The Integration of ART & kserve

Team Number: 7

Team Members:

408530003 資管四 范綱彥

408530004 資管四 潘甫翰

408530007 資管四 謝瀞瑩

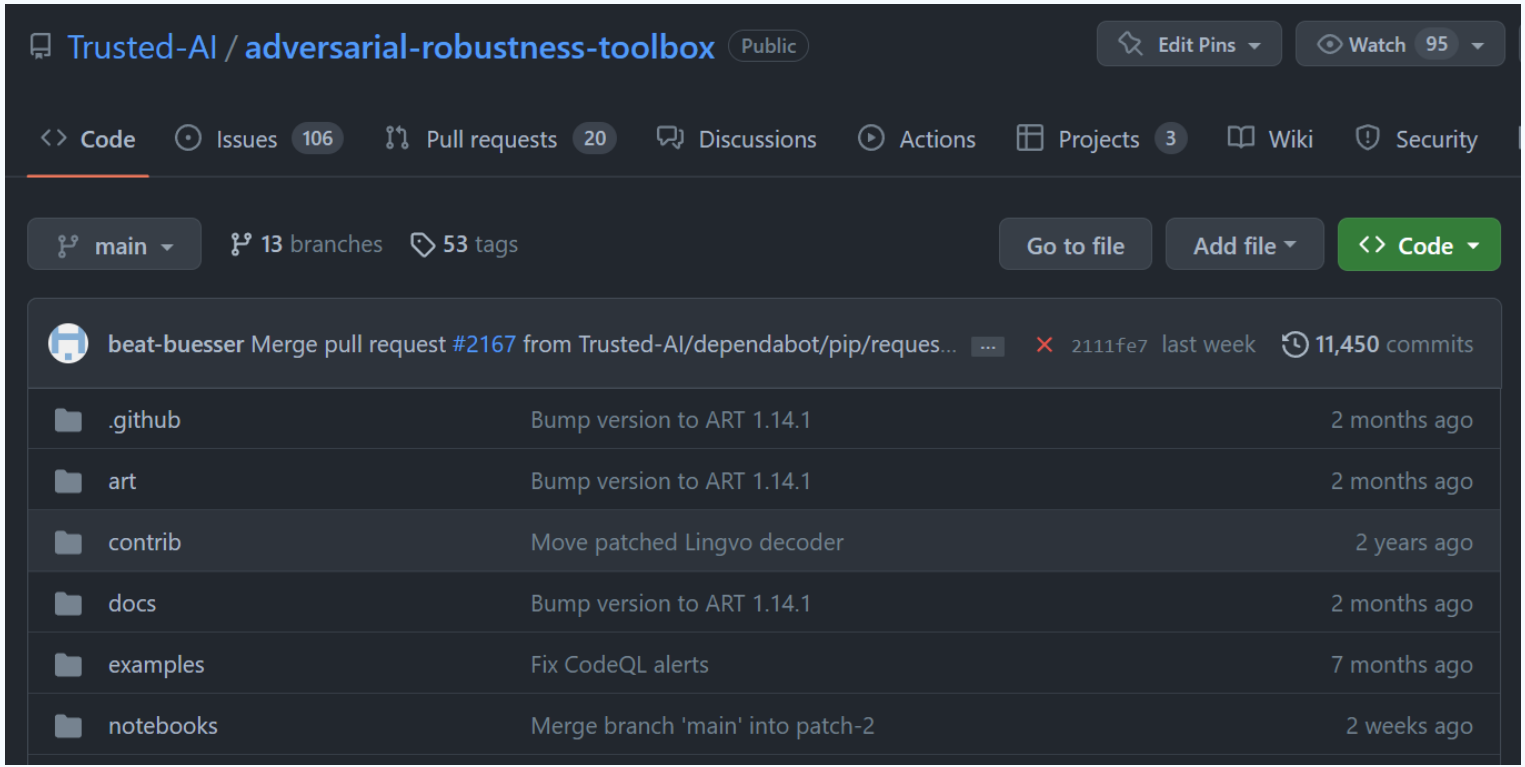
408530028 資管四 楊宗軒

Table of contents

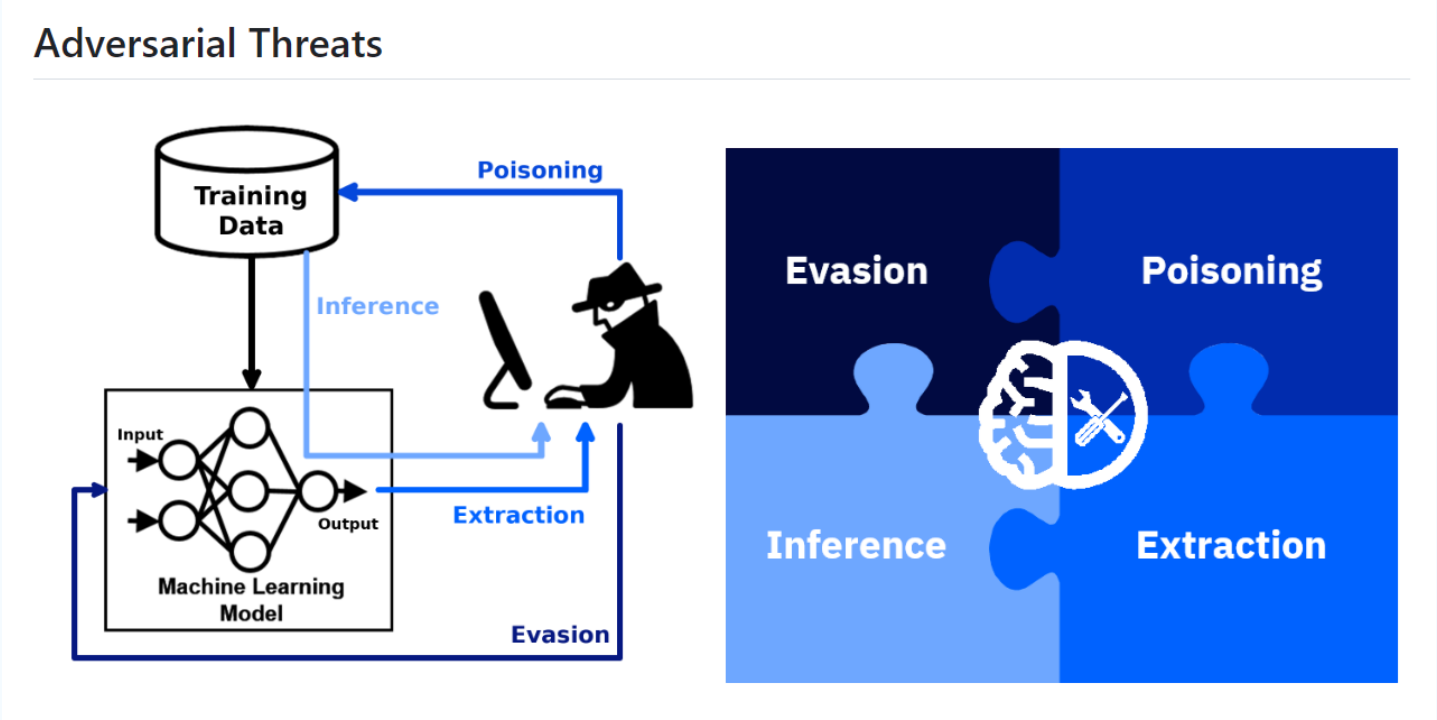
01

Introduction & Project Goal

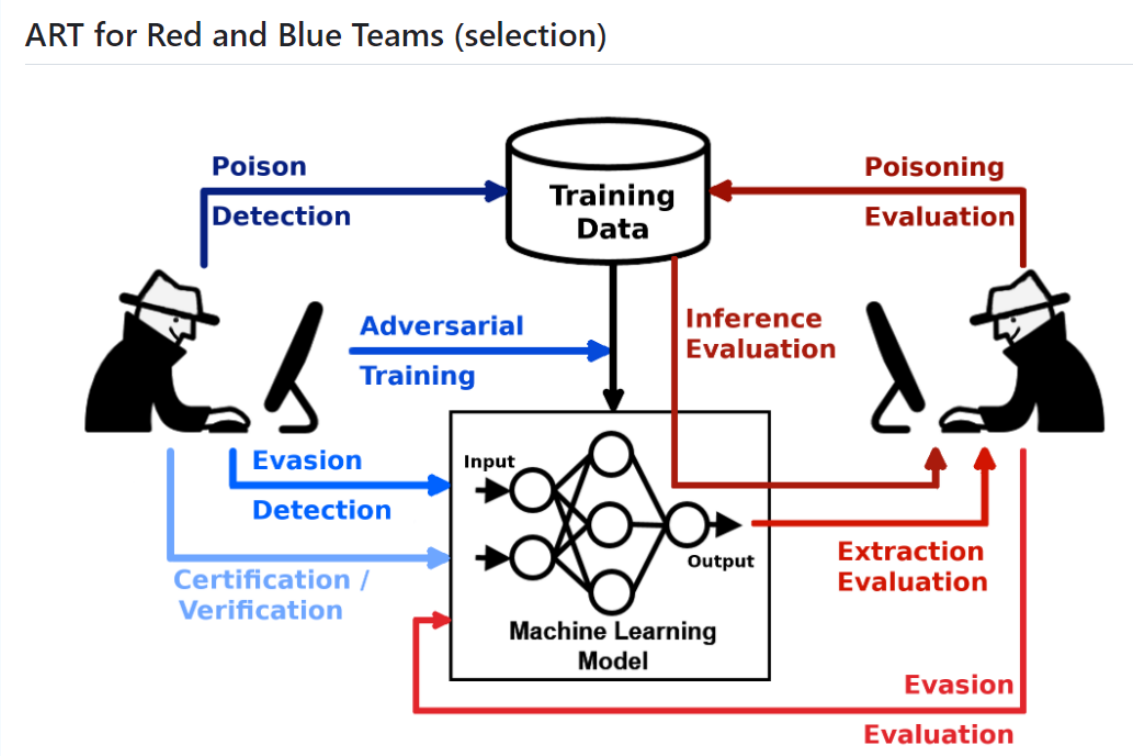
Adversarial-robustness-toolbox(1/3)



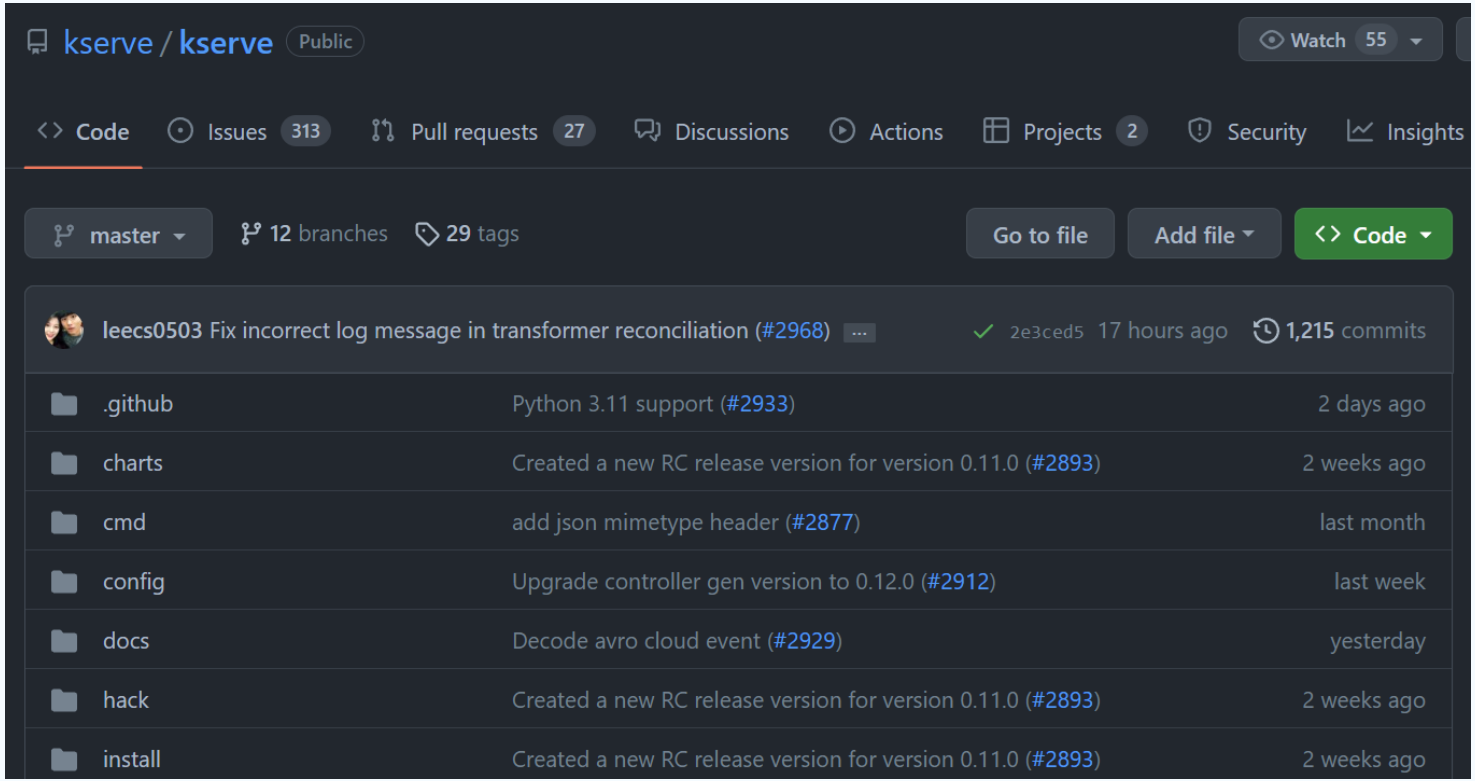
Adversarial-robustness-toolbox(2/3)



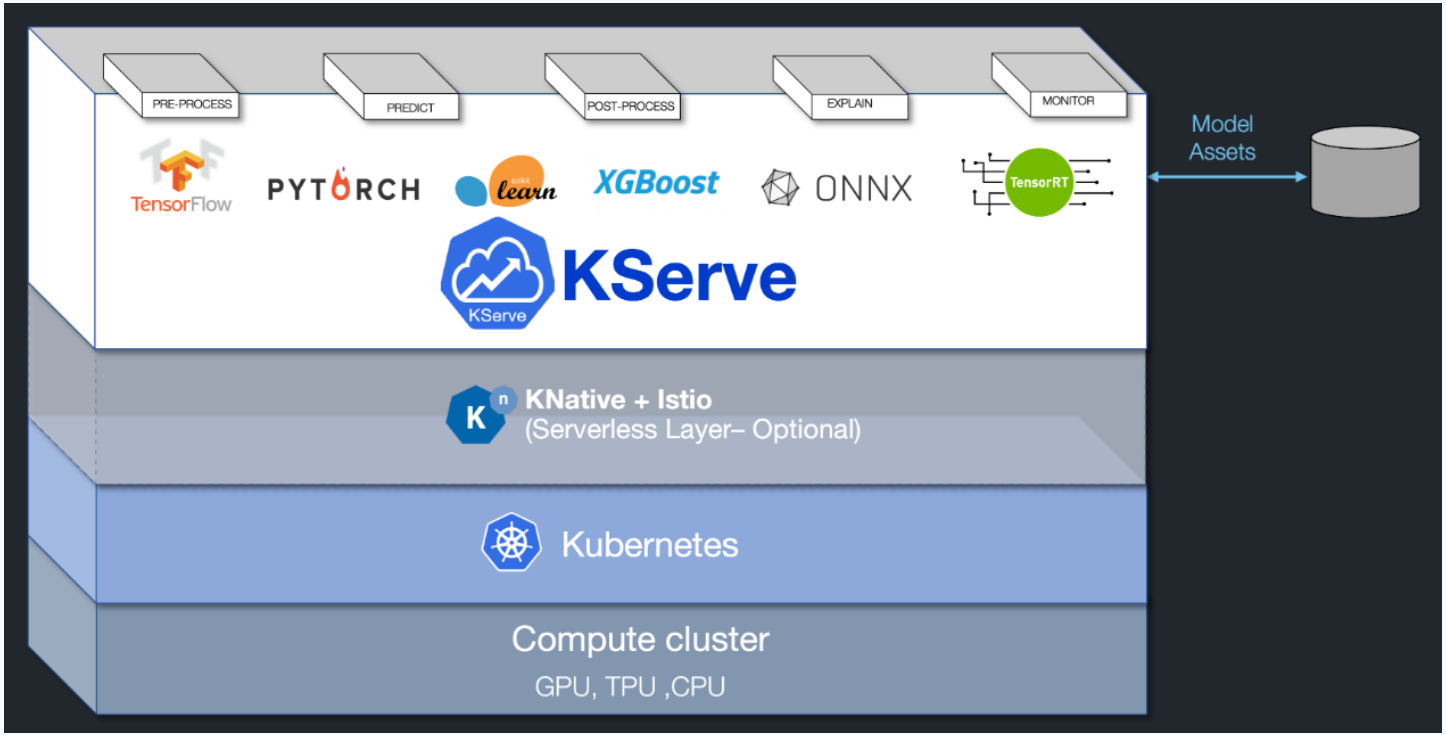
Adversarial-robustness-toolbox(3/3)



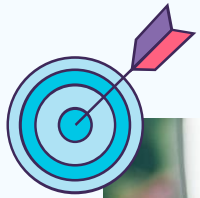
Kserve(1/2)


kserve / kserve
Public
Watch 55
Code
Issues 313
Pull requests 27
Discussions
Actions
Projects 2
Security
Insights
master
12 branches
29 tags
Go to file
Add file
Code
leecs0503 Fix incorrect log message in transformer reconciliation (#2968) ...
2e3ced5 17 hours ago
1,215 commits
.github Python 3.11 support (#2933) 2 days ago
charts Created a new RC release version for version 0.11.0 (#2893) 2 weeks ago
cmd add json mimetype header (#2877) last month
config Upgrade controller gen version to 0.12.0 (#2912) last week
docs Decode avro cloud event (#2929) yesterday
hack Created a new RC release version for version 0.11.0 (#2893) 2 weeks ago
install Created a new RC release version for version 0.11.0 (#2893) 2 weeks ago

Kserve(2/2)


PRE-PROCESS
PREDICT
POST-PROCESS
EXPLAIN
MONITOR
Model Assets
TensorFlow
PYTORCH
learn
XGBoost
ONNX
TensorRT
KServe
KServe
KNative + Istio
(Serverless Layer– Optional)
Kubernetes
Compute cluster
GPU, TPU ,CPU

Project Goal



Our project aims to integrate and contribute some functionalities from the adversarial-robustness-toolbox open-source project to enhance the capabilities of the kserve open-source project.

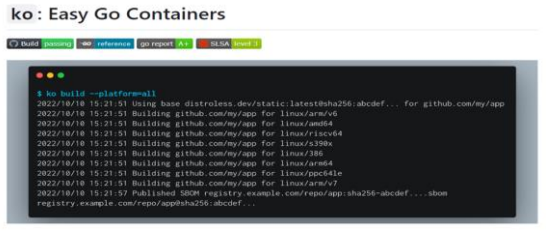
02

Kserve Environment Setup (for development)

Install Necessary Tools

https://zh.wikipedia.org/wiki/Go

https://titangene.github.io/article/git--blob-object.html



https://github.com/ko-build/ko

kubectl

yq

minikube

kustomize

minikube Setup

This command depends on current environment, here are few examples :

```
minikube start
```

```
minikube start --vm-driver=docker --force
```

Install Knative which uses Istio for traffic routing and ingress

1. Install `cosign` and `jq` .

2. Extract the images from a manifeset and verify the signatures :

```
curl -sSL
https://github.com/knative/serving/releases/download/knative-
v1.10.1/serving-core.yaml \
| grep 'gcr.io/' | awk '{print $2}' | sort | uniq \
| xargs -n 1 \
    cosign verify -o text \
    --certificate-identity=signer@knative-
releases.iam.gserviceaccount.com \
    --certificate-oidc-issuer=https://accounts.google.com
```

3. Install the latest stable Operator release :

```
kubectl apply -f
https://github.com/knative/operator/releases/download/knative-
v1.10.1/operator.yaml
```

4. Verify the installation :

```
kubectl config set-context --current --namespace=default
```

```
kubectl get deployment knative-operator
```


```
kubectl get deployment knative-operator
```


Expected output :

```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
knative-operator   1/1     1            1           1h
```

Log tracking :

```
kubectl logs -f deploy/knative-operator
```

5. Install Knative Serving

Store this text into a file :

```
apiVersion: v1
kind: Namespace
metadata:
name: knative-serving
---
apiVersion: operator.knative.dev/v1beta1
kind: KnativeServing
metadata:
name: knative-serving
namespace: knative-serving
```

Apply YAML :

```
kubectl apply -f <filename>.yaml
```

6. Install Istio :

```
curl -L https://istio.io/downloadIstio | sh -
```

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Namespace
metadata:
name: istio-system
labels:
    istio-injection: disabled
EOF
```

Install & Remove files

```
cat << EOF > ./istio-minimal-operator.yaml
apiVersion: install.istio.io/v1beta1
kind: IstioOperator
spec:
values:
    global:
    proxy:
        autoInject: disabled
    useMCP: false

meshConfig:
    accessLogFile: /dev/stdout

components:
    ingressGateways:
    - name: istio-ingressgateway
        enabled: true
        k8s:
        podAnnotations:
            cluster-autoscaler.kubernetes.io/safe-to-evict: "true"
    pilot:
    enabled: true
    k8s:
        resources:
        requests:
            cpu: 200m
            memory: 200Mi
        podAnnotations:
        cluster-autoscaler.kubernetes.io/safe-to-evict: "true"
        env:
        - name: PILOT_ENABLE_CONFIG_DISTRIBUTION_TRACKING
        value: "false"
EOF
```

```
istio-1.17.2/bin/istioctl manifest apply -f istio-minimal-operator.yaml -y
```

```
rm -rf istio-1.17.2
rm istio-minimal-operator.yaml
```

7. Install CRD, Core, istio controller :

```
kubectl apply -f https://github.com/knative/serving/releases/download/knative-v1.10.1/serving-crds.yaml
```

```
kubectl apply -f https://github.com/knative/serving/releases/download/knative-v1.10.1/serving-core.yaml
```

```
kubectl apply -f https://github.com/knative/net-istio/releases/download/knative-v1.10.0/release.yaml
```

8. Wait intil all STATUS are Running :

```
kubectl get pods -n knative-serving
```

Install Cert Manager

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.11.0/cert-manager.yaml
```

Wait intil all STATUS are Running :

```
kubectl get pods -n cert-manager
```

Adding text to ~/.bashrc

```
export KO_DOCKER_REPO='docker.io/<username>'
```

Fork kserve repository and clone it

```
git clone git@github.com:${YOUR_GITHUB_USERNAME}/kserve.git
```

```
cd kserve
```

```
git remote add upstream git@github.com:kserve/kserve.git
```

```
git remote set-url --push upstream no_push
```

Deploy kserve

```
make deploy-dev
```

Check STATUS :

```
kubectl get pods -n kserve
```

Testing (Create Inference Service)

```
kubectl create namespace kserve-test
kubectl apply -n kserve-test -f - <<EOF
apiVersion: "serving.kserve.io/v1beta1"
kind: "InferenceService"
metadata:
  name: "sklearn-iris"
spec:
  predictor:
    model:
      modelFormat:
        name: sklearn
      storageUri: "gs://kfserving-examples/models/sklearn/1.0/model"
EOF
```

```
kubectl get pods -n kserve-test
```

```
kubectl get inferenceservices -n kserve-test
```

```
minikube tunnel
```

On another terminal

```
INFERENCE_SERVICE_POD_NAME=$(kubectl get pods -n kserve-test -o jsonpath='{.items[0].metadata.name}')
```

```
kubectl port-forward -n kserve-test ${INFERENCE_SERVICE_POD_NAME} 8080:8080
```

On another terminal

```
cat <<EOF > "./iris-input.json"
{
  "instances": [
    [6.8,  2.8,  4.8,  1.4],
    [6.0,  3.4,  4.5,  1.6]
  ]
}
EOF
```

```
INGRESS_HOST=localhost
INGRESS_PORT=8080
SERVICE_HOSTNAME=$(kubectl get inferenceservice sklearn-iris -n kserve-test -o jsonpath='{.status.url}' | cut -d "/" -f
3)
curl -v -H "Host: ${SERVICE_HOSTNAME}" -H "Content-Type: application/json"
"http://${INGRESS_HOST}:${INGRESS_PORT}/v1/models/sklearn-iris:predict" -d @./iris-input.json
```

Expected output :

```
{"predictions":[1,1]}
```

01 Testing (Unit / Integration)

See Reference :
https://kserve.github.io/website/master/developer/developer/#running-unitintegration-tests

02 Testing (e2e)

See Reference :
https://kserve.github.io/website/master/developer/developer/#run-e2e-tests-locally

03

Project Model

Model

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv_1 = nn.Conv2d(in_channels=1, out_channels=4, kernel_size=5, stride=1)
        self.conv_2 = nn.Conv2d(in_channels=4, out_channels=10, kernel_size=5, stride=1)
        self.fc_1 = nn.Linear(in_features=4 * 4 * 10, out_features=100)
        self.fc_2 = nn.Linear(in_features=100, out_features=10)

    def forward(self, x):
        x = F.relu(self.conv_1(x))
        x = F.max_pool2d(x, 2, 2)
        x = F.relu(self.conv_2(x))
        x = F.max_pool2d(x, 2, 2)
        x = x.view(-1, 4 * 4 * 10)
        x = F.relu(self.fc_1(x))
        x = self.fc_2(x)
        return x
```

Source

https://github.com/Trusted-AI/adversarial-robustness-toolbox

Shadow Attack Introduction

Shadow Attack is a hybrid model that allows various kinds of attacks to be compounded together, resulting in perturbations of large radii. It can be seen as the generalization of the well-known PGD attack.

Code – Perturbation Initialization

```
perturbation = (
    np.random.uniform(
        low=self.estimator.clip_values[0], high=self.estimator.clip_values[1], size=x.shape
    ).astype(ART_NUMPY_DTYPE)
    - (self.estimator.clip_values[1] - self.estimator.clip_values[0]) / 2
)
```

Code – Training, Updating Perturbation Overview

```
for _ in trange(self.nb_steps, desc="Shadow attack", disable=not self.verbose):
    gradients_ce = np.mean(
        self.estimator.loss_gradient(x=x_batch + perturbation, y=y_batch, sampling=False)
        * (1 - 2 * int(self.targeted)),
        axis=0,
        keepdims=True,
    )
    gradients = gradients_ce - self._get_regularisation_loss_gradients(perturbation)
    perturbation += self.learning_rate * gradients
```

Code – _get_regularisation_loss_gradients : 3 channel loss

```
if perturbation_t.shape[1] == 1:
    loss_s = 0.0
elif perturbation_t.shape[1] == 3:
    loss_s = tf.norm(
        (perturbation_t[:, 0, :, :] - perturbation_t[:, 1, :, :]) ** 2
        + (perturbation_t[:, 1, :, :] - perturbation_t[:, 2, :, :]) ** 2
        + (perturbation_t[:, 0, :, :] - perturbation_t[:, 2, :, :]) ** 2,
        ord=2,
        axis=(1, 2),
    )
else:
    raise ValueError("Value for number of channels in `perturbation_t.shape` not recognized.")
```

Code – _get_regularisation_loss_gradients : Loss

```
loss = torch.mean(self.lambda_tv * loss_tv + self.lambda_s * loss_s + self.lambda_c * loss_c)
```

Code – Back to training loop : total loss

```
for _ in trange(self.nb_steps, desc="Shadow attack", disable=not self.verbose):
    gradients_ce = np.mean(
        self.estimator.loss_gradient(x=x_batch + perturbation, y=y_batch, sampling=False)
        * (1 - 2 * int(self.targeted)),
        axis=0,
        keepdims=True,
    )
    gradients = gradients_ce - self._get_regularisation_loss_gradients(perturbation)
    perturbation += self.learning_rate * gradients
```

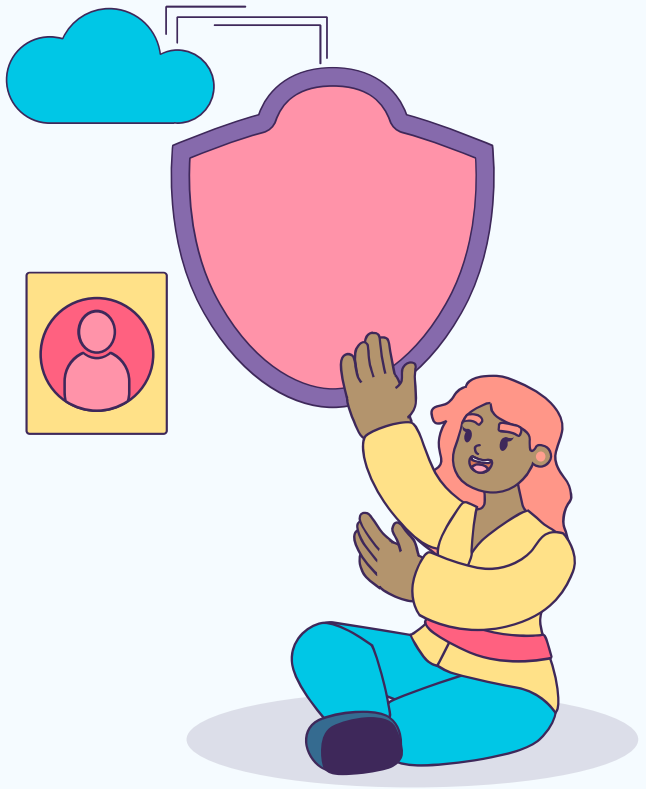
Code – Return Adversarial Example : x_adv

```
x_p = x + perturbation
x_adv = np.clip(x_p, a_min=self.estimator.clip_values[0], a_max=self.estimator.clip_values[1]).astype(
    ART_NUMPY_DTYPE
)

return x_adv
```

04

Our Work & DEMO



Expectation

```yaml
apiVersion: "serving.kserve.io/v1beta1"
kind: "InferenceService"
metadata:
  name: "artserver"
spec:
  predictor:
    model:
      modelFormat:
        name: sklearn
      storageUri: gs://kfserving-examples/models/sklearn/mnist/art
  explainer:
    art:
      type: SquareAttack # Add one more option ShadowAttack
      config:
        nb_classes: "10"
```

Process


Add Code
Build Image
Apply YAML
Create Inference Service
Test Explain

Add Code - Necessaries Definition

```
AVAILABLE_ADVERSARY_TYPES = ["squareattack", "shadowattack"]

You, 13 小時前 | 1 author (You)
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv_1 = nn.Conv2d(in_channels=1, out_channels=4, kernel_size=5, stride=1)
        self.conv_2 = nn.Conv2d(in_channels=4, out_channels=10, kernel_size=5, stride=1)
        self.fc_1 = nn.Linear(in_features=4 * 4 * 10, out_features=100)
        self.fc_2 = nn.Linear(in_features=100, out_features=10)

    def forward(self, x):
        x = F.relu(self.conv_1(x))
        x = F.max_pool2d(x, 2, 2)
        x = F.relu(self.conv_2(x))
        x = F.max_pool2d(x, 2, 2)
        x = x.view(-1, 4 * 4 * 10)
        x = F.relu(self.fc_1(x))
        x = self.fc_2(x)
        return x
```

Add Code - Initialization

```
def __init__(self, name: str, predictor_host: str, adversary_type: str,
             nb_classes: str, max_iter: str):
    super().__init__(name)
    self.name = name
    self.predictor_host = predictor_host
    if str.lower(adversary_type) not in AVAILABLE_ADVERSARY_TYPES:
        raise Exception("Invalid adversary type: %s" % adversary_type)
    self.adversary_type = adversary_type
    self.nb_classes = int(nb_classes)
    self.max_iter = int(max_iter)
    self.ready = False
    self.count = 0
```

```
    (x_train, y_train), _, _, _ = load_mnist()
    x_train = np.transpose(x_train, (0, 3, 1, 2)).astype(np.float32)

    model = Net()

    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.01)

    classifier = PyTorchClassifier(
        model=model,
        clip_values=(-1, 1),
        loss=criterion,
        optimizer=optimizer,
        input_shape=(1, 28, 28),
        nb_classes=self.nb_classes,
    )

    classifier.fit(x_train, y_train, batch_size=64, nb_epochs=3)

    self.classifier = classifier
```

Add Code - Explain Function

```
adversary_type = str.lower(self.adversary_type)
if adversary_type == "squareattack":
    classifier = BlackBoxClassifierNeuralNetwork(self._predict, inputs.shape, self.nb_classes,
                                                 channels_first=False, clip_values=(-np.inf, np.inf))
    preds = np.argmax(classifier.predict(inputs, batch_size=1))
    attack = SquareAttack(estimator=classifier, max_iter=self.max_iter)
    x_adv = attack.generate(x=inputs, y=label)

    adv_preds = np.argmax(classifier.predict(x_adv))
    l2_error = np.linalg.norm(np.reshape(x_adv[0] - inputs, [-1]))

    return {"explanations": {"adversarial_example": x_adv.tolist(), "L2 error": l2_error.tolist(),
                             "adversarial_prediction": adv_preds.tolist(), "prediction": preds.tolist()}}
elif adversary_type == "shadowattack":
    inputs = np.transpose(inputs, (0, 3, 1, 2)).astype(np.float32)

    preds = np.argmax(self.classifier.predict(inputs, batch_size=1))
    attack = ShadowAttack(estimator=self.classifier)
    inputs = np.expand_dims(inputs[0], axis=0)
    x_adv = attack.generate(x=inputs)

    adv_preds = np.argmax(self.classifier.predict(x_adv))
    l2_error = np.linalg.norm(np.reshape(x_adv[0] - inputs, [-1]))

    return {"explanations": {"adversarial_example": np.transpose(x_adv, (0, 2, 3, 1)).tolist(), "L2 error": l2_error.tolist(),
                             "adversarial_prediction": adv_preds.tolist(), "prediction": preds.tolist()}}
```

Build Image

```
~/De/kserve/python master !2 ?2 docker build -t lo0k0502/artserver:latest -f artexplainer.Dockerfile .
docker push lo0k0502/artserver:latest
docker rmi $(docker images 'lo0k0502/artserver:latest' -a -q)
```

Apply YAML & Create Inference Service

```yaml
apiVersion: "serving.kserve.io/v1beta1"
kind: "InferenceService"
metadata:
  name: "artserver"
spec:
  predictor:
    model:
      modelFormat:
        name: sklearn
      storageUri: gs://kfserving-examples/models/sklearn/mnist/art
  explainer:
    containers:
    - image: lo0k0502/artserver:latest
      name: artserver
      args:
        - --model_name=artserver
        - --adversary_type=ShadowAttack
        - --predictor_host=artserver.kserve-test.svc.cluster.local
      resources:
        limits:
          memory: "2Gi"
          cpu: "1"
        requests:
          memory: "2Gi"
          cpu: "1"
```

Test Explain

01 Open tunnel

$ minikube tunnel

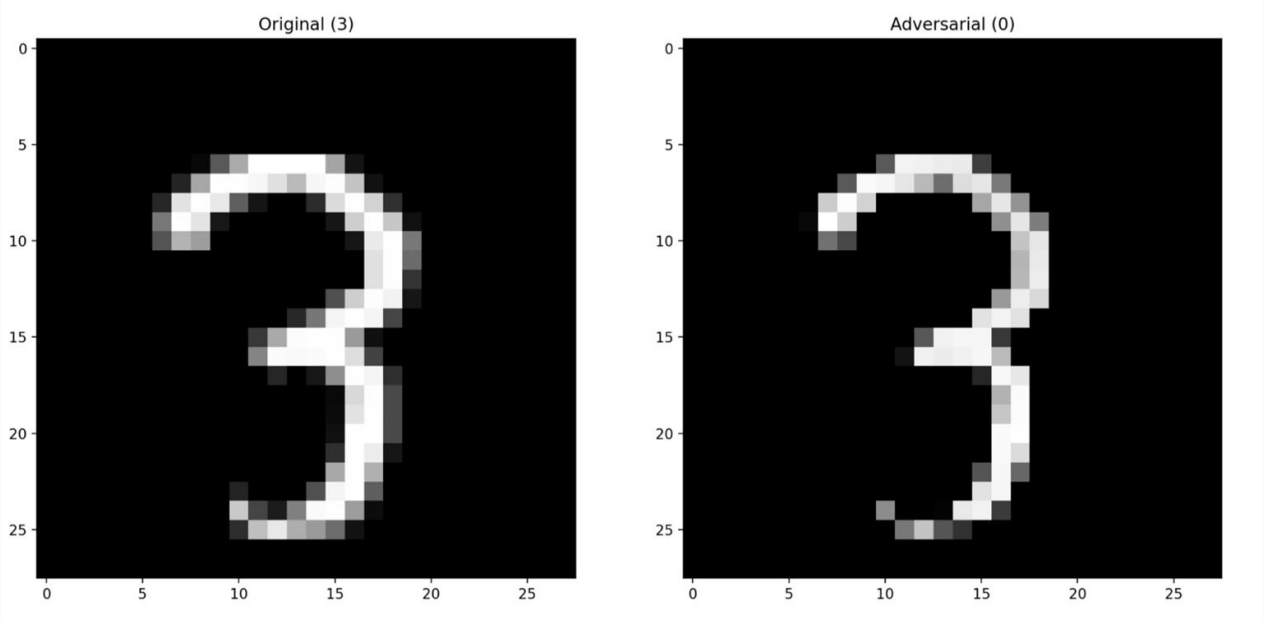
02 Port Forward

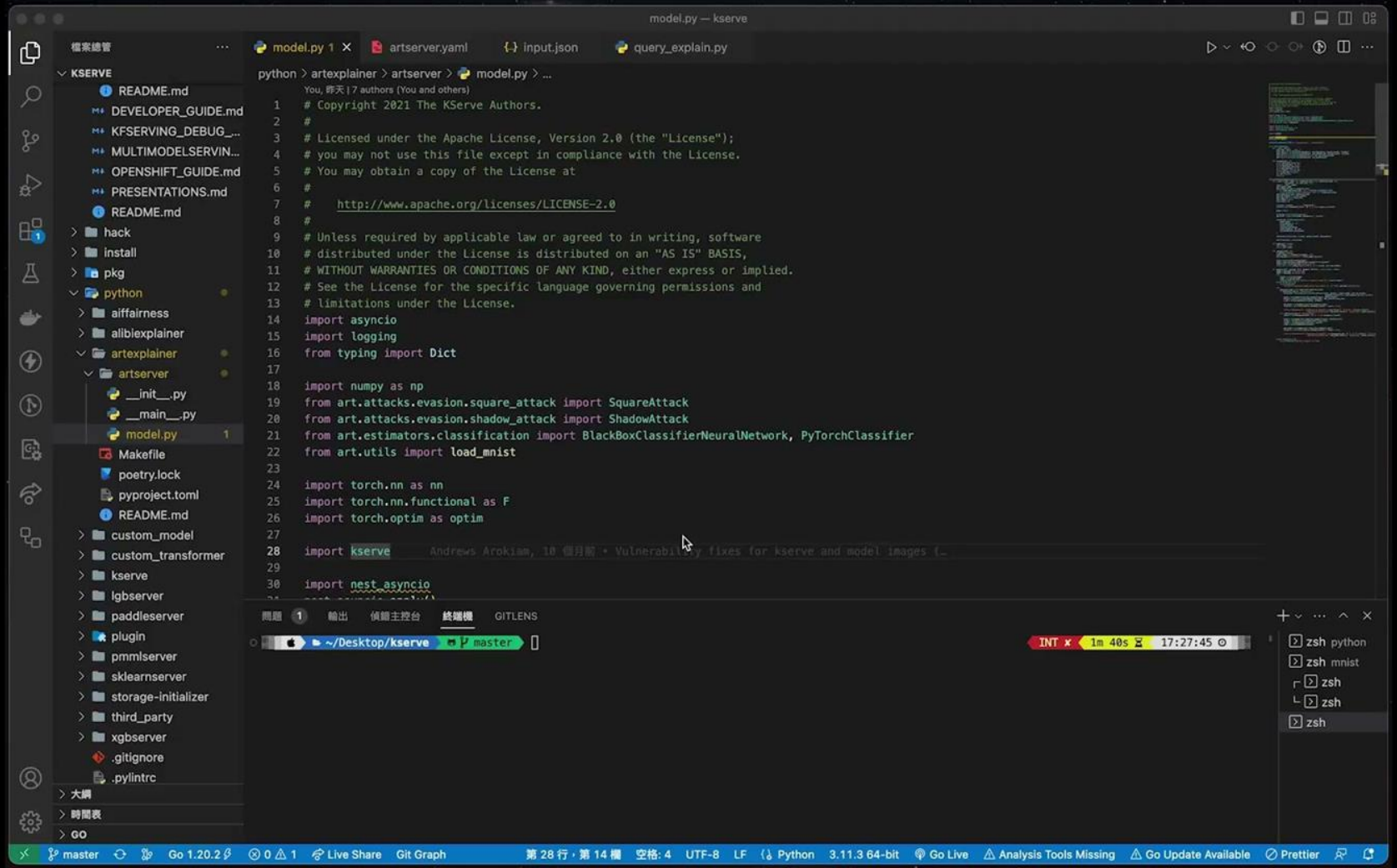
$ kubectl port-forward kubectl port-forward -n kserve-test artserver-explainer-00001-deployment-bd94d4b49-rkxhw 8080:8080

03 Test

$ python3 query_explain.py http://localhost:8080/v1/models/artserver:explain artserver.kserve-test.svc.cluster.local ./input.json

Result




QuickTime Player 檔案 編輯 顯示方式 視窗 輔助說明
SSD 50% CPU 38% RAM 63%
6月7日 週三 17:34
model.py — kserve
檔案總管
KSERVE
README.md
DEVELOPER_GUIDE.md
KFSERVING_DEBUG_...
MULTIMODELSERVIN...
OPENSHIFT_GUIDE.md
PRESENTATIONS.md
README.md
hack
install
pkg
python
aiffairness
alibiexplainer
artexplainer
artserver
__init__.py
__main__.py
model.py 1
Makefile
poetry.lock
pyproject.toml
README.md
custom_model
custom_transformer
kserve
lgbserver
paddleserver
plugin
pmmlserver
sklearnserver
storage-initializer
third_party
xgbserver
.gitignore
.pylintrc
大綱
時間表
GO
model.py 1
artserver.yaml
input.json
query_explain.py
python > artexplainer > artserver > model.py > ...
You, 昨天 | 7 authors (You and others)
Copyright 2021 The KServe Authors.
#
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
#
http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
import asyncio
import logging
from typing import Dict
import numpy as np
from art.attacks.evasion.square_attack import SquareAttack
from art.attacks.evasion.shadow_attack import ShadowAttack
from art.estimators.classification import BlackBoxClassifierNeuralNetwork, PyTorchClassifier
from art.utils import load_mnist
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import kserve
Andrews Arokiam, 10 個月前 • Vulnerability fixes for kserve and model images (...
import nest_asyncio
問題 1 輸出 偵錯主控台 終端機 GITLENS
~/Desktop/kserve master
INT 1m 40s 17:27:45
zsh python
zsh mnist
zsh
zsh
zsh
master Go 1.20.2 0 1 Live Share Git Graph
第 28 行，第 14 欄 空格: 4 UTF-8 LF Python 3.11.3 64-bit Go Live Analysis Tools Missing Go Update Available Prettier

Conclusion

We have made substantial progress in our project, reaching 80% of our goals. Our primary achievement was the successful integration of various art attack techniques into the kserve platform. By incorporating these techniques, we have expanded the capabilities and effectiveness of kserve. To showcase the functionality of these additions, we have developed a functional prototype that effectively demonstrates and simulates the art attack methods.

05

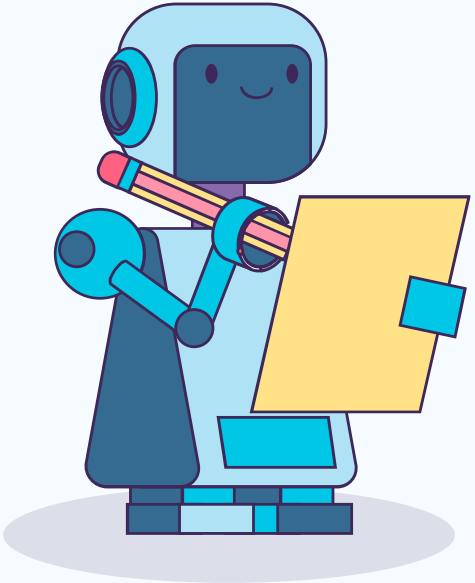
What we have learned

Work Allocation :

- **408530003 資管四 范綱彥**
  - Conducted research on kserve installation and explored deployment strategies using Docker files and YAML configurations. Additionally, worked on the preliminary integration of shadow attack techniques into kserve's art-explainer module.
- **408530004 資管四 潘甫翰 (Leader)**
  - Conducted research on shadow attacks and Kubernetes, focusing on understanding their concepts and methodologies. Additionally, performed unit testing and integrated them into end-to-end (e2e) testing processes.
- **408530007 資管四 謝瀞瑩**
  - Conducted research on kserve installation and gained insights into Kubernetes concepts. Also contributed to the design and creation of the project presentation, ensuring it effectively communicates the project's progress and findings.
- **408530028 資管四 楊宗軒**
  - Responsibilities: Conducted research on shadow attacks and focused on understanding the installation process of the kserve platform. Explored various methods and techniques related to kserve installation.

What we learned

- Kubernetes
- kserve
- kubectl
- minikube
- docker
- clustering
- shadow attack
- art evasion
- Git commiting
- how to ask good question
- try and error



Thanks!