Simulink Kullanarak NVIDIA Jetson AGX Orin Üzerinde Gerçek Zamanlı YOLOv9 Nesne Tespiti Dağıtımı

Bu örnek, bir IP kamera akışı kullanarak gerçek zamanlı nesne tespiti yapmak amacıyla bir Simulink® modelinin NVIDIA® Jetson AGX Orin™ kartına nasıl dağıtılacağını (yükleneceğini) gösterir. Bu örnek, gelişmiş YOLOv9 derin öğrenme ağını kullanarak nesneleri gerçek zamanlı olarak tespit eder ve takip eder. Örnekteki Simulink modeli; bir IP kameradan canlı video akışını yakalamak, TensorRT™ ile optimize edilmiş CUDA® kodunu kullanarak bu akışı işlemek ve tespit edilen sınırlayıcı kutular (bounding boxes) ile tahmin sonuçlarını Jetson platformuna bağlı bir monitörde görüntülemek için "NVIDIA Jetson ve NVIDIA DRIVE™ Platformları için MATLAB® Coder™ Destek Paketi"nde yer alan ağ akışı girişi (network stream input) ve görüntüleme (display) bloklarını kullanır.

Bu örnekte spesifik olarak bir NVIDIA Jetson AGX Orin Geliştirici Kiti (Developer Kit) kullanılmaktadır.

Önkoşullar

Hedef Kart Gereksinimleri

- NVIDIA Jetson AGX Orin gömülü platformu.
- Hedef kart ile ana bilgisayarı (host PC) bağlamak için Ethernet kablosu (veya IP kamera için ağ bağlantısı).
- Ağ üzerinden erişilebilen bir IP Kamera (RTSP/HTTP akışı).
- Hedef cihazın ekran çıkışına (display port) bağlı bir monitör.
- IP akışının kodunu çözmek (decoding) için hedef üzerinde GStreamer kütüphaneleri.
- Hedef üzerinde NVIDIA CUDA® araç kiti (toolkit) ve sürücüsü.
- Yüksek performanslı çıkarım (inference) işlemleri için hedef üzerinde NVIDIA cuDNN ve TensorRT™ kütüphaneleri.
- Derleyiciler ve kütüphaneler için hedef üzerinde tanımlanmış ortam değişkenleri (environment variables). Daha fazla bilgi için bkz. *NVIDIA Kartı İçin Kod Üretme Önkoşulları (Prerequisites for Generating Code for NVIDIA Board)*.

Geliştirme Sunucusu Gereksinimleri

- Derleyiciler ve kütüphaneler için ortam değişkenleri. Daha fazla bilgi için, Üçüncü Taraf Donanım (GPU Coder) ve Önkoşul Ürünlerinin Kurulumu (GPU Coder) bölümlerine bakın.

NVIDIA Jetson'a Bağlanma

Destek paketi, Jetson platformlarında üretilen CUDA kodunu derlerken ve çalıştırırken komutları yürütmek için TCP/IP üzerinden bir SSH bağlantısı kullanır. Hedef platformu, ana bilgisayarla aynı ağa bağlayın.

NVIDIA donanımıyla iletişim kurmak için jetson fonksiyonunu kullanarak canlı bir donanım bağlantı nesnesi oluşturun.

```
hwobj = jetson("jetson-deviceaddress","username","password");
```

NVIDIA donanımıyla iletişim kurmak için jetson fonksiyonunu kullanarak canlı bir donanım bağlantı nesnesi oluşturun. Hedef karta ilk kez bağlanırken; hedef kartın IP adresini, kullanıcı adını ve şifresini girin.

```
hwobj = jetson;
```

```
### Checking for CUDA availability on the target...
### Checking for 'nvcc' in the target system path...
### Checking for cuDNN library availability on the target...
### Checking for TensorRT library availability on the target...
### Checking for prerequisite libraries is complete.
### Gathering hardware details...
### Checking for third-party library availability on the target...
### Launching the server
### Server launch is successful
### Gathering hardware details is complete.
Board name              : NVIDIA Jetson AGX Orin Developer Kit
CUDA Version            : 11.4
cuDNN Version           : 8.6
TensorRT Version        : 8.5
GStreamer Version       : 1.16.3
V4L2 Version            : 1.18.0-2build1
SDL Version             : 1.2
OpenCV Version          : 4.5.4
Available Webcams       : USB 2.0 Camera: USB Camera
Available GPUs          : Orin
Available Digital Pins  : 7  11  12  13  15  16  18  19  21  22  23  24  26  29  31  32  33  35  36  37  38  40
```

Bir donanım nesnesi oluşturduğunuzda; yazılım donanım ve yazılım kontrollerini gerçekleştirir, hedef karta MATLAB IO sunucusunu yükler ve bilgi toplar. Uzaktan derleme işlemi gerçekleştireceğiniz donanım kartını seçmek için `setupCodegenContext()` metodunu kullanın.

```
setupCodegenContext(hwobj);
```

Hedef Kartta GPU Ortamını Doğrulama

Bu örneği çalıştırmak için gerekli derleyicilerin ve kütüphanelerin doğru şekilde ayarlandığını doğrulamak için `coder.checkGpuInstall` fonksiyonunu kullanın. Bu örnek, optimize edilmiş çıkarım (inference) için TensorRT kullandığından, derin öğrenme hedef kütüphanesini `tensorrt` olarak ayarlayın.

```
envCfg = coder.gpuEnvConfig('jetson');
envCfg.DeepLibTarget = 'tensorrt';
envCfg.DeepCodegen = 1;
envCfg.HardwareObject = hwobj;
coder.checkGpuInstall(envCfg);
```

```
Compatible GPU          : PASSED
CUDA Environment        : PASSED
    Runtime   : PASSED
    cuFFT     : PASSED
    cuSOLVER  : PASSED
    cuBLAS    : PASSED
cuDNN Environment       : PASSED (Warning: Deep learning code generation has been tested with cuDNN v8.9. The provi
TensorRT Environment    : PASSED (Warning: Deep learning code generation has been tested with TensorRT v8.6. The pr
Deep Learning (TensorRT) Code Generation: PASSED
TensorRT FP32 Compute Capability Check: PASSED
```

Gerçek Zamanlı YOLOv9 Tespiti İçin Simulink Modeli

Nesne tespiti için Simulink modeli; sınırlayıcı kutuları (bounding boxes) ve etiketleri tahmin etmek için bir derin öğrenme bloğu, canlı IP kamera akışını yakalamak için bir video kaynağı bloğu ve sonuçları görselleştirmek için bir ekran (display) bloğu içerir. Görüntü verilerini biçimlendirmek ve çıktıya açıklamalar eklemek (annotate) için ilave işleme adımları uygulanmıştır.

```
open_system('yolov9_orin_ipcam_detect');
```

Model beş ana aşamaya ayrılmıştır:

1. **NVIDIA Video Source (NVIDIA Video Kaynağı):** Standart USB web kameralarından farklı olarak bu blok, ağ üzerinden bir IP kameradan RTSP/HTTP video akışı alacak şekilde yapılandırılmıştır. Her bir zaman adımı (time-step) için ham R, G ve B renk kanallarını ayrıştırır.
2. **RGBToImg:** Bu alt sistem, video kaynağından gelen ayrı R, G ve B bileşen çıktılarını, nesne tespiti ağının gerektirdiği düzlemsel (planar) RGB görüntü formatına dönüştürür.
3. **YOLO-V9 (Detection Model - Tespit Modeli):** Bu blok, görüntü karesini (frame) alır ve YOLOv9 derin öğrenme ağını kullanarak tahmin (prediction) işlemini gerçekleştirir. AGX Orin üzerinde gerçek zamanlı performans elde etmek için bu blok, NVIDIA TensorRT kullanılarak optimize edilmiştir. Çıktı; sınırlayıcı kutu koordinatlarını (bboxes), güven skorlarını (scores) ve sınıf etiketlerini (labelIds) içerir.
4. **Detection and Tracking (Tespit ve Takip):** Bu işleme bloğu, saniyedeki kare sayısını (FPS) hesaplar, nesneleri kareler boyunca takip eder ve girdi görüntüsü üzerine tahmin etiketlerini, bunlara karşılık gelen skorları ve sınırlayıcı kutuları ekleyerek görüntüyü işaretler (annotate).
5. **Video Viewer / Display (Video Görüntüleyici / Ekran):** Elde edilen çıktı görüntüsü NVIDIA Display bloğuna iletilir; bu blok da Jetson hedefine bağlı monitörde nihai işaretlenmiş video akışını gösteren bir SDL görüntüleme penceresi açar.

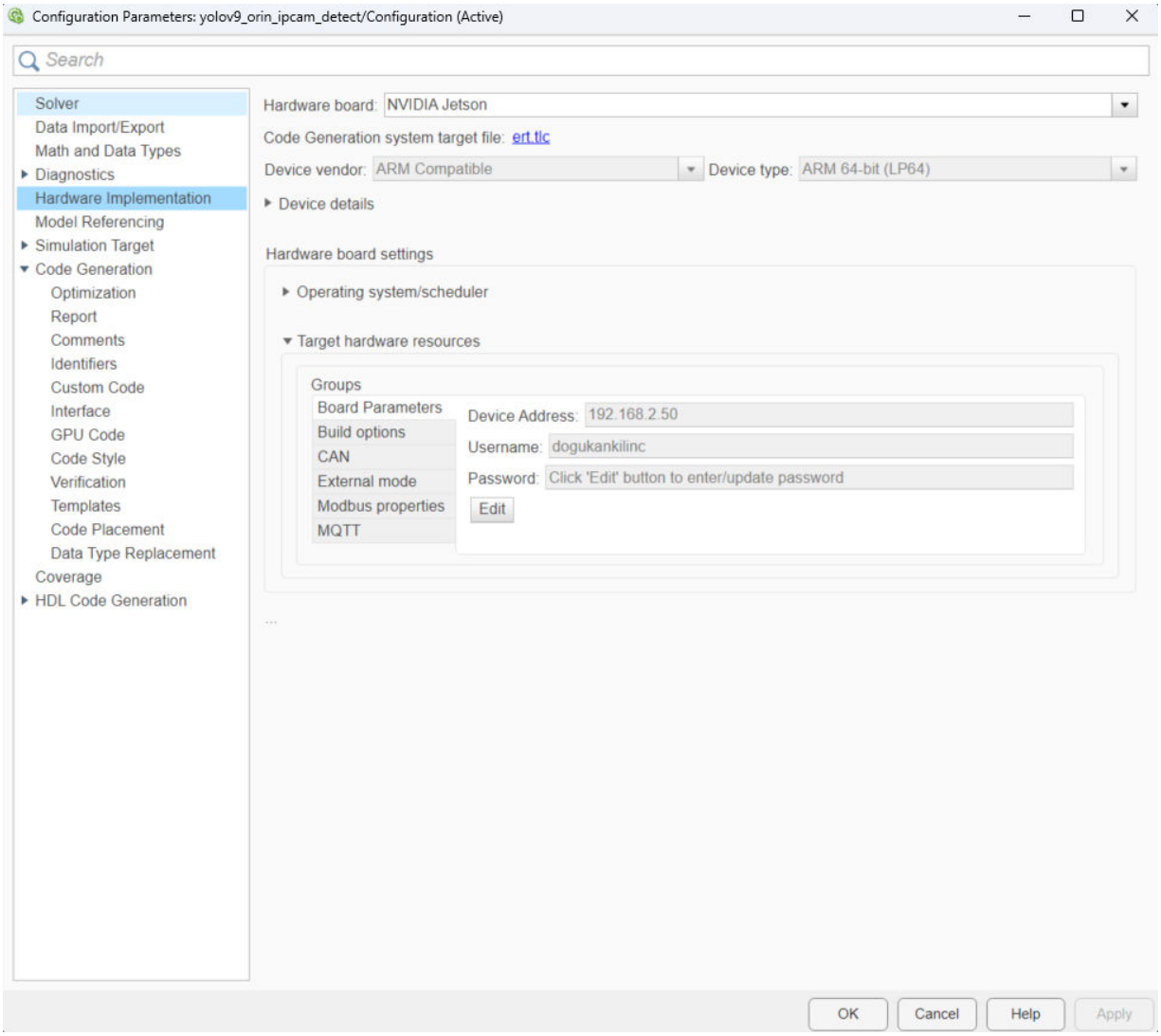
Modeli Simülasyon ve Dağıtım İçin Yapılandırma

CUDA kodu üretmek için modelin belirli kod üretimi (code generation) ayarlarıyla etkinleştirilmesi gerekir. Bu örnekteki model, NVIDIA Jetson platformu için optimize edilmiş C++ CUDA kodu üretmek üzere önceden yapılandırılmıştır.

1. Donanım Uygulama (Hardware Implementation) Ayarları

Kendi hedef kartınızın bağlantısını yapılandırmak için:

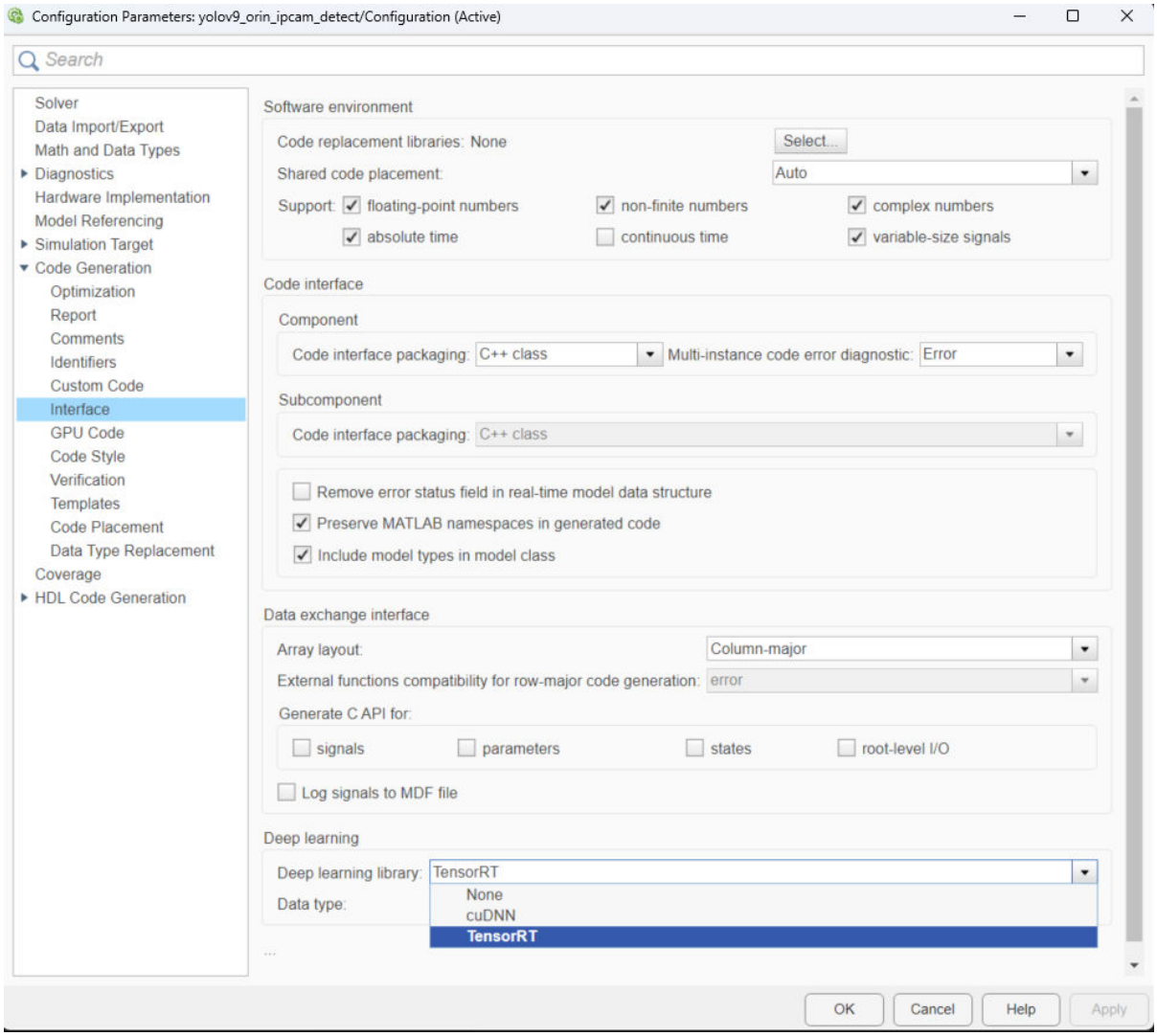
- **Configuration Parameters** (Yapılandırma Parametreleri) iletişim kutusunu açın ve **Hardware Implementation** bölmesini seçin.
- **Hardware board** seçeneğinin **NVIDIA Jetson** olarak ayarlandığından emin olun.
- **Target hardware resources** (Hedef donanım kaynakları) bölümünün altından **Board Parameters**'ı seçin.
- NVIDIA Jetson AGX Orin hedef kartınıza ait **Device Address** (Cihaz Adresi, örn. 192.168.2.50), **Username** (Kullanıcı Adı, örn. dogukankilinc) ve **Password** (Şifre) bilgilerini girin.



Derin Öğrenme Hedef Kütüphanesi Yapılandırması

Simulink allows you to generate code using different deep learning libraries depending on your performance requirements and target hardware. Simulink, performans gereksinimlerinize ve hedef donanımınıza bağlı olarak farklı derin öğrenme kütüphanelerini kullanarak kod üretmenize olanak tanır.

- **Configuration Parameters** (Yapılandırma Parametreleri) iletişim kutusunda, **Code Generation > Interface** yolunu izleyin.
- **Deep learning** bölümüne aşağı kaydırın.
- **Deep learning library** parametresi için **None**, **cuDNN** ve **TensorRT** arasından seçim yapabilirsiniz. AGX Orin üzerinde yüksek performanslı, gerçek zamanlı nesne tespitini hedeflediğimiz için **TensorRT**'yi seçin.

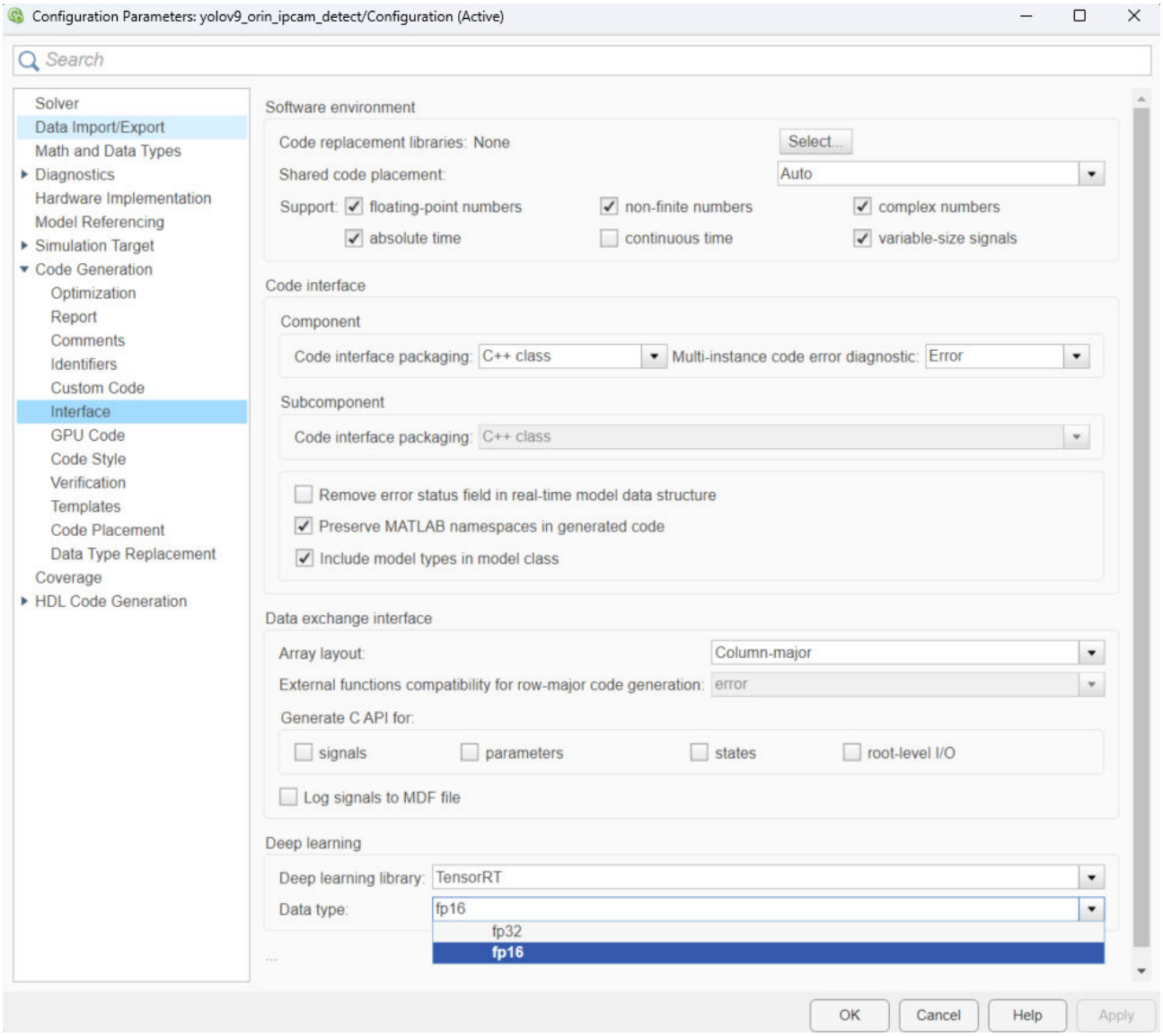


Hassasiyet ve Veri Tipi Optimizasyonu (FP16)

TensorRT kullanırken, çıkarım (inference) hızını artırmak için ağın matematiksel hassasiyetini optimize etme avantajına sahipsiniz.

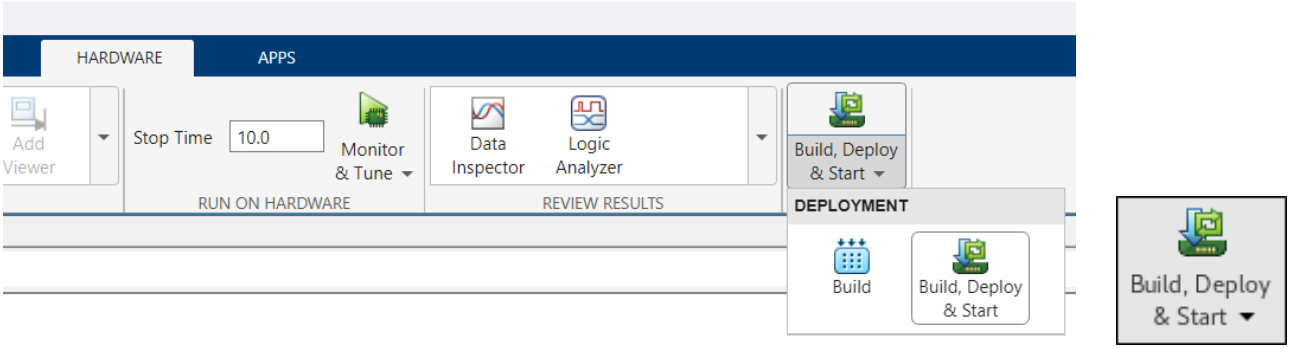
- Yine **Deep learning** bölümü altında, **Data type** (Veri tipi) açılır menüsünü bulun.
- **fp32** (32-bit floating-point / kayan nokta) veya **fp16** (16-bit half-precision floating-point / yarı hassasiyetli kayan nokta) kullanarak kod üretme seçeneğine sahipsiniz.

- Bu örnek için **fp16**'yı seçiyoruz. **fp16** hassasiyetini kullanmak, Jetson AGX Orin üzerindeki Tensor Çekirdeklerinden (Tensor Cores) tam anlamıyla faydalanmanızı sağlar ve tespit doğruluğu (detection accuracy) üzerinde minimum etkiyle YOLOv9 modeli için saniyedeki kare sayısını (FPS) önemli ölçüde artırır.
- Yapılandırmanızı kaydetmek için **Apply** (Uygula) ve **OK** (Tamam) butonlarına tıklayın.



Modeli Hedef Kart Üzerinde Üretme ve Dağıtma

1. Model için CUDA kodu üretmek, çalıştırılabilir dosyayı (executable) hedefe dağıtmak ve uygulamayı çalıştırmak için Simulink Editöründeki **Hardware** (Donanım) sekmesini açın.
2. İşlemi başlatmak için **Build, Deploy & Start** (Derle, Dağıt ve Başlat) seçeneğine tıklayın.

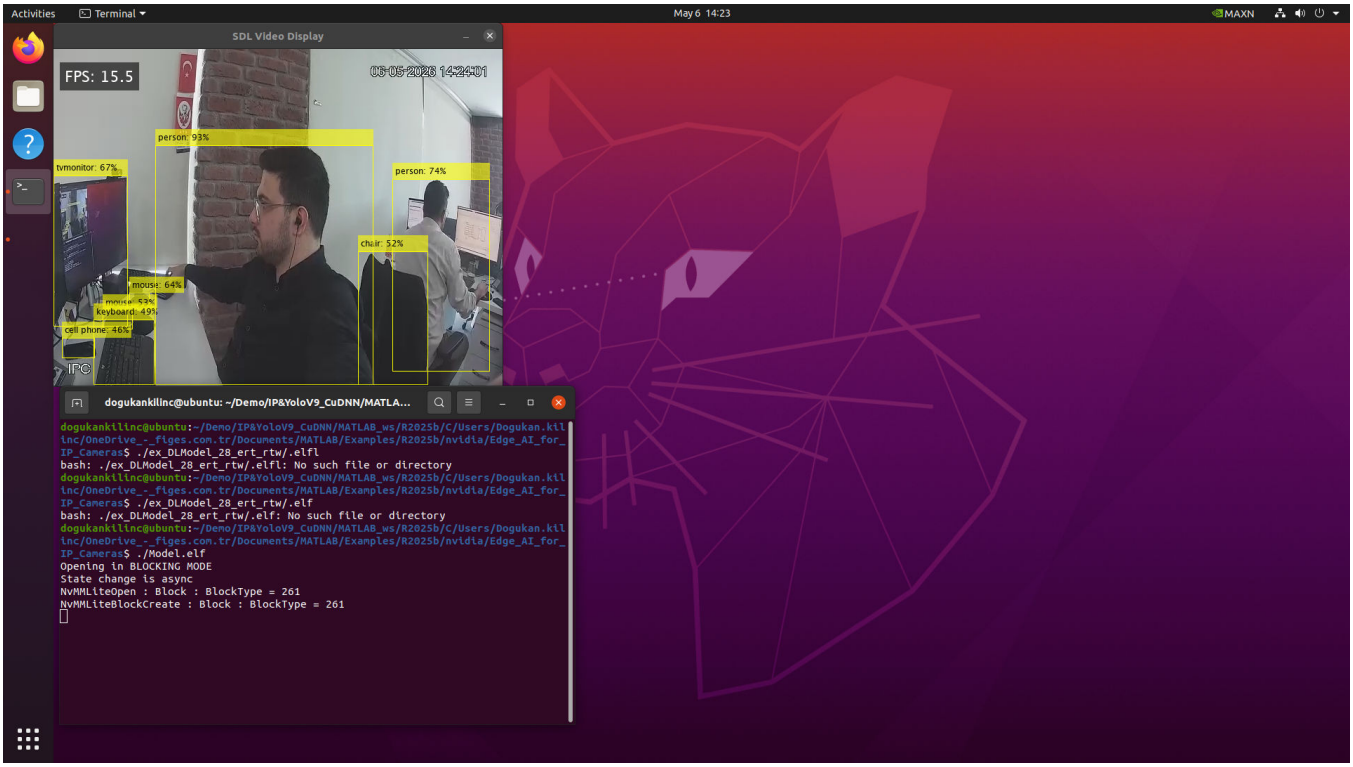


Not: *Derin öğrenme hedef kütüphanesi TensorRT olarak ayarlandığından, kod üretici ilk dağıtım (deployment) sırasında YOLOv9 ağına ve AGX Orin donanımına özgü bir TensorRT motoru oluşturacaktır. Bu motor oluşturma süreci birkaç dakika sürebilir. Sonraki derlemelerde (builds), üretilen bu motor yeniden kullanılacak ve dağıtım işlemi çok daha hızlı gerçekleşecektir.*

NVIDIA Jetson Üzerinde Tespit Sonuçlarını Görüntüleme

Uygulama hedef kart üzerinde başarıyla başladığında, Jetson AGX Orin'e bağlı monitörde bir SDL ekran penceresi açılır.

Bu pencere, IP kameradan alınan gerçek zamanlı video akışını gösterecektir. YOLOv9 ağı her bir kareyi (frame) işler ve takip alt sistemi, tespit edilen nesnelerin etrafına sınırlayıcı kutular (bounding boxes) çizer. Sınıf etiketlerini (örn. insan, sandalye, monitör), bunlara karşılık gelen güven skorlarını (confidence scores) ve sistem tarafından hesaplanan gerçek zamanlı ortalama FPS değerini video akışı üzerine yerleştirilmiş (overlay) şekilde göreceksiniz.



Uygulamayı Durdurma

Hedef kart üzerinde çalışan uygulamayı ana (host) MATLAB ortamınızdan durdurmak için donanım bağlantı nesnesinin `stopModel` metodunu kullanın.

```
stopModel(hwobj, 'yolov9_orin_ipcam_detect');
```

Modeli Kapatma

Simulink modelini çalışma alanındaki kaydedilmemiş değişiklikleri kaydetmeden kapatmak için `close_system` fonksiyonunu çalıştırın.

```
close_system('yolov9_orin_ipcam_detect', 0);
```

Doğukan M. KILINÇ tarafından geliştirilmiştir | 6 Mayıs 2026