Confidential ML Inference

simple and safe

avato is a cloud-based software platform enabling simple and safe data collaboration. **avato**'s Confidential ML Inference allows running machine learning (ML) inference in a privacy-preserving and secure way. When performing inference with **avato**, the data and the model are provably kept confidential from all parties, including decentriq and any infrastructure provider.

avato's Confidential ML Inference integrates seamlessly into workflows without compromising speed and scalability. This opens up fundamentally new ways for model owners to utilize and monetize their models while protecting intellectual property of their models and data privacy of their users.

The problem **avato** solves

Organizations are increasingly relying on ML models to support important business decisions. Building well-performing ML models requires substantial resources which makes models valuable intellectual property (IP). Due to this, organizations often are unwilling to share their ML models with third parties.

Inference is the process of using an ML model for making a prediction on input data. To perform inference, both model and input data must be on the same computer. In many cases however, the input data are privacy-sensitive and the data owners are reluctant to share them.

In situations where the model and the input data are coming from different organizations, this leads to problems: Not only the model IP must be protected, but also the input data are sensitive and cannot be shared. Current options fall into two categories and both require one party to compromise on security and privacy.

- Input data shared with model owner - The first option is for the data owner to share their input data with the model owner. This option has privacy risks for the data owner but also data breach risks for the model owner.
- Model shared with input data owner - The second option is to deploy the model on the data owner's premises. In this case, the model owner risks revealing their IP and depends on the data owner having the required infrastructure for performing the inference.

What if you could perform inference on public cloud infrastructure
while keeping model and data provably confidential?

Solution

With the **avato** platform, decentriq offers a model inference framework based on confidential computing using the latest advancements in trusted hardware. **avato** enables organizations to run ML inference with sensitive input data on public cloud infrastructure without ever exposing the model or the input data to any party, including decentriq and any infrastructure provider.

The **avato** platform can be hosted on several public cloud providers or on-premises. Model owners upload ML models through a dedicated Python API, data owners can obtain inference results by using a web application or a Python API as well. Most models supported by the TensorFlow framework can be used. Inference execution time in **avato** is similar to native non-GPU execution. More details about the supported models, execution speed, and available cloud providers can be found below.

The underlying technology

avato uses Intel SGX, an implementation of secure enclaves (SE). Enabled by dedicated hardware design, SEs enable additional security and privacy guarantees:

- They can prove their program identity (program code) to remote users through a process called remote attestation. This proves to users that their data cannot be used for unintended purposes.
- Through dedicated memory encryption and isolation, the memory of SEs is protected from all access, including the operating system. This proves to users that the data within the SE program cannot be accessed even by system administrators.
- Consequently, encrypted data can be processed in the SE while preventing any party, including decentriq and a potential cloud provider, from accessing the plaintext data.

For more information regarding Intel SGX and the security and privacy guarantees avato gives, visit this blog we wrote together with Intel or the official Intel SGX website.

User workflow

Figure 1: Simplified Confidential ML Inference workflow for one data owner and one model owner.

To illustrate how Confidential ML Inference works, we present an example workflow illustrated in **Figure 1**. Alice works for a model-owning company selling predictive services to customers. Alice wants to enable Bob to confidentially use her deep learning model for making predictions on Bob's sensitive input data.

By going through the following steps, Bob can use Alice's model without them having to share their model and input data with each other or anyone else, including decentriq. This workflow example is complemented by a Python API usage example below.

Model owner (Alice)

1. Alice creates an account with decentriq enabling her to create a Confidential ML Inference secure enclave.
2. She selects a cloud provider and the type of machine she wants to run her inference on.
3. Alice downloads the **avato** Python API provided by decentriq.
4. Alice uses the API to log in. She creates the secure enclave and specifies the users who can use her model for inference. In this case, only Bob`s account.
5. Alice goes through the process of remote attestation. As part of this process, she receives an encryption key from the secure enclave which she uses in the next step.
6. Alice selects her TensorFlow model, converts it to the TensorFlow Lite format and encrypts it locally with the key she received before.
7. Alice sends the encrypted model to the secure enclave.
8. Alice shares the unique ID of the secure enclave with Bob.

Data owner (Bob)

1. Bob registers with decentriq. He decides to use the Confidential ML Inference web application instead of the Python API.
2. Bob uses the secure enclave's ID which he received from Alice to connect to it.
3. The web application goes through the process of remote attestation.
4. Bob selects his input data which get encrypted locally with the secure enclave's key he received in the previous step.
5. The encrypted data are sent over to the secure enclave and the inference is executed.
6. Bob receives the encrypted inference result while being sure that his data were only used for performing the inference and have provably been deleted afterward.

Available models and performance

We are using the TensorFlow Lite for Microcontrollers (TFLiteMicro) framework inside **avato** to enable confidential inference. TFLiteMicro is optimized for use on constrained devices without full operating system access and therefore it is ideal for use inside secure enclaves. As the TFLiteMicro framework, our platform expects models to be in the .tflite format. Publicly available converters are used to transform TensorFlow's standard Keras or SavedModel formats.

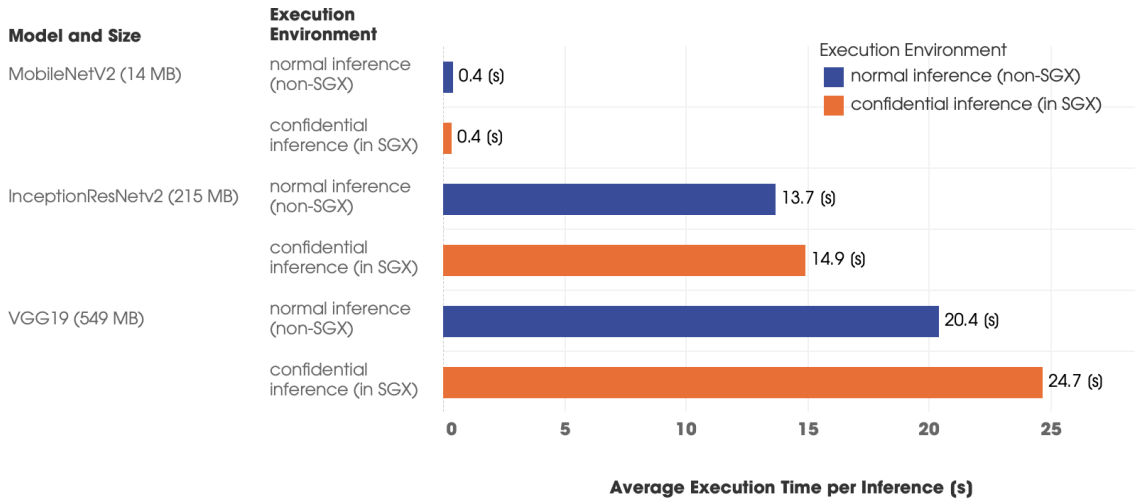


Figure 2: Execution times of three different models tested within **avato** (in SGX) and normally (non-SGX). For each model and execution environment, ten tests were performed using the same hardware specifications.

The **avato** Confidential ML Inference module can run any model that can be run in TFLiteMicro. This includes most "non-exotic" models and we (and Google) are heavily working to support all TensorFlow models.

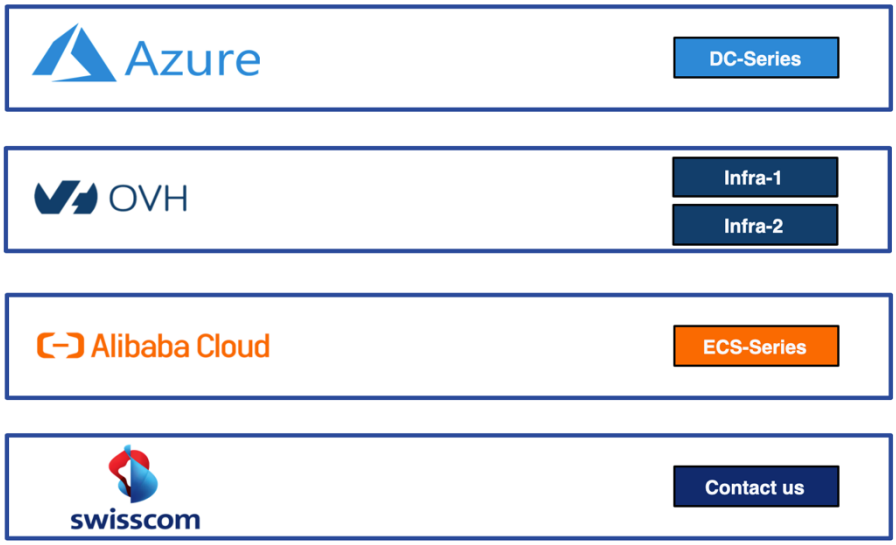
We conducted performance benchmarking to compare the inference speeds between normal and confidential (Intel SGX-based) inference. We used three pre-trained models from the Keras library and did inference on randomly generated inputs of the same size. Model complexity ranged from 3.5 million parameters (14 MB) to 143 million parameters (549 MB). The average execution times for one inference are shown in **Figure 2** per model and execution environment. Clearly, running inference confidentially (in SGX) only moderately slows down the execution (0-21% slower). The observed variance in the results was similar for both execution environments and less than 10% of the average.

Pricing

avato takes care of scaling the cloud infrastructure to the needs of the users. After the selection of the cloud provider and the type of machine, **avato** provides elastic resource provisioning. The **avato** platform reserves as many resources as needed in the cloud to run the service optimally, no matter the size.

Pricing is calculated separately from the cloud infrastructure costs. It is based on a usage per hour model. For more information on pricing, contact us.

Cloud provider options



Contact

For booking your personal demo or getting more information contact

info@decentriq.ch www.decentriq.ch

API example usage

```python
# This is an example workflow of two independent parties, model_owner and inference_user. For simplicity, both are
workflows are shown in the same script while in reality they would most likely be executed on different machines. Of
course, there could be multiple inference users.

# ------------------------------------------------------------------------------------------

# Imports

from avato.client import Client
from avato.secret import Secret
from avato.instance import InstanceType

# ------------------------------------------------------------------------------------------

# A - model_owner creates a new instance and uploads model

# A1) Creating a client and signing in
model_owner_client = Client()
model_owner_client.sign_in("model_owner@gmail.com", "password")

# A2) Creating a new instance
model_owner_instance = model_owner_client.create_instance(
    "InferenceDemo",
    InstanceType.TFLITE_instance,
    ["inference_user@gmail.com"]
)

print(model_owner_instance.id)  # This prints the instance ID, here "39asd-did3a-399aa".

# A3) This step performs the remote attestation by validating the cryptographic proof dubbed "fatquote". This step is
crucial for all security guarantees.
# It gets and validates the cryptographic proof from the enclave:
#
# i)   It proves it is a valid SGX enclave (by checking a certificate).
#
# ii)  It compares the hash of the enclave code to an expected value (to verify what code is running in the enclave).
#
# iii) As part of the proof also a public key is transmitted that allows establishing a secure connection into the
enclave as the private key is only known to the enclave. This public-private key pair is randomly generated for every
enclave and the private part cannot be accessed. Hence data encrypted with the public key can only be decrypted by this
particular enclave.

model_owner_instance.validate_fatquote()

# A4) Creating (randomly) a public-private keypair and setting it.
model_owner_instance.set_secret(Secret())

# A5) Uploading a model in .tflite (aka FlatBuffer) format. Before uploading, the model is encrypted using the enclave
public key from step A3, and the model_owner private key from step A4. The model_owner public key is sent together with
the encrypted data.
model_owner_instance.upload_model("/path/to/model.tflite")

# Note: After this point the instance is usable for the inference_user (before setting the model, an error would be
thrown if the inference_user would try to access it).

# ------------------------------------------------------------------------------------------

# B - inference_user obtains a prediction

# B1) Creating a client and signing in.
inference_user_client = Client()
inference_user_client.sign_in("inference_user@gmail.com", "password")

# B2) Connecting to the instance
instance_id_from_model_owner = "39asd-did3a-399aa"  # This instance id here is assumed to be obtained from the
model_owner. The avato backend also allows querying of all available instance.
inference_user_instance = inference_user_client.get_instance(instance_id_from_model_owner)

# B3) This step performs the remote attestation by validating the fatquote. See A3 above.
inference_user_instance.validate_fatquote()

# B4) Creating (randomly) a public-private keypair and setting it. See A4 above.
inference_user_instance.set_secret(Secret())

# B5) Loading input_data and submitting them for inference. Before uploading, the data are encrypted using the enclave
public key from step B3, and the inference_user private key from step B4. The inference_user public key is sent together
with the encrypted data.
with open("/path/to/input_data.json", "rb") as f:
    input_data = f.read()
inference_result = inference_user_instance.predict(input_data)
```