

# Cyber+AI+Quantum

Raj Sharma


Engineer/Entrepreneur/Researcher, Data and AI Strategy and Architecture , Help StartUps to Scale Data & AI, Course Director at Oxford University for AI and Cyber Security , MBA/Master of Computer Science/Master of Information Security /PhD(Cyber+AI+Quantum)  
Email : [raj.oxford.ai@gmail.com](mailto:raj.oxford.ai@gmail.com)





# Agenda

As **cyber threats evolve**, organizations are exploring **Quantum Computing** and **Generative AI** to enhance **cyber defense strategies**. However, quantum advancements also introduce risks, requiring new security models.



01

**Cyber and AI**

02

**Generative AI**

03

**Quantum + Cyber +  
GenAI**

04

**Q&A**



# Cyber & AI

- A network intrusion detection system based on incremental statistics (AfterImage) and an ensemble of autoencoders (KitNET)

## Kitsune :

- <http://arxiv.org/pdf/1802.09089>
- <https://github.com/ymirsky/Kitsune-py>  
<https://github.com/ymirsky/Kitsune-py>

- Neural networks have become an increasingly popular solution for **network intrusion detection systems (NIDS)**.
- Their capability of learning complex patterns and behaviors make them a suitable solution for differentiating between normal traffic and network attacks.
- However, a drawback of neural networks is the amount of resources needed to train them.
- Meaning that an expert must label the network traffic and update the model manually from time to time.
- Kitsune is a plug and play NIDS which can learn to detect attacks on the local network, without supervision, and in an efficient online manner.
- Kitsune's core algorithm (KitNET) uses an ensemble of neural networks called autoencoders to collectively differentiate between normal and abnormal traffic patterns.

- KitNET is supported by a feature extraction framework which efficiently tracks the patterns of every network channel.
- The paper evaluations show that Kitsune can detect various attacks with a performance comparable to offline anomaly detectors, even on a Raspberry PI.

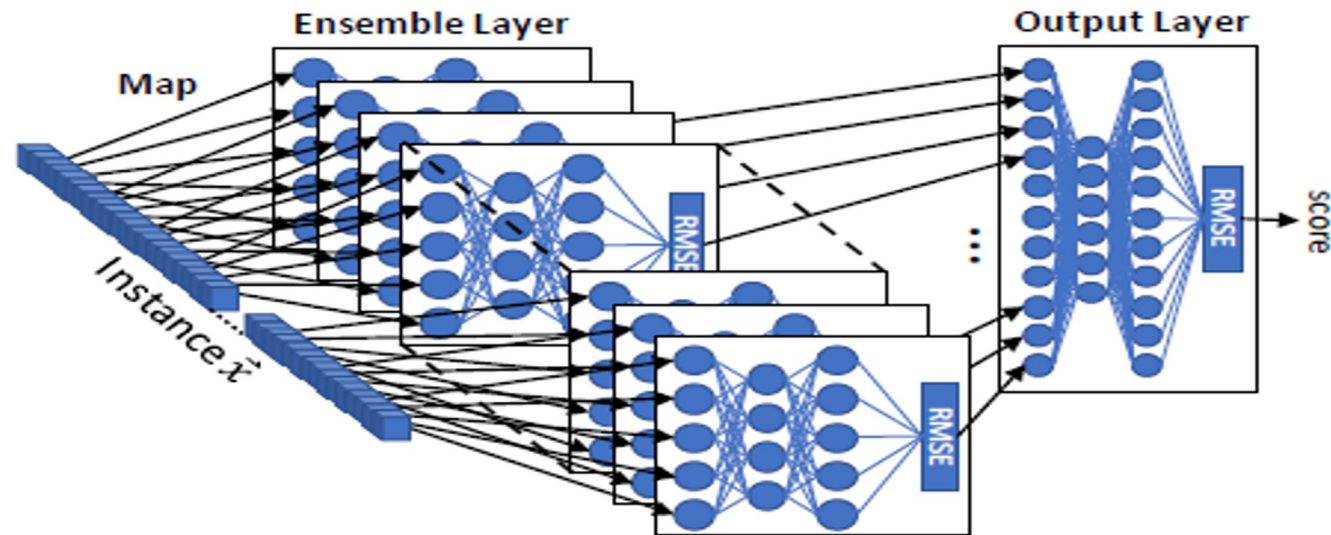


Fig. 1: An illustration of Kitsune's anomaly detection algorithm *KitNET*.

# INTRODUCTION

- An NIDS is a device or software which monitors all traffic passing a strategic point for malicious activities. When such an activity is detected, an alert is generated, and sent to the administrator.
- Conventionally an NIDS is deployed at a single point, for example, at the Internet gateway. This point deployment strategy can detect malicious traffic entering and leaving the network, but not malicious traffic traversing the network itself.

The following shows the typical approach to using an ANN-based classifier in a point deployment strategy:

- 1) Have an expert collect a dataset containing both normal traffic and network attacks.
- 2) Train the ANN to classify the difference between normal and attack traffic, using a strong CPU or GPU.
- 3) Transfer a copy of the trained model to the net-work/organization's NIDS.
- 4) Have the NIDS execute the trained model on the observed network traffic.



- In general, a distributed deployment strategy is only practical if the number of NIDSs can economically scale according to the size of the network.
- It is also not practical to embed the NIDSs directly into inexpensive routers (i.e., with simple hardware).

## **Ideal characteristics of distributed NIDS**

- Online Processing. After the training or executing the model with an instance, the instance is immediately discarded. In practice, a small number of instances can be stored at any given time, as done in stream clustering.
- Unsupervised Learning. Labels, which indicate explicitly whether a packet is malicious or benign, are not used in the training process. Other meta information can be used so long as acquiring the information does not delay the process.

# Kitsune

In this paper, we present Kitsune: a novel ANN-based NIDS which is online, unsupervised, and efficient. A Kitsune, in Japanese folklore, is a mythical fox-like creature that has a number of tails, can mimic different forms, and whose strength increases with experience. Similarly, Kitsune has an ensemble of small neural networks (autoencoders), which are trained to mimic (reconstruct) network traffic patterns, and whose performance incrementally improves overtime.

- A novel autoencoder-based NIDS for simple network devices (Kitsune), which is lightweight and plug-and-play.
- To the best of our knowledge, we are the first to propose the use of autoencoders with or without ensembles for online anomaly detection in computer networks.
- We also present the core algorithm (KitNET) as a generic online unsupervised anomaly detection algorithm, and provide the source code for download

- A feature extraction framework for dynamically maintaining and extracting implicit contextual features from network traffic. The framework has a small memory footprint since the statistics are updated incrementally over damped windows.
- Features Extracted for Kitsune: Whenever a packet arrives, we extract a behavioral snapshot of the hosts and protocols which communicated the given packet. The snapshot consists of 115 traffic statistics capturing a small temporal window into
  - (1) the packet's sender in general
  - (2) the traffic between the packet's sender and receiver. Specifically, the statistics summarize all of the traffic.

- In order to extract the current behavior of a data stream, we must forget older instances.
- The naive approach is to maintain a sliding window of values. However, this approach has a memory and runtime complexity of  $O(n)$ , in contrast to  $O(1)$  for an incremental statistic.
- Furthermore, the sliding window approach does not consider the amount of time spanned by the window. For example, the last 100 instances could have arrived over the last hour or in the last few seconds

TABLE II: The statistics (features) extracted from each time window  $\lambda$  when a packet arrives.

The packet's...	Statistics	Aggregated by	# Features	Description of the Statistics
...size	$\mu_i, \sigma_i$	SrcMAC-IP, SrcIP, Channel, Socket	8	<i>Bandwidth of the outbound traffic</i>
...size	$\ S_i, S_j\ , R_{S_i, S_j}, Cov_{S_i, S_j}, P_{S_i, S_j}$	Channel, Socket	8	<i>Bandwidth of the outbound and inbound traffic together</i>
...count	$w_i$	SrcMAC-IP, SrcIP, Channel, Socket	4	<i>Packet rate of the outbound traffic</i>
...jitter	$w_i, \mu_i, \sigma_i$	Channel	3	<i>Inter-packet delays of the outbound traffic</i>



- An online technique for automatically constructing the ensemble of autoencoders (i.e., mapping features to ANN inputs) in an unsupervised manner. The method involves the incremental hierarchical clustering of the feature-space (transpose of the unbounded dataset), and bounding of cluster sizes.
- Experimental results on an operational IP camera video surveillance network, IoT network, and a wide variety of attacks. We also demonstrate the algorithm's efficiency, and ability to run on a simple router, by performing benchmarks on a Raspberry PI.

# **III. BACKGROUND: AUTOENCODERS**

- Autoencoders for anomaly detection – reconstruction error
- Autoencoders are the foundation building blocks of Kitsune.
- We note that autoencoders can be deep (have many hidden layers).
- In general, deeper and wider networks can learn concepts which are more complex.  
However, as shown above, deep networks can be computationally expensive to train and execute.
- This is why in KitNET we ensure that each autoencoder is limited to three layers with at most seven visible neurons.

# THE KITSUNE NIDS

- Kisune NIDS covers the packet preprocessing framework, the feature extractor, and the core anomaly detection algorithm.
- Kitsune is a plug-and-play NIDS, based on neural networks, and designed for the efficient detection of abnormal patterns in network traffic.
- It operates by (1) monitoring the statistical patterns of recent network traffic, and (2) detecting anomalous patterns via an ensemble of autoencoders. Each autoencoder in the ensemble is responsible for detecting anomalies relating to a specific aspect of the network's behavior. Since Kitsune is designed to run on simple network routers, and in real-time, Kitsune has been designed with small memory footprint and a low computational complexity.

## Components

Kitsune's framework is composed of the following components:

- Packet Capturer: The external library responsible for acquiring the raw packet ex wireshark.
- Packet Parser: The external library responsible for parsing raw packets to obtain the meta information required by the Feature Extractor. Example libraries: Packet++3, and tshark.
- Feature Extractor (FE): The component responsible for extracting n features from the arriving packets to create creating the instance . The features of describe the packet, and the network channel from which it came.
- Feature Mapper (FM): The component responsible for creating a set of smaller instances (denoted as  $v$ ) from , and passing  $v$  to the in the Anomaly Detector (AD). This component is also responsible for learning the mapping.

- Anomaly Detector (AD): The component responsible for detecting abnormal packets, given a packet's representation
- Since the Packet Capturer and Packet Extractor are not the contributions of this paper, we will focus on the FE, FM, and AD components.
- We note that FM and AD components are task generic (i.e., solely depend on the input features), and therefore can be reapplied as a generic online anomaly detection algorithm. Moreover, we refer to the generic algorithm in the AD component as KitNET.

- KitNET has one main input parameter,  $m$ : the maximum number of inputs for each autoencoder in KitNET's ensemble.
- This parameter affects the complexity of the ensemble in KitNET. Since  $m$  involves a trade-off between detection and runtime performance, the user of Kitsune must decide what is more important (detection rate vs packet processing rate).

# Process flow

The process which occurs when a packet acquired

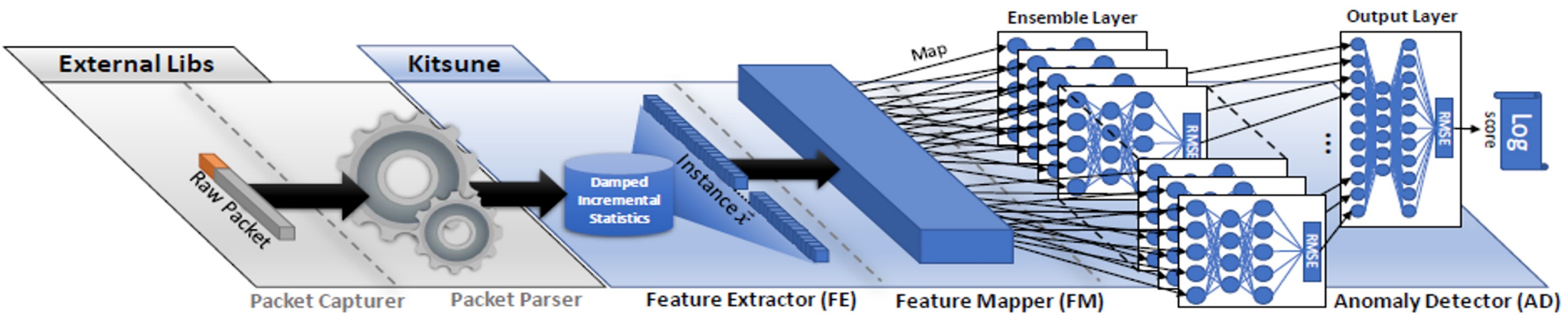


Fig. 3: An illustration of Kitsune's Architecture.



# Feature Extractor (FE)

- Feature extraction is the process of obtaining or engineering a vector of values which describe a real world observation.
- In network anomaly detection, it is important to extract features which capture the context and purpose of each packet traversing the network. For example, consider a single TCP SYN packet. The packet may be a benign attempt to establish a connection with a server, or it may be one of millions of similar packets sent in an attempt to cause a denial of service attack (DoS). As another example, consider a video stream sent from an IP surveillance camera. Although the contents of the packets are legitimate, there may suddenly appear a consistently significant rise in jitter. This may indicate the traffic is being sniffed in a man-in-the-middle attack.

- In network anomaly detection, it is important to extract features which capture the context and purpose of each packet traversing the network. For example, consider a single TCP SYN packet.
- The packet may be a benign attempt to establish a connection with a server, or it may be one of millions of similar packets sent in an attempt to cause a denial of service attack (DoS). As another example, consider a video stream sent from an IP surveillance camera.
- Although the contents of the packets are legitimate, there may suddenly appear a consistently significant rise in jitter. This may indicate the traffic is being sniffed in a man-in-the-middle attack.

- Using a damped window means that the extracted features are temporal (capture the recent behavior of the packet's channel), and that an incremental statistic can be deleted when its dampening weight becomes zero (saving additional memory).
- (A damped window model associates weights with the data in the stream, and gives higher weights to recent data than those in the past.)

- **Feature Mapper (FM)**

The purpose of the FM is to map  $n$  features (dimensions) into  $k$  smaller sub-instances, one sub-instance for each autoencoder in the Ensemble Layer of the AD.

- **Anomaly Detector (AD)**

As depicted in Fig. 3, the AD component contains a special neural network we refer to as a KitNET (Kitsune NETWORK). KitNET is an unsupervised ANN designed for the task of online anomaly detection. KitNET is composed of two layers of autoencoders: the Ensemble Layer and the Output Layer.

We will now detail how KitNET operates the Ensemble and Output Layers.

- 1) Initialization:
- 2) Train-mode:
- 3) Execute-mode:
- 4) Anomaly Scoring:

# Evaluation

- The goal of Kitsune is to provide a light weight IDS which can handle many packets per second on a simple router.
- Given this goal, kitsune was evaluated in detecting attacks in a real IP camera video surveillance network (affecting the availability and integrity of the video uplinks).

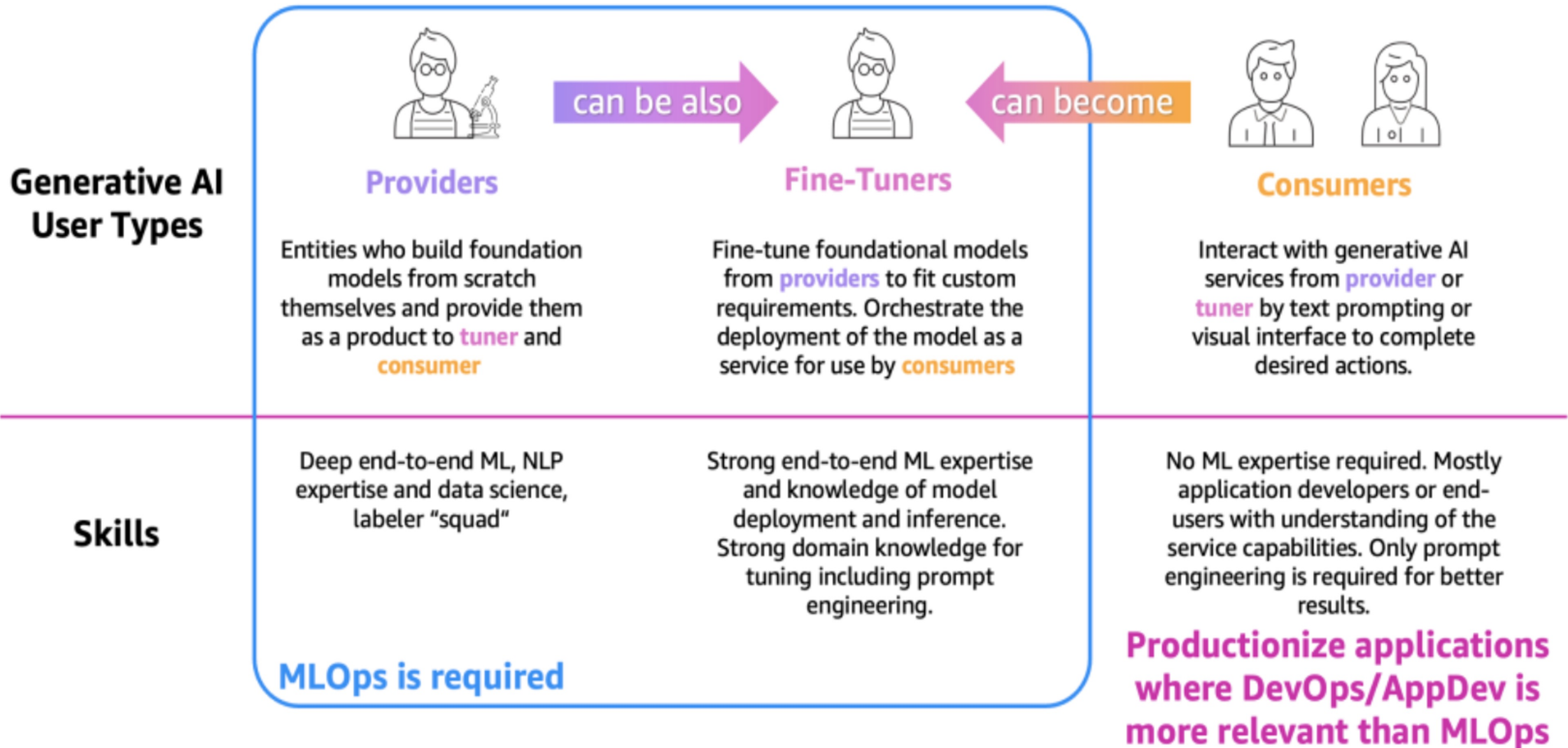
# Evaluation Metrics

- The output of an anomaly detector (s) is a value on the range of  $[0; 1)$ , where larger values indicate greater anomalies (e.g., the RMSE of an autoencoder).
- This output is typically normalized such that scores which have a value less than 1 are normal, and greater than 1 are anomalies.
- The detection performance of an algorithm, on a particular dataset, can be measured in terms of its true positives (T P ), true negatives (T N), false positives (F P ), and false negatives (F N).

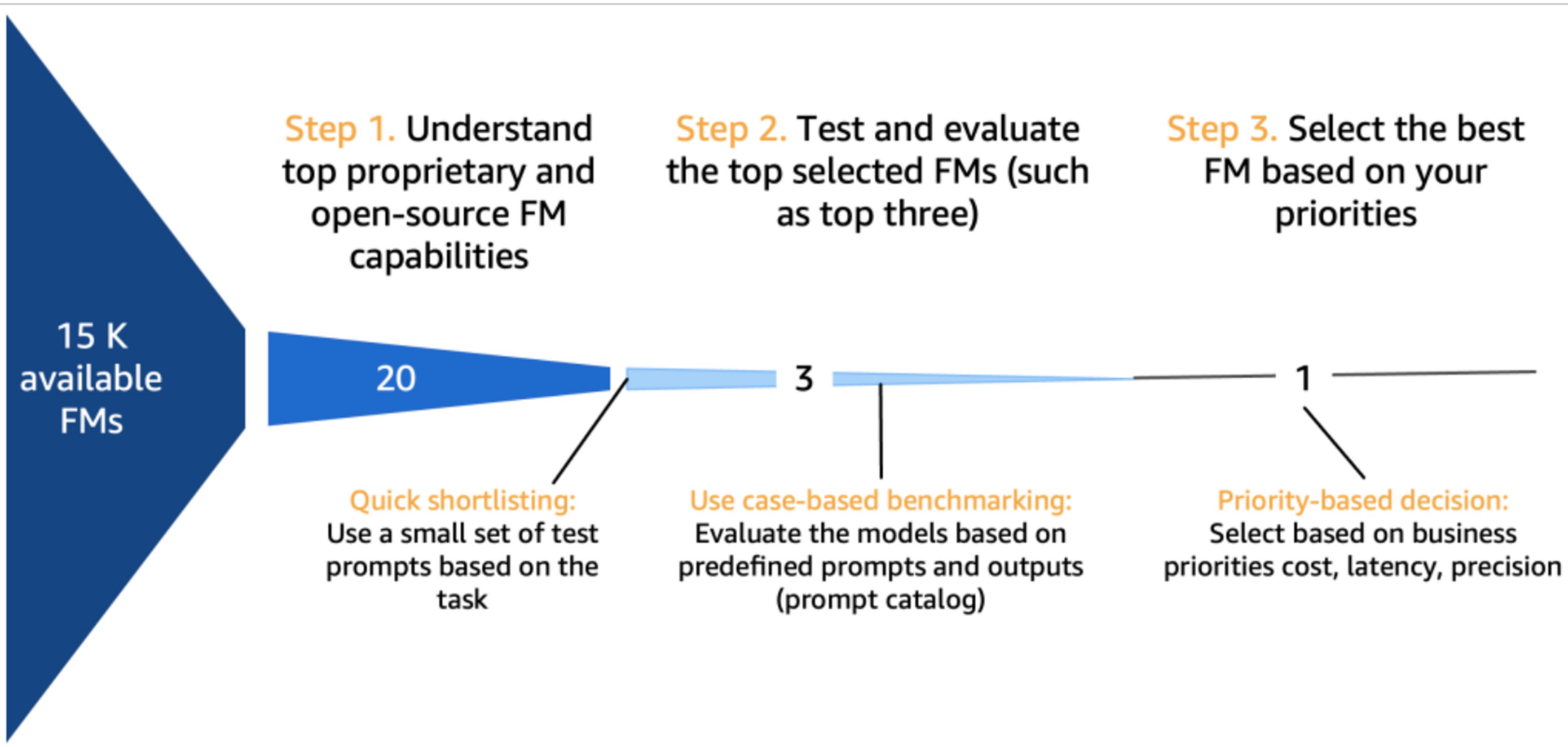
# Generative AI



# Generative AI and differences to Machine Learning Operations - Changing the Future



# Selection of GenAI on basis of Business Priorities



## Open-source Commercial AI Models

Company Name	Model Name	Can be used Commercially	# Params	GPU instance req.	Available on AWS	Speed	Context Window	Trained on	Fine-tunable
AI21	J2 Ultra Instruct	Yes	178 B	p4d.24xl	Bedrock, Jumpstart/SM	-	8 K	Internet Data, Code, Instructions	No
	J2 Mid Instruct	Yes	17 B	g5.12xl	Bedrock, Jumpstart/SM	-	8 K	Internet Data, Code, Instructions	No
	AI21 Summarize	Yes		g4dn.12xl	Jumpstart/SM	-	~13 K	Internet Data, Instructions	No
Amazon	Titan Text Large	Yes	n/a	n/a	Bedrock	-	4 K	n/a	No
Anthropic	Claude	Yes	n/a	n/a	Bedrock	-	12 K	Internet Data, Code, Instructions, Human feedback	No
Cohere	Generate Model Command	Yes	n/a (50 B)	n/a	Jumpstart/SM	-	4 K	Internet Data, Instructions	No
	Generate Model Command-Light	Yes	n/a (6 B)	n/a	Jumpstart/SM	-	4 K	Internet Data, Instructions	No
LightOn	Lyra-Fr 10B	Yes	10 B	g5.12xl	Jumpstart/SM	-	?	Internet Data (French)	No
Stability AI	SDXL	Yes	n/a	g5.xl	Bedrock, Jumpstart/SM	-	-	<Text, Image>	No

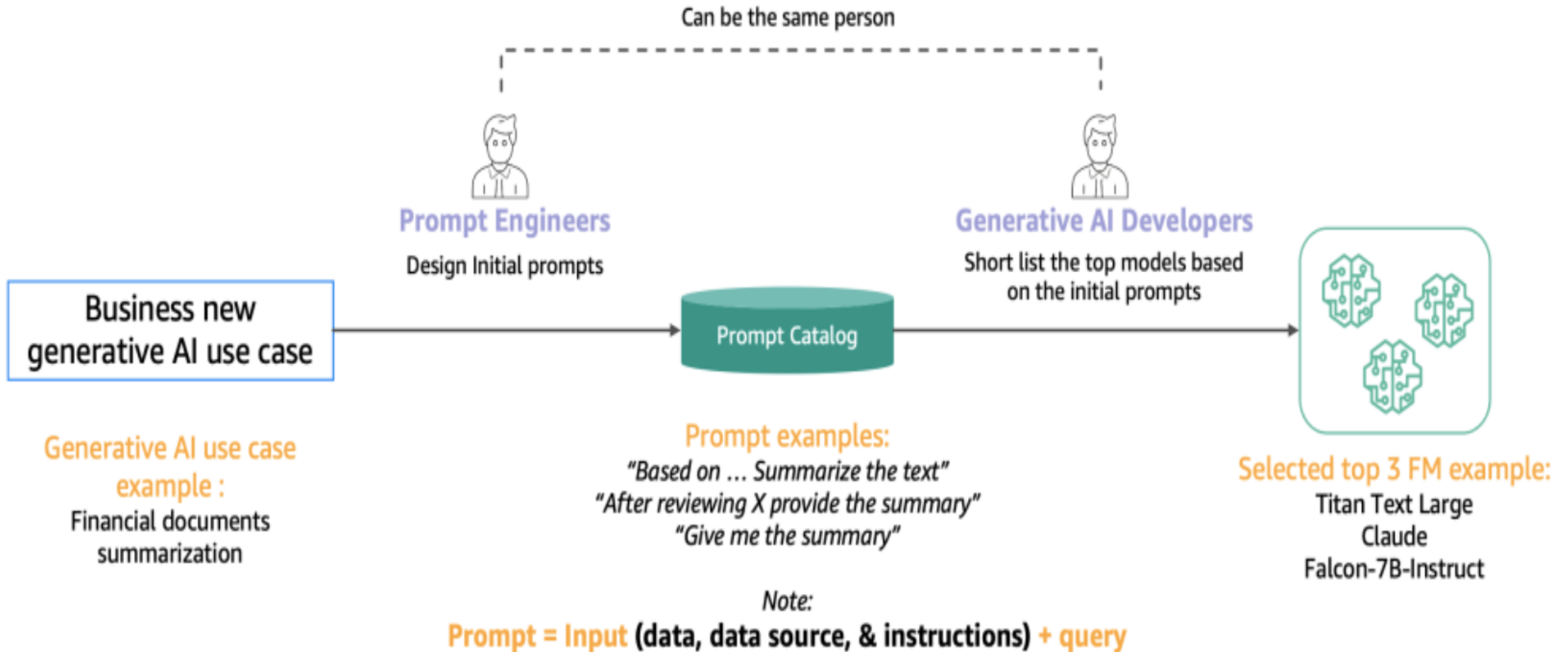
\*Last update July 2023

# Open Source Generative AI Models

Company Name	Model Name	Can be used Commercially	# Params	GPU instance req.	Available on AWS	Speed	Context Window	Trained on	Fine-tunable
Google	FLAN-UL2	Yes	20 B	g5.12xl	Jumpstart/SM	-	2 K	Internet Data, Code, Instructions	Yes
	FLAN-T5-XXL	Yes	11 B	g5.xl	Jumpstart/SM	-	512	Internet Data, Code, Instructions	Yes
Eleuther	GPT-J	Yes	6 B	g5.xl	Jumpstart/SM	-	512	Internet Data, Code	Yes
TII	Falcon-40B-Instruct	Yes	40 B	g5.12xl	Jumpstart/SM	-	2 K	Internet Data, Code, Instructions	Yes
	Falcon-7B-Instruct	Yes	7 B	g5.xl	Jumpstart/SM	-	2 K	Internet Data, Code, Instructions	Yes
BigCode	StarCoder	Yes	15 B	g5.12xl	SM	-	8 K	Code	Yes
	Santa Coder	Yes	1.1 B	g5.xl	SM	-	2K	Code	Yes
LMSYS Org	Vicuna-13B	No	13 B	g5.xl	SM	-	2 K	Internet Data, Code, Instructions	Yes
Meta	Llama-65B	No	65 B	g5.48xl	SM	-	2 K	Internet Data, Code	Yes
Stability AI	SD 2.1	Yes	-	g5.xl	Jumpstart/SM	-	-	<Text, Image>	Yes

\*Last update July 2023

# Generative AI Strategy - Select model based on business case



  
Generative AI Developers & Prompt Engineers/Testers

  
DevOps/AppDevs

Backend

Frontend

1. Select FM or Use Fine-tuned Mode

3. Test & Test Prompt lineage (input & outputs)

5. Input/ Output Filtering

2. Prompt Engineering

4. Chain Prompts & Applications

6. Rating Mechanisms (thumbs up/down, rating, text)

New Test Set

Develop & Deploy Web Application

Input/ Output & Rating Interaction

Test Functionality

Create User Account & Share Data

Fine-tune FM using APIs

Fine-tune Personalized Models

WebUI

  
**Generative AI End-users**  
Use web application rate the quality of output

# Quantum

# Quantum Machine Learning (QML)

## Why Quantum Machine Learning?

- ✓ **Exponential Speedup** → Quantum algorithms can outperform classical ML models for certain tasks.
- ✓ **High-Dimensional Computation** → Quantum states can process complex data structures more efficiently.
- ✓ **Better Optimization** → Quantum-inspired optimization algorithms improve performance in AI.
- ✓ **Quantum Data Processing** → ML models can natively process quantum-generated data



# Core Concepts in Quantum Machine Learning

## 1 Superposition

- Classical bits are **either 0 or 1**, but **qubits can be both at the same time**.
- Allows quantum computers to process **multiple states simultaneously**.

## 2 Entanglement

- Qubits can be **correlated**, meaning the state of one qubit affects another.
- This enables **faster information transfer** across the system.

## 3 Quantum Parallelism

- A quantum system can evaluate **many solutions at once**, enabling faster learning.

## 4 Quantum Interference

- Helps refine computation outcomes by **canceling out incorrect solutions**.

# Quantum Computing & Cybersecurity

Quantum computing presents **both opportunities and threats** to cybersecurity. While **quantum cryptography** offers **unbreakable encryption**, **quantum computers** also threaten existing encryption protocols by breaking widely used cryptographic algorithms.

<b>Quantum Security Advantage</b>	<b>Quantum Security Threat</b>
<b>Quantum Cryptography</b> (unbreakable encryption)	<b>Quantum attacks on RSA &amp; ECC</b>
<b>Quantum Key Distribution (QKD)</b> (secure key exchange)	<b>Grover's Algorithm weakens symmetric encryption</b>
<b>Post-Quantum Cryptography (PQC)</b> (quantum-resistant algorithms)	<b>Shor's Algorithm breaks public-key cryptography</b>

# Quantum Security Solutions

## Quantum Cryptography (Unbreakable Security)

### ✓ Quantum Key Distribution (QKD)

- Uses **quantum entanglement & superposition** to exchange encryption keys.
- Any interception **destroys the data**, making eavesdropping impossible.
- **Example Protocols:** BB84, E91.

### ✓ Quantum-Resistant Cryptography (PQC)

- New cryptographic algorithms that resist **quantum attacks**.
- **NIST Post-Quantum Cryptography Standardization:**
  - **Lattice-based cryptography** (e.g., CRYSTALS-Kyber, NTRUEncrypt).
  - **Hash-based cryptography** (e.g., SPHINCS+).
  - **Multivariate Polynomial Cryptography.**

### ✓ Quantum Random Number Generators (QRNG)

- Uses quantum mechanics to generate **truly random numbers** for encryption keys.

# Cybersecurity Strategies in the Quantum Era

## 1. Migrating to Post-Quantum Cryptography

- Organizations must **replace RSA/ECC** with **quantum-resistant algorithms**.
- Example: **Use CRYSTALS-Kyber instead of RSA** for secure communications.

## 2. Implementing Hybrid Cryptographic Systems

- Combining **classical and quantum-resistant cryptography** to ensure long-term security.

## 3. Securing Critical Infrastructure

- Financial institutions, government agencies, and healthcare must **prepare for "Harvest Now, Decrypt Later" (HNDL) attacks**.
- **Action Plan:** Encrypt sensitive data today using **quantum-safe algorithms**.

# **How Quantum Can Improve Generative AI**

# Faster Training & Optimization (Quantum Machine Learning)

 **Problem: Current AI training is slow & costly**

- Training models like **GPT-4 or DALL-E** requires **massive GPU clusters and weeks of processing**.
- **Quantum computers** can speed up matrix multiplications and gradient calculations.

 **Solution: Quantum-Assisted Training**

- **Quantum Variational Circuits (VQCs)** optimize deep learning models **faster than classical methods**.
- **Quantum-Inspired Gradient Descent** improves efficiency.

**Example: Quantum Neural Network (QNN) with Qiskit**

# Quantum-Inspired Optimization for AI Models

💡 **Problem: AI models require fine-tuning of billions of parameters**

- **Hyperparameter tuning** is a major bottleneck in AI development.
- Classical optimization algorithms struggle with **high-dimensional search spaces**.

🔧 **Solution: Quantum Annealing (D-Wave)**

- **Quantum annealers** solve optimization problems **faster than classical computers**.
- **Example:** Optimizing GPT's transformer layers with quantum methods.

✅ **Impact → Better AI performance with fewer resources.**

# Quantum Data Representation for Creative AI

 **Problem: Classical AI struggles with high-dimensional creativity**

- Quantum states naturally encode high-dimensional data.
- AI-generated music, images, and text could become more expressive.

 **Solution: Quantum Generative Models**

- Quantum Boltzmann Machines (QBM) for deep learning.
- Quantum Variational Autoencoders (QVAE) for generative tasks.

 **Impact → More creative & diverse AI-generated content.**



# Quantum Neural Networks (QNNs)




💡 **Problem: Classical Neural Networks reach efficiency limits**

- **Quantum neurons** could replace traditional artificial neurons.
- QNNs explore new AI architectures **beyond classical deep learning**.

🔧 **Solution: Quantum Perceptrons**

# The Future of Quantum-Enhanced Generative AI

 By 2030, Quantum AI could:

-  **Reduce AI training costs** by replacing expensive GPUs with quantum circuits.
-  **Enable real-time AI creativity** (instant AI-generated movies, books, designs).
-  **Revolutionize AI agents** with more intelligent decision-making.

THANK YOU

Email : [raj.oxford.ai@gmail.com](mailto:raj.oxford.ai@gmail.com)

