questions, and even then the answers to choices would only be at the segment level. But in our case, we want more personalised insights at scale. The key goals include hyper-personalised customer services and addressing customer needs as they arise, with a focus on optimising longer term customer engagement.

This journey is more complex than the one we saw in the mouse example for many reasons such as human behaviour; multiple products/services on offer; potential time duration of relationship; regulations affecting the bank's actions, and market dynamics including competitors, etc.

Let's walk through the 5 key components of RL to understand the roles of the major players in the algorithm:

1. **The environment** – The user/customer represents the environment. The observable parts of the environment consist of information about the customer such as their identity, demographics and occupation. It also includes their product holding. There are latent features of the environment such as customer behaviour. It is important to note that not all the information on the environment or that impacts the environment is known.

2. **State** – The observable part of the environment at a given point in time.

3. **Agent** – The model we want to train: the RL algorithm.

4. **Actions** – The actions are services, products, insights, reports and communications that the bank can offer to the customer.

5. **Rewards** – The customer reaction is the feedback that determines reward or penalty. Rewards can be short term or longer term.

Other useful terms

- **Episode** – An episode is the completion of a task or a sub task which has a definite horizon in dimensions, such as the number of steps, length of time, or victory/loss in a game. Typically, RL agents get better at completing the task by learning from feedback/rewards over multiple episodes. For example, the mouse will get better at escaping the maze by playing the game over and over again. In retail banking, an episode could be the dispatch and subsequent acceptance of a product offer by a customer.

- **Types of algorithms** – The type of RL algorithm depends on the type of problem that needs to be solved. For example, **Model-based RL** is used in cases where the model has all the information on the environment it is interacting with.

  On the other hand, a **Model-free algorithm** is used when the agent has partial information on the environment it is interacting with. Most real-life situations are a setting for Model-free RL. For example, a bank will not have access to all the information on a customer and their behaviour.

  There are other distinctions between RL algorithm classes based on what exactly the algorithm is trying to learn. For instance, is it trying to learn the value of following a given policy of actions, or is it trying to learn the optimal policy of actions?

- **Exploration vs. Exploitation** – Major classes of RL algorithm such as Model-free use the process of exploration and exploitation to master a given task.

  Exploration is when the agent tests different actions in response to a given situation and receives feedback (rewards) from the environment. Note that in the beginning of the exploration phase the agent does not have much information on which action to take and when.

  Over time the agent starts learning what the best action is for a given situation and it then starts to **exploit this knowledge** if a similar situation arises in the future, thereby mastering execution of the given task.

The value of further reading

Please note that RL is a vast field, and it is not within the scope of this paper to review all its technical details. Our intention here is to briefly introduce some key technical RL concepts that will help the reader to develop a basic understanding of what RL is and can achieve. There is an extensive body of work on this topic that will reward further exploration.

"What we want is a machine that can learn from experience."

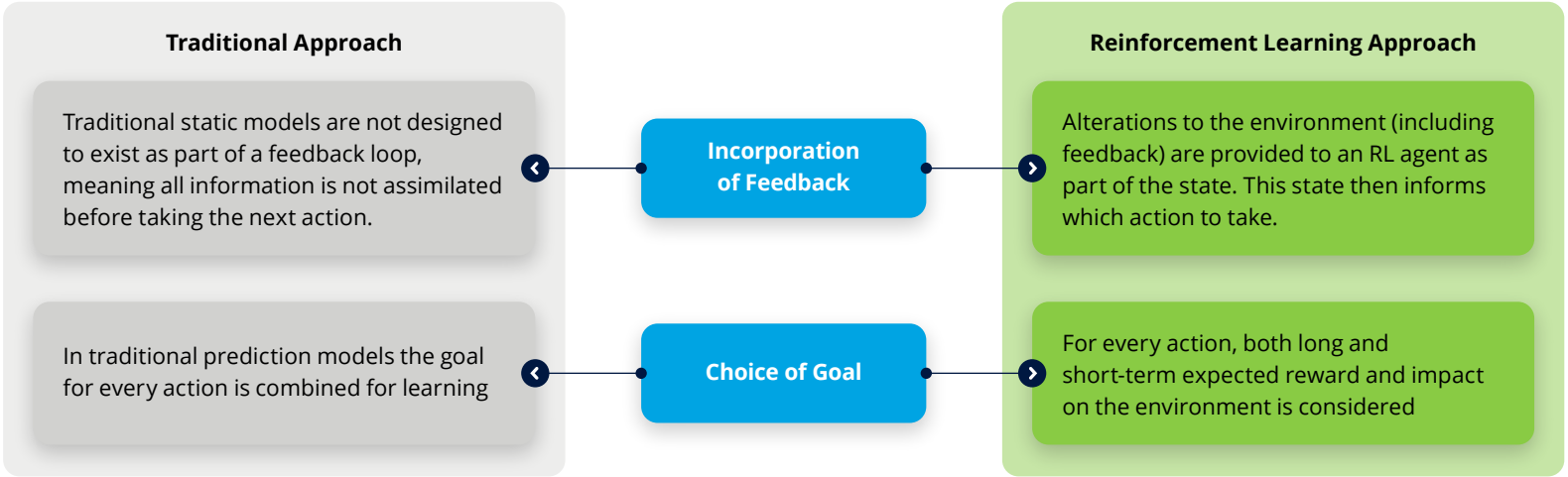
Alan Turing 1947

The challenges for traditional banking analytics

Having covered a brief introduction to RL and noted some simple applications in retail banking, let's revisit the key differences between traditional banking analytics and a customer-centric approach powered by RL.

Since the construction of the problem and the goal of the solution differs between the traditional approach and the customer-centric RL approach, a corresponding difference exists in the process of model development, infrastructure requirements and data processing. RL has been gaining media attention and is being applied to many fields such as robotics, manufacturing and energy management. However, using it to train artificial agents to play games is arguably the best-known success story.

In the context of recommender systems to power digital applications in retail banking, RL can potentially help maximise longer term customer outcomes by taking advantage of sequential interactions available through digital channels. However, there are significant challenges to this – not least the technical complexity of training and deploying RL.[14]



It's not a game

To begin with, unlike in a video game where the agent can be afforded the luxury of learning to play the game 'live' over many episodes, the AI in B2C digital applications needs to be trained before it interacts with the customer. A RL agent taking random actions without any understanding of the customer's needs is a recipe for disaster. The time taken for the RL agent to learn the task effectively can involve many episodes and conversations. This is a major hurdle to learning live – especially in the case of commercial applications.

> "A Reinforcement Learning agent taking random actions without any understanding of the customer's needs is a recipe for disaster."

In addition, typical examples of RL applications are activities from the physical world and in many cases, these could be categorised as finite games with a definite time horizon, defined rules and rewards. Retail banking, or any other business, are like infinite games, with a changing environment and rewards.

Therefore, it is difficult to define what an Episode is in the context of retail banking. The relationships that banks have with their customers are multi-faceted, potentially longer term and akin to continuous play.

Additionally, the customer relationship is not driven by repeated purchase of a commodity, as happens in retail consumer e-commerce – an example of an episode in a real-life commercial setting.

Environmental randomness

The RL agent may not be aware of competitor banks vying for the same customer. Negative feedback from a customer or a no response on an offer may actually mean a better offer from a competitor. This increases environmental randomness, impacting performance because the agent is not able to effectively correlate action taken with feedback received.

This brings us to another challenge concerning commercial applications. The RL agent needs to learn and incorporate newer actions**,** which is quite different from video games. As businesses respond to market dynamics or customer needs by, for example, introducing new products or features, an RL agent will need to learn how to incorporate these new offers in its repertoire of available actions.

Lack of history

In almost all circumstances, the agent will have limited or no historical data to learn from, especially for individual customers. It will not be able to compare how a new action is better/worse off than an older available action that it has already trained to execute. In addition, a bank can take many different types of actions for each customer and these actions can be of different types – increasing both the size of the action space and the complexity of the task.

A related challenge with commercial businesses is that many actions cannot be trialled multiple times by the RL agent to master execution, unlike in a video game where an episode can be repeatedly played. For example, if a product is only valid at a certain life stage, the agent can only offer it if the customer is at that life stage. This limits the data available to the agent to learn and improve performance.

However, the most complex challenge is the formulation of an appropriate reward and penalty mechanism for the RL agent. It is this mechanism that helps the agent learn an optimal policy of actions to accomplish the objective. In a video game the objective is simple and clear because there is no disconnect between the objective – maximise the score – and the reward. However, in real life this could be much more complicated especially when the agent is dealing with human beings.

The key challenges in formulating a reward mechanism are:

- **Human psychology** – The incomplete information available to the RL agent when dealing with customers is exacerbated by human psychology and potential economic irrationality exhibited through our decision biases. This causes a deeper disconnect between action and reward, confusing the agent. It also makes it difficult to construct a reward function that confirms to the economically rational approach that is potentially easier for AI to learn.

- **Sparse rewards** – In certain instances, the agent may have to take multiple steps before it is rewarded. So, if an RL-powered recommender engages with the customer every day, but only gets rewarded if the customer is retained at the end of every year, then the motivation for the agent to learn to reach the objective is diminished. At worst it may not learn the appropriate actions to reach the objective, and at best it may take a long time to learn. This can cause a poor customer experience.

- **Sub goals** – These are objectives given to the RL agent in addition to the main objective. They can help reduce sparsity in rewards as the agent gets rewards in shorter steps on its journey to the final objective. However, as sub goals increase so does the complexity in building a working reward function. The more multidimensional the reward function, the more complex the training of the algorithm will be. In most cases, complex reward functions hinder the agent from learning policies to maximise the primary objective.[15]

  For example, a banking app with a primary objective to maximise customer engagement, can use several sequential actions to achieve that objective. Some of these actions could be lower value (in terms of reward) than others. A mis-constructed reward function can push the agent into learning lower value actions or shortcuts that are not beneficial to the customer or the bank.

- **Real life penalties** – The penalty of losing a customer is significant for the bank. Once lost, the customer may never come back – unlike video games when we can always play again. But building a reward function that takes into account real life penalties is also complex.

  For example, disproportionately penalising the RL agent for losing a customer may cause the agent to freeze learning and get into a 'do nothing' loop. Always remember that positive rewards may lead to addiction and negative rewards to avoidance.

- **Erratic feedback** – Customer feedback – on which rewards are based – can be erratic. For example, the customer might not give a desirable response to an offer – and then accept it after a long delay. Such instances cause the relationship between action and reward to weaken and reduce the performance of the RL agent.

"Your most unhappy customers are your greatest source of learning."

Bill Gates

Powering digital interactions with a multi-disciplinary approach to reinforcement learning

Projects to apply RL and leverage the digital feedback loop for customer centricity need careful planning and a multi-disciplinary approach.

Structure the Problem

Assemble the problem by breaking it down into smaller concepts. Leverage existing analytics and processes where concepts are already well known and function well. This will make sure that industry standards are well considered.

Define your States, Actions and Rewards

Define your environment consisting of the following elements:

- **States** – States should hold the information of customer available for agent to observe, understand and learn.
- **Actions** – Design your actions carefully to increase the likelihood they address the problem properly. Consider the type of actions you can take and the implications; for example, selecting continuous or discrete actions changes the type of algorithm you would use.
- **Rewards** – Behavioural psychology tells us "you get what you incentivise, not what you intend". Rewards need to be constructed with care, linked to the objective and delivered in a timely matter. They must also be directly attributable to the agent's behaviour and the agent must understand the rationale for the reward.

Setup Algorithm, Training and Updates

Use offline learning and existing propensity models as a starting point. Then decide which parts of the algorithm should use reinforcement learning, and which parts should remain the same. Consider the different types of training available, from curriculum learning and SME training to novel solutions; and the time required.

Define policies for Exploitation and Exploration

Clear goals and policies are required to balance exploration and exploitation. From a technical perspective, too much exploration gives junk data, and the agent learns nothing; while too much exploitation leads to suboptimal behaviours in the policy.

Productionise your Reinforcement Learning Agent

Plan production such that the reinforcement learning powered system is not only capable of serving multiple customers but also capable of learning and generalising from customer feedback asynchronously.

Whether it's a use case where a lot of data is generated and the scale of personalisation required is large, or an issue that requires a highly personalised solution, structuring the problem and setting up the correct learning objectives can be tricky. Following best practice can resolve these complexities:

1. **Arrange the problem into smaller concepts[16]**.
   As previously mentioned, real-life engagements, such as between a bank and a customer, are multi-faceted and have many objectives. Additionally, the task of achieving a single objective may have many sub parts. For example, the task of retaining a customer over 5 years could be broken down into the task of retaining in year 1, then in year 2 and so on. Each of these sub tasks could be further broken down into steps such as assessing which customers are likely to leave the bank, assessing what will stop them from leaving, and then engaging them with the right solution. Breaking down the problem into smaller concepts helps to clearly define the RL problem and reduce the complexity of the final solution. It is also important to understand which part of the problem is a good setting for RL and which parts can be catered for by existing analytics or other simpler techniques that already function well.

2. **Consider the length of the episode carefully.**
   Once the task where RL is to be applied is identified, it is important to consider other features that define an episode, such as how long the episode is. For example, if the objective is to maximise retention at end of year 1, then the episode will run for 1 year and presumably have multiple sub tasks along the way. On the other hand, if the task is to maximise engagement by only contacting customers at the 'right time', then an episode could be on a much shorter time horizon. Generally, for RL to achieve success at mastering the objective, we need to reduce complexity so that tasks with less sub steps will be mastered with less effort than longer ones.

3. **While formulating the problem, ask the following questions:**

   - What is the data velocity like?
   - Is the problem I am trying to solve divisible?
   - Can we clearly define the utility/reward function?
   - Will we get sufficient feedback?

4. **Design the actions that the RL algorithm can take carefully**. Actions can be continuous (quantitative) or discrete (classes). For example, the action set of a RL algorithm that is trained to pick the best time to contact a customer is continuous. On the other hand, the action set of an RL agent trained to provide a product recommendation consists of discrete classes. The type of action determines the type of algorithm that will be used and the complexity of learning. It is always better to keep the number of possible actions limited and relevant to the primary goal. The greater the relationship between reward and action, the better the learning chances.

"The greater the relationship between reward and action, the better the learning chances."

5. **Constructing the reward function is one of the most important and complex steps**.[17] The rewards incentivise the agent to learn and master a given task. As already noted, designing a reward function for real-life situations is vastly different from designing them for an artificial environment, such as a video game. Banks conduct multiple interactions and initiate several actions with the customer within a given period – such as sending balance information, giving insights on spending, resolving complaints, etc. Each of these tasks may have a different value to the customer and the bank. Also, cumulatively these actions may impact a larger goal. For example, affecting the behaviour of a more engaged customer who prioritises the bank over competitors whenever a financial need arises. The common approach for formulating the reward function for such a scenario is by utilising knowledge from domain specialists (such as product, customer and other teams) and insights from customer data (priors). This process helps to identify and appropriately reward intermediate steps that lead to achieving the main goal. However, care must be taken not to transfer process inefficiencies and previous biases to the RL agent by rewarding the wrong behaviours.

With that in mind, here are some key points to consider when formulating rewards:

- **Remember – "you get what you incentivise, not what you intend".**[17] Rewards need to be constructed with care using specialists from business, consumer psychology, risk and other disciplines. This will be a multi-disciplinary effort.
- **There's a delicate balance between too few rewards and too much reward-shaping**. Reward as the agent nears the goal. However, only reward for behaviour directly attributable to the agent.
- **Rewards should be traceable** to what the agent observes in the state/environment. An agent needs to understand why a certain reward was achieved, which is why the input from the environment should be clear.
- **Make sure the reward function is temporally relevant** – the right reward at the right time.

- While working on reward shaping, **build a function where short-term rewards are related to the longer-term objective**. There are other methods to overcome sparse rewards. For example, in a paper on curiosity driven exploration[18], the agent was trained to look at the right parts of the environment to maximise chances of achieving the objective. This potentially prevents the RL agent from learning suboptimal behaviours and not learning how to achieve the main goal. The main focus is to formulate a reward function that motivates exploration of areas that are good for the long-term objective and that stops the agent from getting stuck in unnecessary suboptimal exploration.

Summary


Reward actions that link with the final objective.


Reward informed exploration – based on priors.


Weigh in on shorter term vs. longer term goals.


You may progressively increase the penalty if the agent repeats undesirable behaviour – another complexity of working with reward function.

6. **The following tips will reduce training time and increase efficiency:**

   - In addition to breaking the problem into simpler goals, also **train the algorithm in a phased manner by including easier examples at the start, then gradually including harder examples in the training data set**. This approach to training is called Curriculum Learning (CL).[19] A good analogy is how children are taught in schools: lessons move from easier to harder. Another type of training approach is Apprenticeship Learning (AL), where the agent learns by observing a human subject matter expert perform the task. In a paper, titled Curriculum Learning, Yoshua Bengio discusses the approach in detail and suggests that cleaner examples may yield better generalisations faster. Also, that the gradual introduction of more difficult examples speeds up online training.[20] Both these approaches have their advantages and disadvantages. The application of either depends on what the use case is. For example, if we want to automate a task where the current human-led process is exactly what we want, then AL is potentially better. But if we want to innovate over a current human-led process, then CL may be the way forward.
   - **Ensure rewards are clear** and not proxies for the true objective.
   - **Train the model offline**. Warm start by using learnings and priors from other analytics including targeting models, segments, etc.

7. It is likely that there won't be sufficient individual customer interaction data to train the RL agent effectively. Therefore, **utilise segment level and lookalike data to jump-start the model**. As individual customer data builds up, train the model to personalise further.

8. **Recognise that too much exploration or** exploitation is suboptimal.

9. **Train the algorithm using experience replay**[21] – make sure older experiences are represented in the training data in addition to recent experiences. This will stop the agent from suffering "catastrophic forgetting".[22]

10. **Test the models** using a simulated environment[23] before they are deployed into production.

Controlling the learning environment

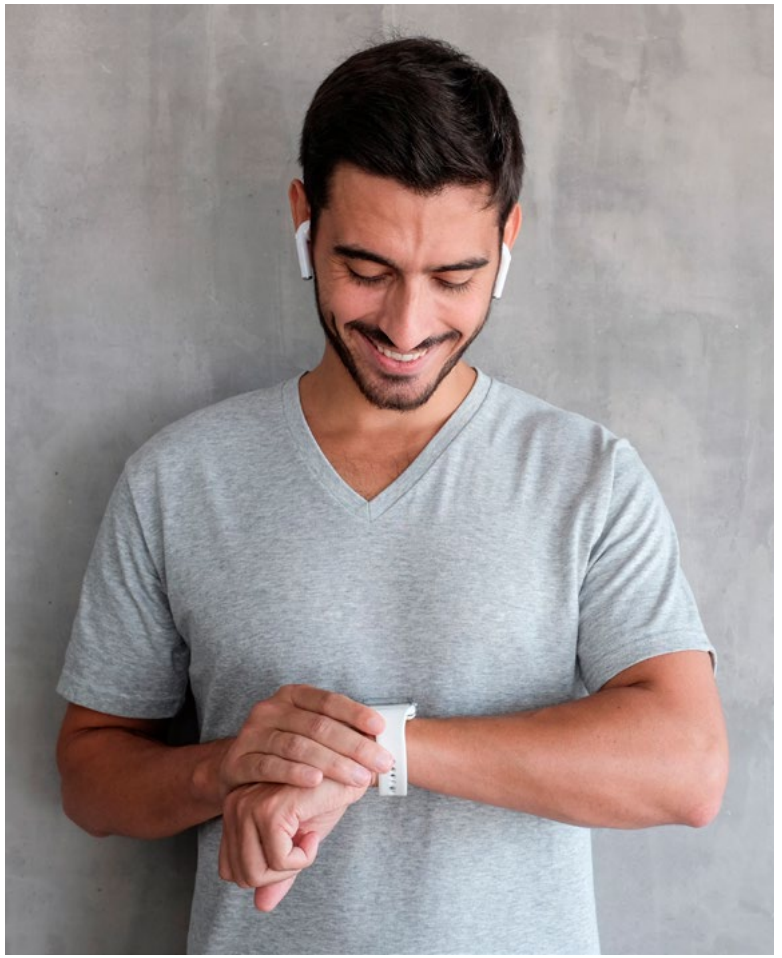
As already noted, we cannot deploy an RL-powered agent for live customer interactions if it has not been sufficiently trained and tested in a development environment. The best practice covered in this section helps to train an RL algorithm in an offline[24] batch mode with historical data and other inputs, such as from subject matter experts, before it is deployed for customer interaction.

> "In commercial digital applications such as in retail banking, the number of actions the agent can take will expand as banks introduce newer customer touchpoints."

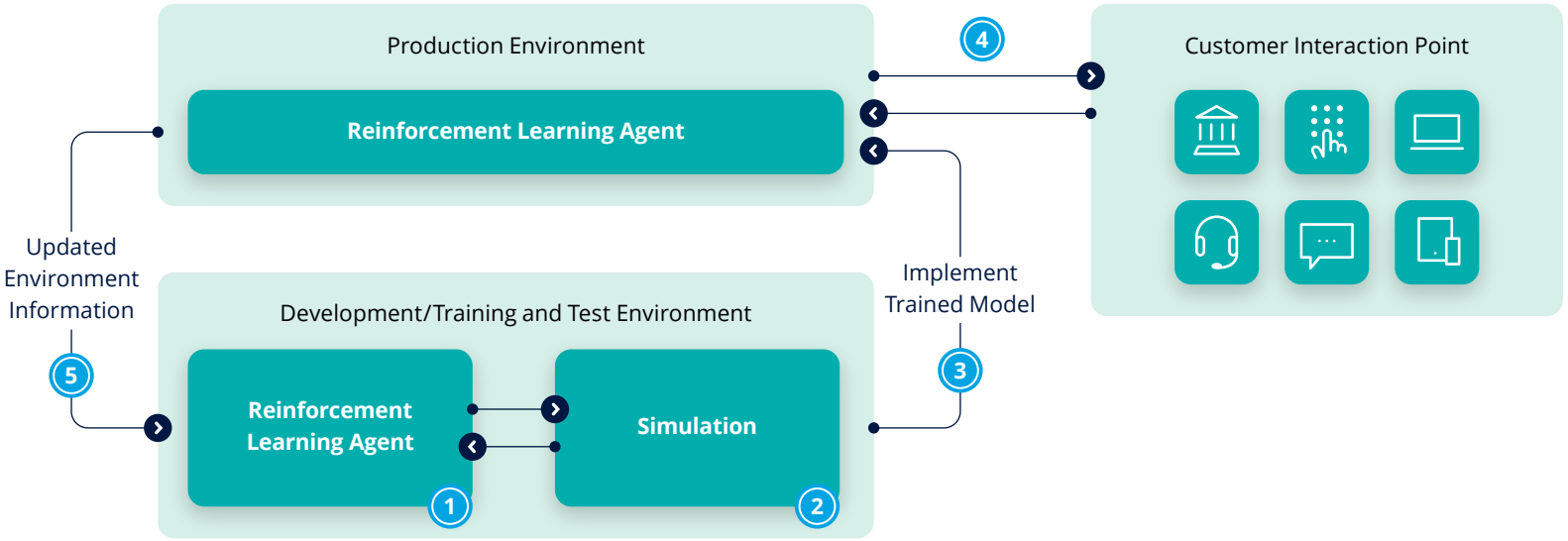
However, once the agent is deployed it should be allowed to learn (perturbation) within certain constraints – such as limited access to interaction data when learning live so that exploration is restricted. In this way, we restrict a poor customer experience or a risky/unapproved decision that the agent may take.

It follows that careful consideration has to be given to the design of this process of deployment, particularly when deciding at what point the algorithm is good enough to be transferred to the production/live environment, and what learning objectives are allowed in the live environment. Additionally, in commercial digital applications such as in retail banking, the number of actions that the agent can take will expand as banks introduce newer customer touchpoints, offers and services.

This is unlike the typical RL applications where the action set is constant. The growing list of actions poses a challenge to traditional RL as expensive retraining of the whole system may be required every time a new feature is introduced by the bank. In addition to the costs of retraining you also don't want exploration to restart from scratch every time a new action is introduced. This can be resolved by mapping available features (actions) to classes of similar features. For example, both a birthday message and a wedding anniversary message are anniversaries. They are more similar to each other than, say, to an account balance message. Put simply, training the RL agent on classes of actions and other latent variables that define actions, helps the RL algorithm to generalise learning across actions within that class. This makes it possible to introduce newer actions in that class. For the interested reader, we recommend a paper by researchers from University of South California[25] where the topic of generalising actions in the context of reinforcement



The process of training, testing and deployment



1	The Reinforcement Learning Agent is trained in batches following a curriculum. A curriculum-based approach is used with the agent learning easier questions before being retrained with harder questions.
2	A simulation is used to test the model in the development environment.
3	The trained model is then pushed to the production environment.
4	Agent interacts with customer in live production environment via a customer interaction point (e.g. phone, webchat, ATM, etc.)
5	On a regular basis, feedback received in the production environment (e.g. customer responses) are used to update the development environment's information and the process begins again.

Example Environment Information

Domain Knowledge & Fundamentals
Prior probabilities and segmentations, sampled distributions, survey feedback

Core Banking Data
Customer information, product information, demographics

Interactions Data
Inbound – Requests and enquiries, complaints, feedback
Outbound – marketing, query resolution, service messages

learning in discussed in more detail.

Now that we have briefly touched upon different aspects for offline/online training of RL algorithms, let's explore how to scale RL in a live environment. In the next section, we will introduce key technical components that need to be considered to put RL into production.

Please note: *This section is for the technical reader and can be skipped if this information isn't relevant to you.*

Productionalising reinforcement learning in banking

Generally, there are 5 important parts to the machine learning (ML) engineering production design:

1. Data processing/feature engineering
2. Model training/re-training
3. Model deployment
4. Model Serving
5. Model Monitoring

Bringing these together in a continuous process is what constitutes the production system.

The key components of ML and RL are similar but there are subtle differences that we will point out in the following process. We will also lay down the key components and design consideration for each stage of the journey.

Data processing and feature engineering

This consists of two parts, feature engineering on historical data for training and incremental feature for model serving and model re-training.

The key dependency of feature engineering is data ingestion from various data sources or an enterprise data warehouse. In retail banking, this will be various data which are necessary to define the state of the customer, previous actions and associated utility from the customer.

These include transactions from different products, account information of different products, as well as in-bound and out-bound interaction through web, email, mobile, etc.

To process this information, it's necessary to have event-driven architecture for dynamically changing data. All the data sources need to be connected through a common pipeline designed to ingest data at one place using streaming technologies such as Kafka[26], RabbitMQ and Pulsar. There are also cloud-managed versions of these tools, like AWS Kinesis, AWS MSK, and Google's native Pub/Sub.

Processing historical data for training

This is done on distributed architecture with components like Apache Spark[27]. Again, depending on various factors such as availability of data and resources, this could be executed on-cloud or on-premise.

The cloud pipeline would typically include open-source tools like Apache Beam[28] and Apache Spark. Apache Spark integrates well with Hadoop and various big data formats. This pipeline could be run on-premise for the purpose of training.

"The key dependency of feature engineering is data ingestion from various data sources or an enterprise data warehouse."

Computing features in near real-time for model serving

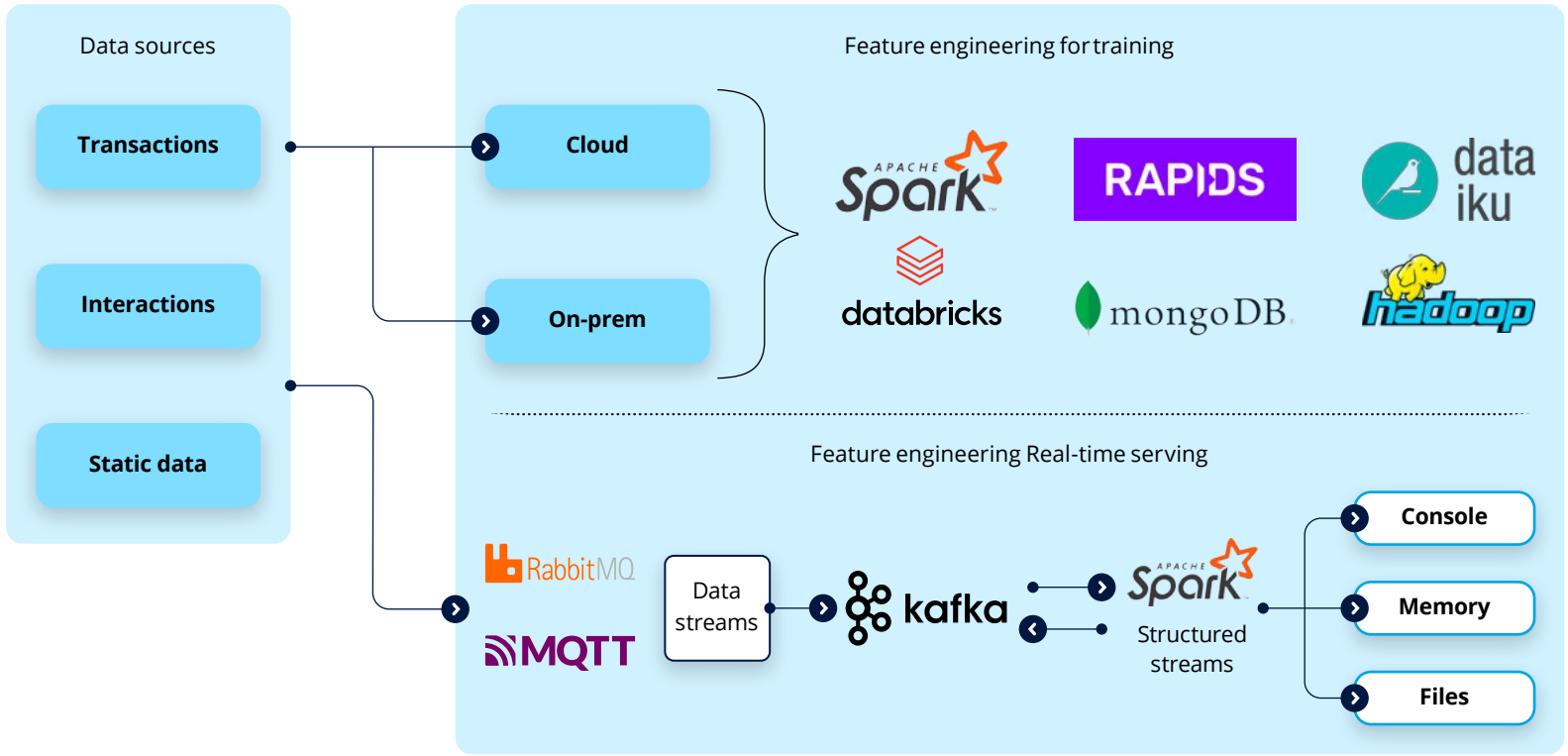
This is probably the most complex part of process, mainly because of the processing time involved. The RL agent requires information about the changing environment: the change in state after a transaction, product uptake or upgrade, negative feedback, etc.

Model training and re-training

An event-driven infrastructure is a key requirement for an RL-based system to work. This can be deployed on-premise, on-cloud, or in a hybrid environment.

Given the requirement of constant/near real-time learning (re-training) in a business application with potentially millions of customers and millions of interactions, Graphical Processing Units (GPU)[29] are a necessity.

As already mentioned, RL-based model development is computationally intensive as well as iterative and the cost of training the algorithms, both in batch as well as in online mode, needs to be considered carefully. When looking at the business case of deploying such technology, there's a choice of different approaches – including the hybrid approach of training and re-training on-premise, and live on-cloud deployment.

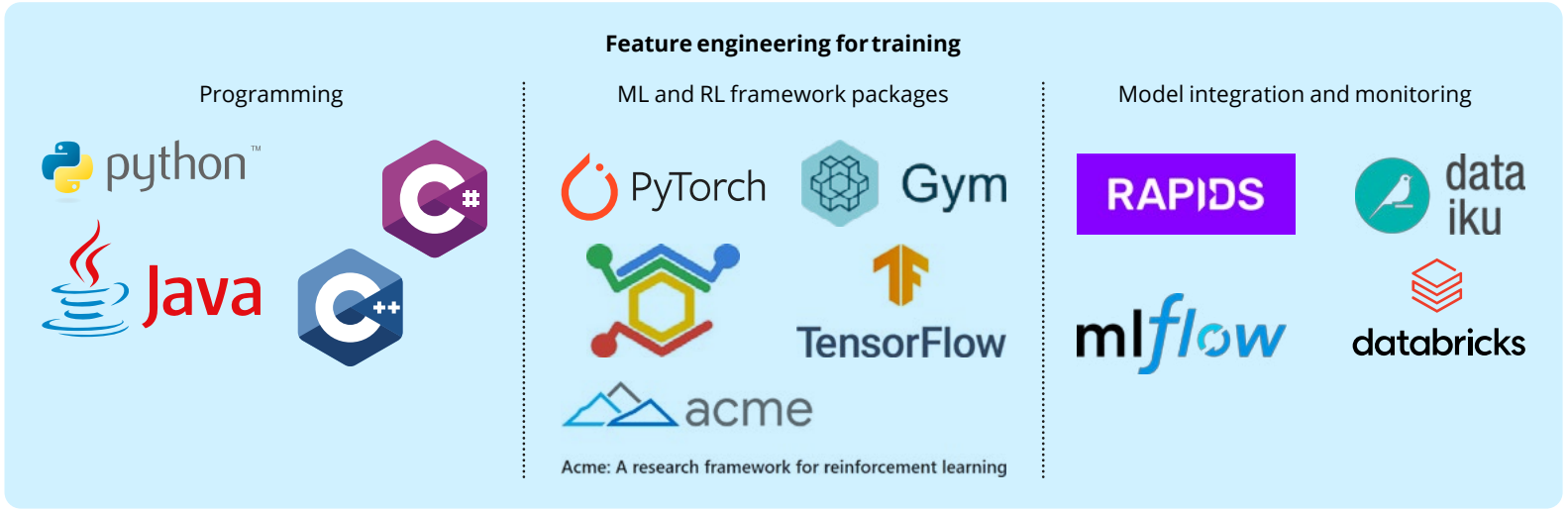


The right tools

Apart from the GPU stack, the key architectural components in this event-driven architecture are the pipelines that capture and transfer data from different source systems, or feature stores to the compute environment.

A typical toolset for training would consist of:

- **Programming language** – Python, C++, Java, C#. Python with its ML package ecosystem is highly preferred for prototyping, as well as for production. Packages like Pytorch also provide C++ binding.
- **RL/ML packages** – There are few choices in this area as well. Tensorflow and Pytorch are well known for supporting modern, deep RL frameworks. Open-source frameworks such as Rapids integrate these distributed learning (DL) frameworks with GPU-based compute.
- **Model monitoring and versioning** – Although we can implement a proprietary model monitoring component. There are many open-source frameworks as well cloud native frameworks which are tailor-made for model monitoring and version control.



Model serving

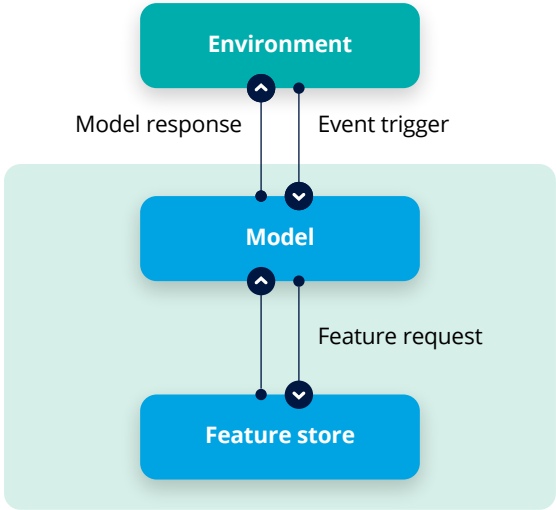
In supervised or unsupervised learning, once the model is deployed it only has to output the inference/prediction for any new observation. However, to produce its next action the model does not need to track this last observation and consequent prediction.

Once an RL model is trained offline (batch) and is deployed to interact with individual users (e.g., customers), it will act as an individual agent for each user, resulting in hyper-personalisation.

This means that an individual RL agent needs to track any new data it observes and action it takes in near real-time for each user. Since in production many such individual RL agents will each be taking individual actions for their respective users, the cumulative data on the actions of all the agents will have to be simultaneously recorded to be used for future batch trainings.

A typical stack for deployment with a scalable infrastructure is required to manage the serving load for such sophisticated models.

The diagram below is a simple representation of what an RL workflow looks like.



The next section will help us visualise the differences that contemporary architectural designs have vis-à-vis this simple representation, especially in the model component.

Contemporary designs for distributed learning in RL and model serving

You will recall that in digital applications such as Retail banking, the RL-powered system will:

- **Act as individual agents** for respective customers
- **Not have sufficient individual customer interaction data** to train the RL agent effectively for that customer
- Receive feedback from individual customers at a different pace

So, how do we efficiently train and deploy an RL-powered system that is not only capable of serving multiple customers but also capable of learning and generalising from customer feedback **asynchronously**?

Research groups from technology companies and academia have been working to solve this problem. We have briefly touched upon work carried out by groups at Alphabet (Deepmind and Google AI)[30, 31] to show example architectures that can be useful in retail banking digital applications. In these studies the RL model has been divided into actors and learners.

The actors are defined as data generators and in our case, these are the individual agents interacting with their respective customers. The learners learn from all the data generated by the actors at any given moment and update the RL model parameters to improve the model.

Note that the learnings of this newly updated model can be generalised across different customers irrespective of individual customer responses or lack thereof. This learning process is also asynchronous, which means that the RL model does not have to wait for all the customers to feedback before it retrains.

DeepMind's Acme framework

Figure 1. Represents the classical setup single actor and learner

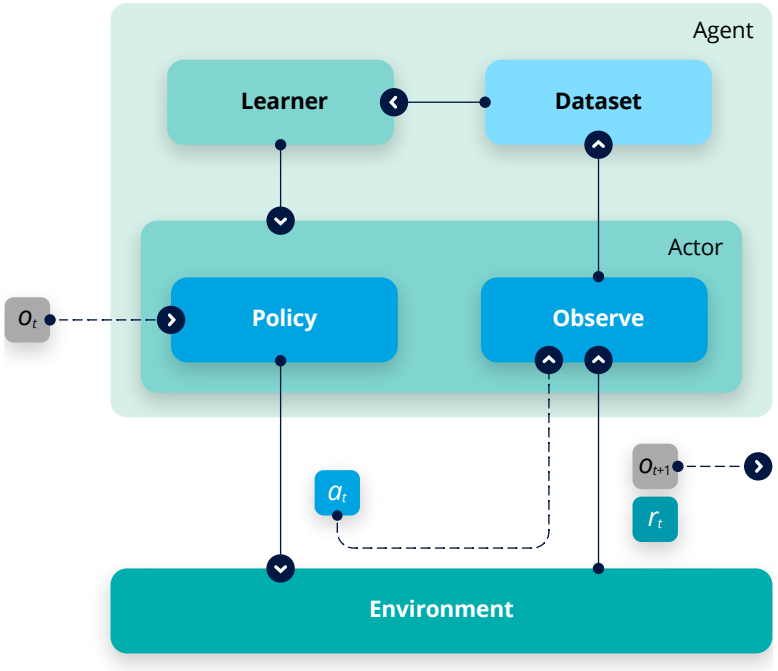
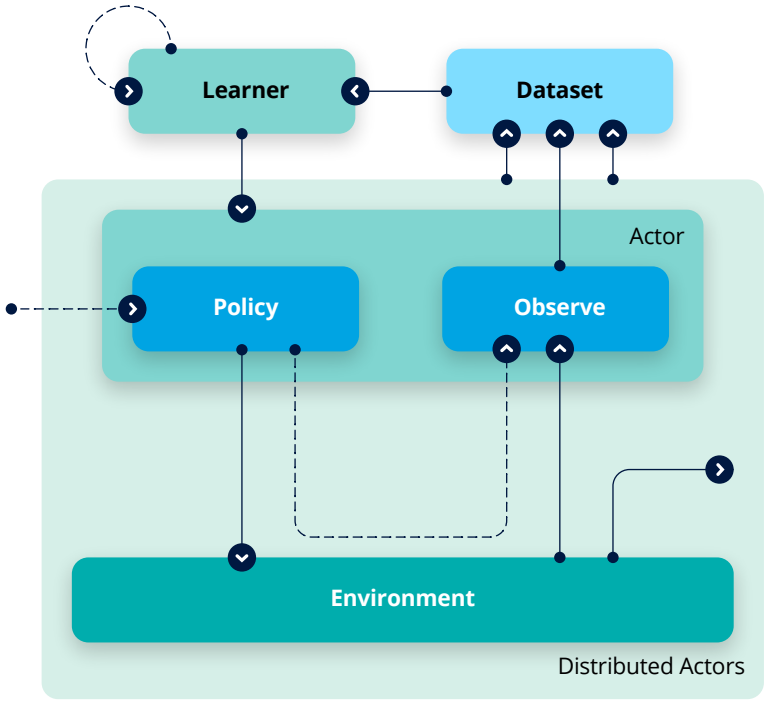


Figure 2. Represents the distributed setup with multiple actors interacting with a learner asynchronously



We have reproduced these from https://deepmind.com/research/publications/Acme

Google's Seed framework

Figure 3: Actions taken by different actors sent as an update from the learner

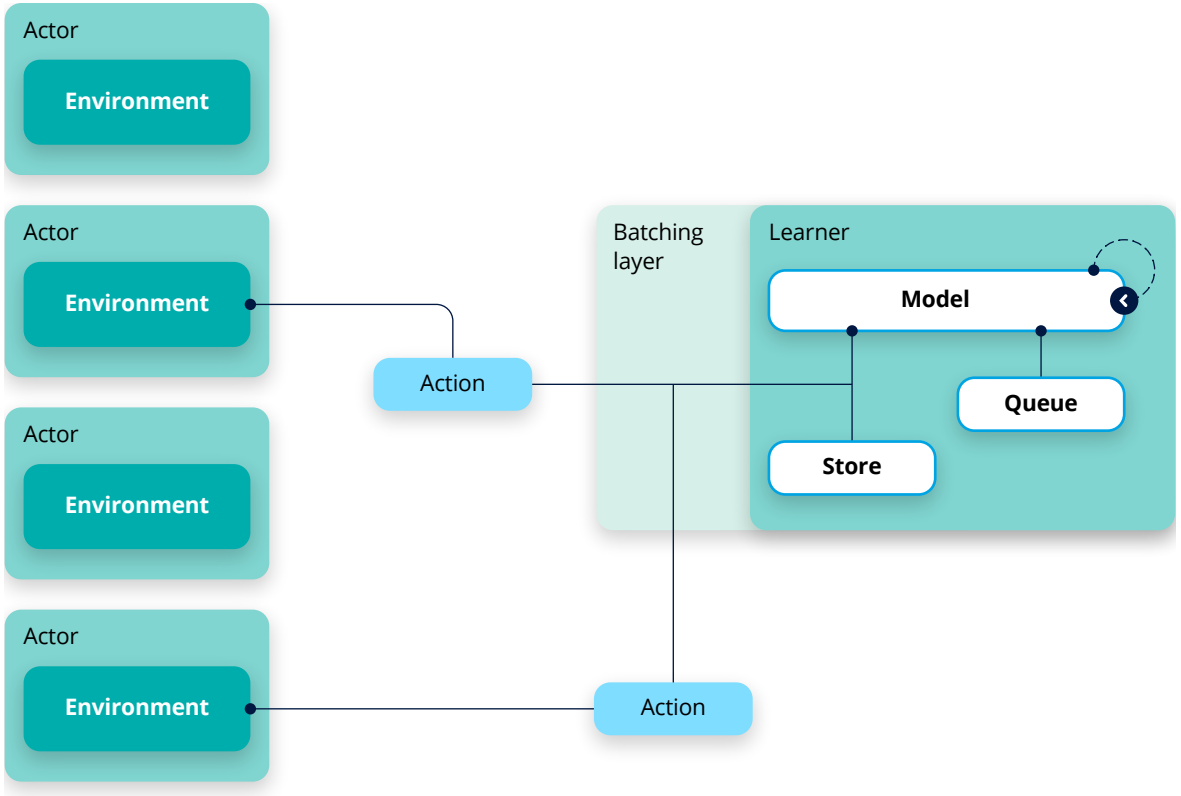
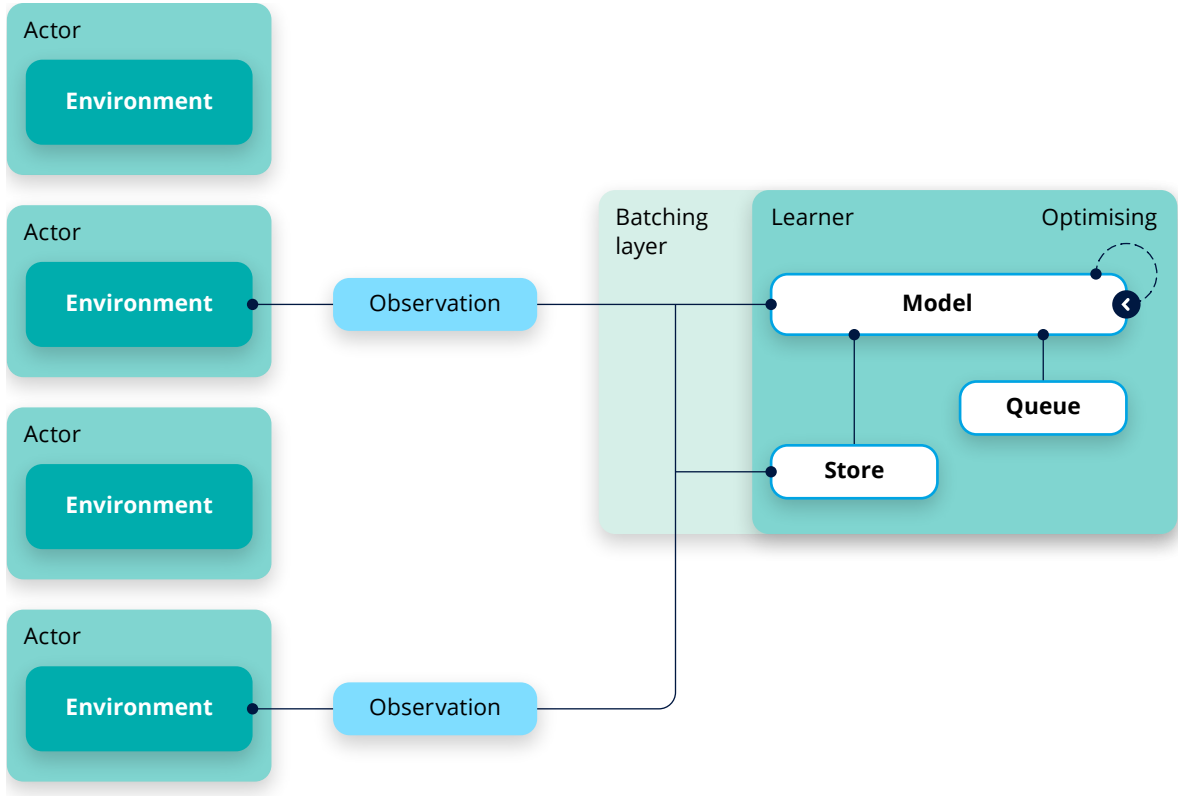


Figure 4. Represents the observation generated by the actor's interaction with the environment feeding into the learner



We have reproduced these from https://ai.googleblog.com/2020/03/massively-scaling reinforcement.html

Risks and governance

As RL models have a live learning component, these present a whole new level of risk, regulatory and legal considerations – especially in financial services.

Additionally, as RL agents powering digital apps are trained on data generated through the interaction with customers, this presents its own challenges. Through such interaction data, app users may reveal hidden vulnerabilities or unconscious biases – such as when they respond to communications out of fear, or if they identify a risk. The RL agent may learn to maximise engagement by exploiting these communications, showing content that it knows will prompt a customer to use the app.

There is a growing awareness of this 'vulnerability and bias mining' – an issue underlined by recent experiences with social media.[32]

As most organisations digitise and embed AI, like RL, within apps to manage customer engagement and personalisation, they will invariably face the challenge of how to control their algorithms from learning psychological or primordial vulnerabilities and biases. This is in addition to other biases that creep into data used for modelling.

Careful consideration

The point to bear in mind throughout these implementations, is that the teams which build these models generally do not have psychologists as members. Appropriate safeguards need to be in place to prevent these models from introducing serious risks across all industries by miss-selling – unwittingly exploiting consumer fears and shopping addictions.

The process to manage the risks associated with RL and to govern such systems efficiently, needs careful consideration. From employing multidisciplinary teams to design and develop unbiased models, to applying appropriate human control on the system, while restricting the majority of the learning to offline mode, will all help to mitigate risk. as will a multi-tiered governance process with live tracking of model performance. We will cover such strategies in detail in a follow-up to this paper.

"Appropriate safeguards need to be in place to prevent these models from introducing serious risks across all industries."

Sources

01. What Consumers Expect When it Comes to Your Digital Experience (Survey)- https://www.contentstack.com/blog/all-about-headless/what-consumers-expect-digital-experience-survey/#:~:text=Everyone%20Expects%20a%20Consistent%20Digital,features%20as%20their%20desktop%20website.

02. Lasting lockdown habits: a new digital consumer? - https://www2.deloitte.com/uk/en/pages/technology-media-and-telecommunications/articles/digital-consumer-trends-lockdown-behaviour.html

03. 2021 banking and capital markets outlook - https://www2.deloitte.com/us/en/insights/industry/financial-services/financial-services-industry-outlooks/banking-industry-outlook.html

04. Data Is Great — But It's Not a Replacement for Talking to Customers - https://hbr.org/2021/03/data-is-great-but-its-not-a-replacement-for-talking-to-customers

05. Forging New Pathways: The next evolution of innovation in Financial Services -https://www.weforum.org/reports/forging-new-pathways-the-next-evolution-of-innovation-in-financial-services

06. The future of retail banking - The hyper-personalisation imperative -https://www2.deloitte.com/content/dam/Deloitte/uk/Documents/financial-services/deloitte-uk-hp-the-future-of-retail-banking.pdf

07. HSBC Sees Hyper-personalization in Banks' Future - https://www.cdotrends.com/story/14529/hsbc-sees-hyper-personalization-banks%E2%80%99-future

08. "Reinforcement Learning for Recommender Systems: A Case Study on Youtube," by Minmin Chen - https://www.youtube.com/watch?v=HEqQ2_1XRTs

09. Cross Channel Optimized Marketing by Reinforcement Learning - https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.2279&rep=rep1&type=pdf

10. Reinforcement Learning to Optimize Long-term User Engagement in Recommender Systems - https://dl.acm.org/doi/10.1145/3292500.3330668

11. RECSIM: A Configurable Simulation Platform for Recommender Systems - https://arxiv.org/pdf/1909.04847.pdf

12. Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto -https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

13. AlphaGo - https://deepmind.com/research/case-studies/alphago-the-story-so-far

14. @misc{rlblogpost, title={Deep Reinforcement Learning Doesn't Work Yet}, author={Irpan, Alex}, howpublished={\url{https://www.alexirpan.com/2018/02/14/rl-hard.html year={2018}

15. Deep Reinforcement Learning in Production at Zynga | Patrick Halina & Mehdi Ben Ayed - https://towardsdatascience.com/deep-reinforcement-learning-in-production-at-zynga-334cd285c550

16. Deep Reinforcement Learning in the Enterprise: Bridging the Gap from Games to Industry - https://www.youtube.com/watch?v=GOsUHlr4DKE

17. Deep Reinforcement Learning Models: Tips & Tricks for Writing Reward Functions - https://medium.com/@BonsaiAI/deep-reinforcement-learning-models-tips-tricks-for-writing-reward-functions-a84fe525e8e0

18. Curiosity-driven Exploration by Self-supervised Prediction - https://arxiv.org/pdf/1705.05363.pdf

19. @article{weng2020curriculum, title = "Curriculum for Reinforcement Learning", author = "Weng, Lilian", journal = "lilianweng.github.io/lil-log", year = "2020", url = "https://lilianweng.github.io/lil-log/2020/01/29/curriculum-for-reinforcement-learning.html"}

20. Curriculum Learning - https://www.researchgate.net/publication/221344862_Curriculum_learning

21. Revisiting Fundamentals of Experience Replay - https://arxiv.org/pdf/2007.06700v1.pdf

22. Experience Replay for Continual Learning - https://openreview.net/pdf/55439317e3a70c6f158499eb0f211362d6d3a37a.pdf

23. RecSim NG: Toward Principled Uncertainty Modeling for Recommender Ecosystems - https://arxiv.org/pdf/2103.08057.pdf

24. Offline Reinforcement Learning by Sergey Levine - https://www.youtube.com/watch?v=qgZPZREor5I

25. Generalization to New Actions in Reinforcement Learning - https://arxiv.org/pdf/2011.01928.pdf

26. Real-Time End-to-End Integration with Apache Kafka in Apache Spark's Structured Streaming - https://databricks.com/blog/2017/04/04/real-time-end-to-end-integration-with-apache-kafka-in-apache-sparks-structured-streaming.html

27. https://spark.apache.org/

28. https://beam.apache.org/

29. How GPUs Can Democratize Deep Reinforcement Learning for Robotics Development - https://blogs.nvidia.com/blog/2020/12/10/deep-reinforcement-learning-gpus-robotics/

30. Acme: A new framework for distributed reinforcement learning - https://deepmind.com/research/publications/Acme

31. Massively Scaling Reinforcement Learning with SEED RL- https://ai.googleblog.com/2020/03/massively-scaling-reinforcement.html

32. Taming the machine: AI Governance - https://www.itpro.co.uk/technology/artificial-intelligence-ai/359374/taming-the-machine-ai-governance

Additional sources that have been referred to during the writing of this paper are below (In no particular order)

33. The Retail Customer Digital Journey, Navigating the regulatory hotspots - https://www2.deloitte.com/uk/en/pages/financial-services/articles/retail-customer-digital-journey.html (Risks)

34. Taming the Noise in Reinforcement Learning via Soft Updates -https://arxiv.org/abs/1512.08562

35. Sequential cost-sensitive decision making with reinforcement learning -https://doi.org/10.1145/775047.775086

36. Machine Learning in Finance – Mathew F. Dixon, Igor Halperin, Paul Bilokon

37. Structured Streaming + Kafka Integration Guide (Kafka broker version 0.10.0 or higher) - https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html

38. MLOps: Continuous delivery and automation pipelines in machine learning -https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

39. Decisions from Data: How Offline Reinforcement Learning Will Change How We Use Machine Learning – https://medium.com/@sergey.levine/decisions-from-data-how-offline-reinforcement-learning-will-change-how-we-use-ml-24d98cb069b0

40. Evolved Policy Gradients – https://storage.googleapis.com/epg-blog-data/epg_2.pdf

41. Hindsight Experience Replay - https://arxiv.org/abs/1611.05397

42. Reinforcement Learning with Unsupervised Auxiliary Tasks - https://arxiv.org/abs/1707.01495

43. Learning from Human Preferences - https://openai.com/blog/deep-reinforcement-learning-from-human-preferences/

44. An EPIC way to evaluate reward functions - https://bair.berkeley.edu/blog/2021/04/20/epic/

45. Can AI model economic choices? - https://www.brookings.edu/research/can-ai-model-economic-choices/

46. Economic reasoning and artificial intelligence - https://science.sciencemag.org/content/349/6245/267

47. Fast reinforcement learning through the composition of behaviours - https://deepmind.com/blog/article/fast-reinforcement-learning-through-the-composition-of-behaviours

48. Understanding dwell time to improve LinkedIn feed ranking - https://engineering.linkedin.com/blog/2020/understanding-feed-dwell-time

49. Optimizing agent behavior over long time scales by transporting value - https://www.nature.com/articles/s41467-019-13073-

50. https://simonsinek.com/product/the-infinite-game/

Deloitte.

Designed and produced by 368 at Deloitte. J21094