# 4.8 AI 모델 추론 애플리케이션 개발 - MONAI Deploy

## Deploying a MONAI Model on IGX Using MONAI Deploy and Holoscan: A Step-by-Step Guide

This guide will **comprehensively** explain how to **train a MONAI model, deploy it using MONAI Deploy, and integrate it with Holoscan for real-time execution on an NVIDIA IGX system**.

---

## 📌 Overview of the Deployment Process

### 1. Train & Deploy the MONAI Model

- Train a **MONAI-based AI model**.
- Convert the model into **ONNX or TensorRT format**.
- Deploy it using **MONAI Deploy** and serve it via **Triton Inference Server**.

### 2. Set Up Holoscan on NVIDIA IGX

- Install the **Holoscan SDK** on IGX.
- Configure **sensor data acquisition (e.g., ultrasound, CT, endoscopy)**.

### 3. Integrate MONAI Deploy with Holoscan

- Load the MONAI model from **Triton Inference Server** inside a Holoscan application.
- Use **TensorRT** for optimized inference.
- Implement **sensor → AI inference → visualization** pipeline.

### 4. Run Holoscan Application on IGX

- Compile and deploy the Holoscan application on IGX.
- Perform real-time inference and visualization.

## 🔷 Step 1: Train & Deploy the MONAI Model with MONAI Deploy

Before integrating with Holoscan, we need to **train a MONAI-based model and deploy it using Triton Inference Server**.

### 📌 1.1 Train and Save a MONAI Model

We will use **MONAI's UNet-based segmentation model** as an example.

```
import torch
from monai.networks.nets import UNet

# Define MONAI UNet model
model = UNet(
    spatial_dims=2,
    in_channels=1,
    out_channels=2,
    channels=(16, 32, 64, 128, 256),
    strides=(2, 2, 2, 2),
    num_res_units=2
)

# Save trained model
torch.save(model.state_dict(), "monai_model.pth")
print("Model saved as monai_model.pth")
```

---

### 📌 1.2 Convert the Model to ONNX for Deployment

Since **Triton Inference Server** supports **ONNX models**, we convert the MONAI model into **ONNX format**.

```
import torch
import onnx

# Load trained MONAI model
model = UNet(
    spatial_dims=2,
```

```
    in_channels=1,
    out_channels=2,
    channels=(16, 32, 64, 128, 256),
    strides=(2, 2, 2, 2),
    num_res_units=2
)
model.load_state_dict(torch.load("monai_model.pth"))
model.eval()

# Convert to ONNX format
dummy_input = torch.randn(1, 1, 128, 128)
onnx_path = "monai_model.onnx"
torch.onnx.export(model, dummy_input, onnx_path, input_name
s=['input'], output_names=['output'])
print(f"Model exported to {onnx_path}")
```

---

## 📌 1.3 Deploy the Model Using MONAI Deploy

We will package the MONAI Deploy app and integrate it with **Triton Inference Server**.

### Create a MONAI Deploy App

Inside a new directory `monai_deploy_app/`, create `app.py`:

```
from monai.deploy.core import Application, resource
from operators.inference_operator import InferenceOperator
from operators.transform_operator import TransformOperator

class MonaiDeployApp(Application):
    @resource(cpu=4, gpu=1, memory="4Gi")
    def compose(self):
        transform_op = TransformOperator()
        inference_op = InferenceOperator()
        self.add_flow(transform_op, inference_op)

if __name__ == "__main__":
```

```
    app = MonaiDeployApp()
    app.run()
```

---

### Run MONAI Deploy and Triton

```
monai-deploy package monai_deploy_app/ --output monai_triton_package
monai-deploy tritonserver --model-repository=monai_triton_package
```

🚀 **Now your MONAI model is deployed as a Triton Inference Service on NVIDIA IGX!**

---

## 🔷 Step 2: Set Up Holoscan on NVIDIA IGX

To run **Holoscan** on **IGX**, we need to install the **Holoscan SDK** and configure real-time medical data acquisition.

### 📌 2.1 Install Holoscan SDK on IGX

Run the following commands on the **IGX system**:

```
# Install Holoscan SDK
sudo apt update
sudo apt install holoscan-sdk
```

Alternatively, use the **Holoscan Docker container**:

```
docker pull nvcr.io/nvidia/holoscan:v0.5
```

---

### 📌 2.2 Configure Real-Time Sensor Data Input

Holoscan can handle **real-time sensor input** (e.g., **ultrasound, endoscopy, CT scanner feeds**).

For an **ultrasound input**, configure:

```
sensor:
  type: "Ultrasound"
```

```
  device: "/dev/video0"
  resolution: [1024, 768]
  frame_rate: 30
```

Now, your **IGX system** can **receive real-time data** for inference.

---

## 🔷 Step 3: Integrate MONAI Deploy with Holoscan

Now, we will create a **Holoscan application** that:

1. **Acquires sensor data (e.g., ultrasound).**
2. **Processes it using a MONAI model running on Triton.**
3. **Uses TensorRT for optimized inference.**
4. **Visualizes the results.**

---

### 📌 3.1 Create a Holoscan App

Modify `src/main.cpp` to **connect sensor → MONAI inference → visualization**:

```
#include <holoscan/holoscan.hpp>
#include "monai_inference.hpp"

class HoloscanApp : public holoscan::Application {
public:
    void compose() override {
        auto source = make_operator<SensorSourceOp>("Ultras
oundSensor");
        auto monai_inference = make_operator<MonaiInference
Op>("MONAIInference");
        auto visualization = make_operator<VisualizationOp>
("Visualization");

        add_flow(source, monai_inference);
        add_flow(monai_inference, visualization);
    }
};

int main(int argc, char** argv) {
```

```
    HoloscanApp app;
    app.run();
    return 0;
}
```

---

### 📌 3.2 Implement MONAI Inference Operator

Modify `src/monai_inference.cpp` to **send data to Triton Inference Server**:

```
#include <holoscan/holoscan.hpp>
#include <tritonclient/http/client.h>

class MonaiInferenceOp : public holoscan::Operator {
public:
    void compute(holoscan::Context &context) override {
        auto input = context.input("input_frame")->receive
();
        std::vector<int64_t> shape = {1, 1, 128, 128};
        auto input_tensor = holoscan::Tensor::create(input.
data(), shape);

        tritonclient::InferenceServerClient client("localho
st:8000");
        auto output = client.infer("monai_model", {{"inpu
t", input_tensor}});

        context.output("segmentation_result")->emit(outpu
t);
    }
};
```

---

## 🔷 Step 4: Run Holoscan Application on IGX

### 📌 4.1 Build the Application

```
mkdir build && cd build
cmake ..
```

```
make
```

---

### 📌 4.2 Run Holoscan Pipeline

```
./holoscan_app --config configs/holoscan.yaml
```

**🚀 Now, Holoscan is running real-time inference on IGX using the MONAI model!**

---

## 🎯 Conclusion

**✅ Trained & deployed a MONAI model**

**✅ Ran it as a Triton Inference Service**

**✅ Integrated it with Holoscan**

**✅ Ran real-time AI inference on IGX**

Now, your **AI model can process live medical data in real-time on IGX**, enabling **edge-based, AI-driven healthcare solutions**.

---

## 🔗 Additional Resources

- 📖 MONAI Deploy Docs
- 📘 Holoscan SDK
- 🔗 Triton Inference Server

Let me know if you need further clarifications! 🚀🔥