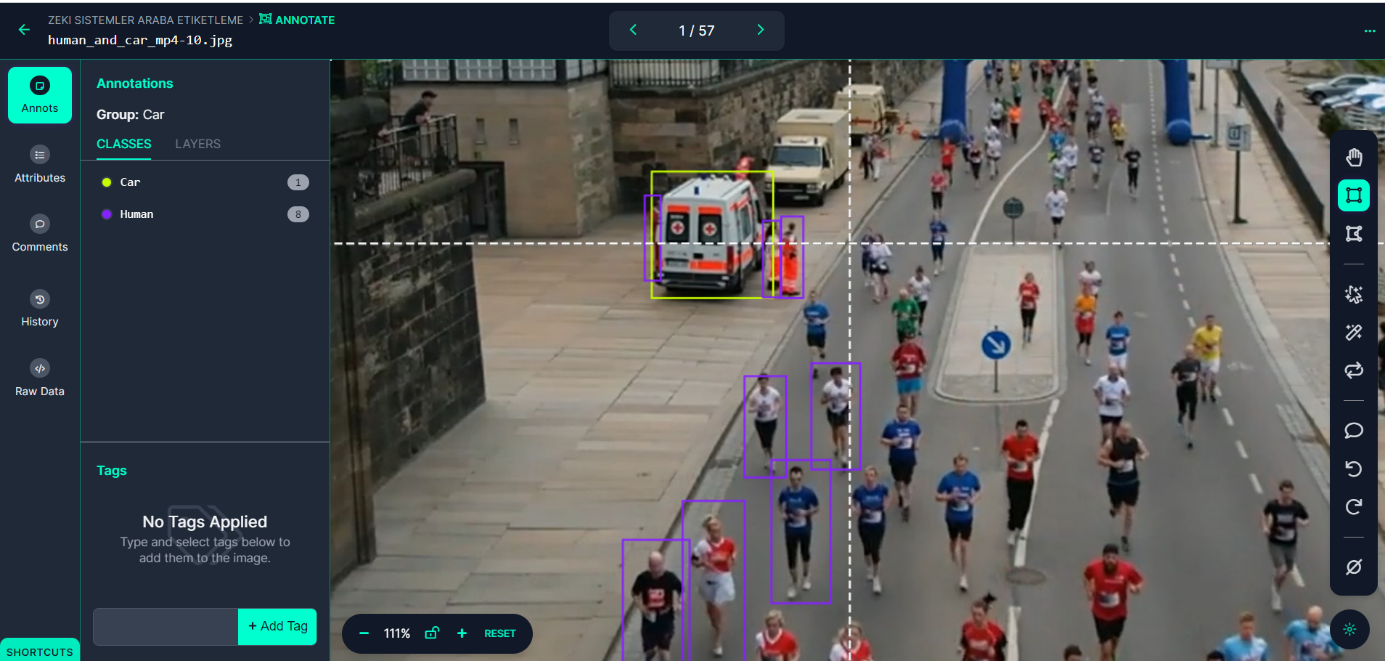
**SIFIRDAN YOLO PROJESİ NASIL OLUŞTURULUR?**

1. **VERİ SETİNİ HAZIRLAMA**

**-**Veri setini hazırlamak için roboflow websitesini kullanıyoruz. Veri setimizi roboflow üzerinden sınıflarımıza göre etiketliyoruz.



**-**Train, test ve valid oranlarımızı seçiyoruz.

metin, yazı tipi, çizgi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**-**Boyutlandırma, sınıfların idex değerleriyle oynama ve benzer diğer kısımları yapıyoruz.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

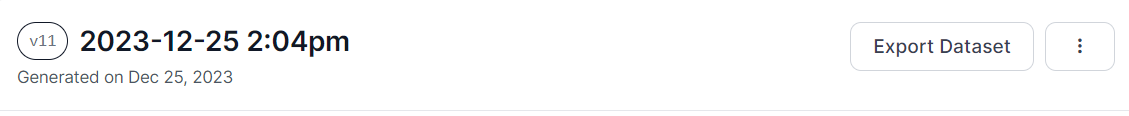
Açıklama otomatik olarak oluşturuldu

**-**Son olarak veri setimizi oluşturuyoruz.

metin, ekran görüntüsü, yazı tipi, cebir içeren bir resim

Açıklama otomatik olarak oluşturuldu

**-**Veri setimizi dışa aktarıyoruz.



**-**Veri setimizi zip formatıyla indiriyoruz.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**-**İndirilen zip dosyasını açıyoruz ve projeyi yapacağımız dosyaya aktarıyoruz.

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. **VERİ SETİMİZLE YOLO V8 ALGORİTMASINI EĞİTME**

**-**data.yaml dosyasını kontrol ediyoruz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**-**Yeni bir py dosyasınaYolo v8 kütüphanesini ekliyoruz.



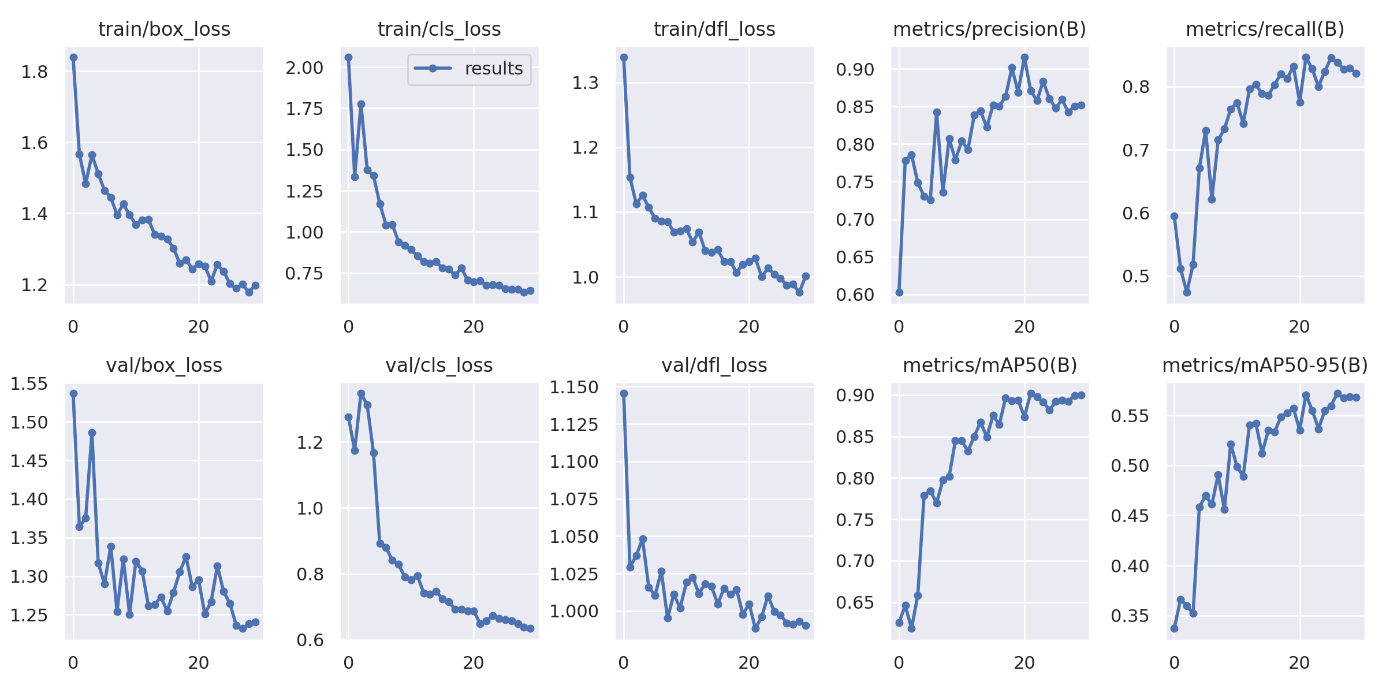
**-**Eğitim aşamasının kodlarını yazıyoruz. model.train() kısmında datamızın yolunu yazıyoruz ve epoch, batch değerlerini girdikten sonra eğitimi başlatıyoruz.

metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

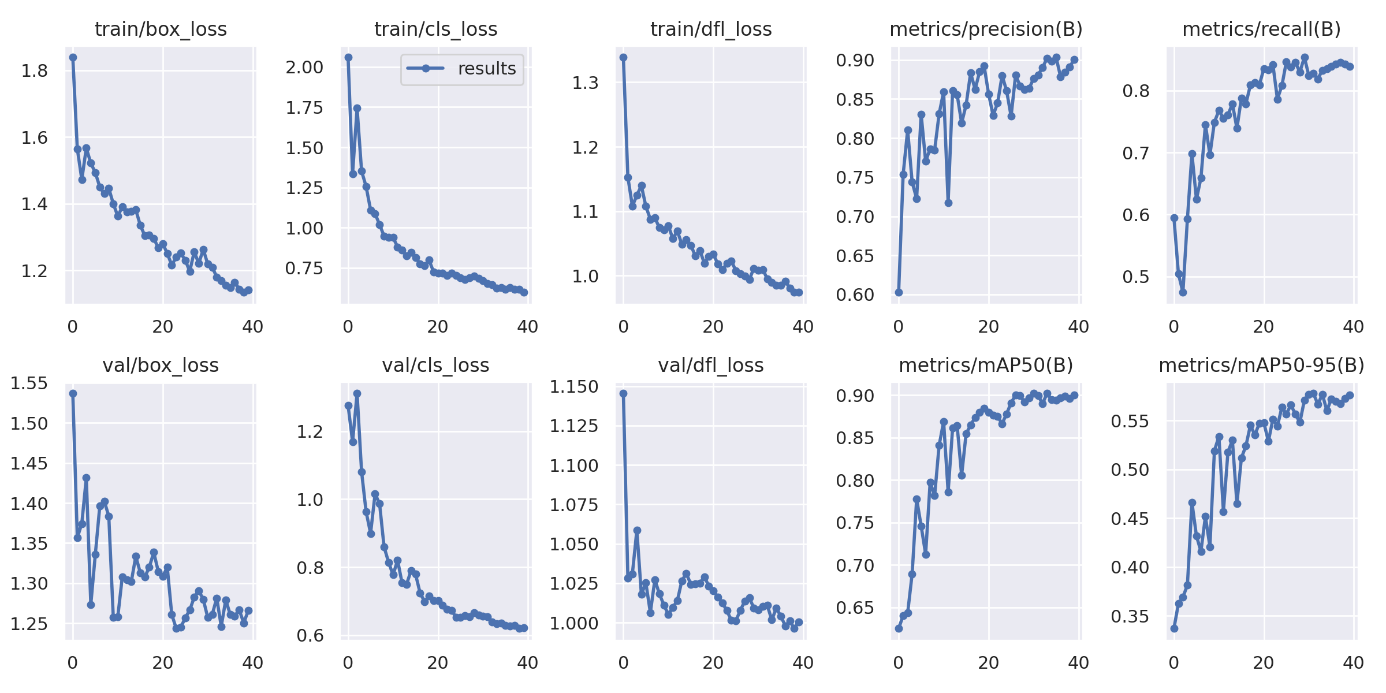
Açıklama otomatik olarak oluşturuldu

**-**Eğitilen veri setinde epoch ve batch değerlerine göre alınan sonuçlar

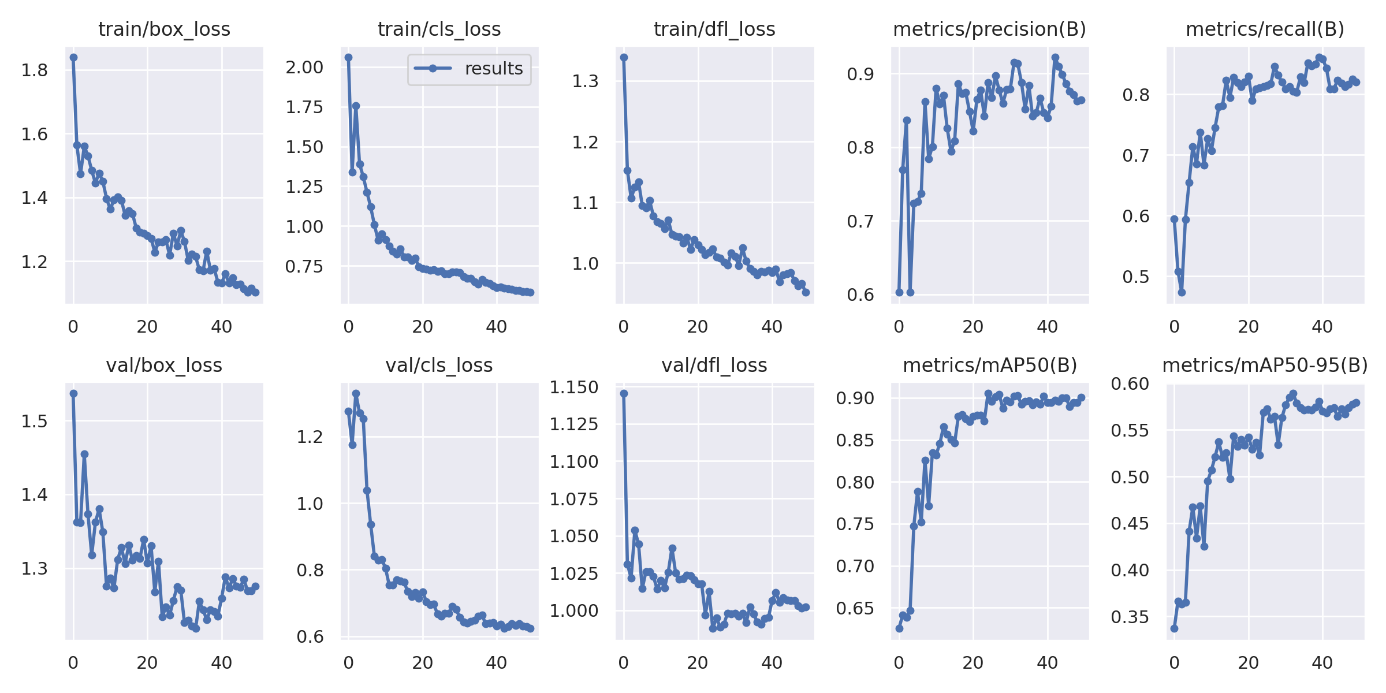
* Batch = 2, Epoch = 30



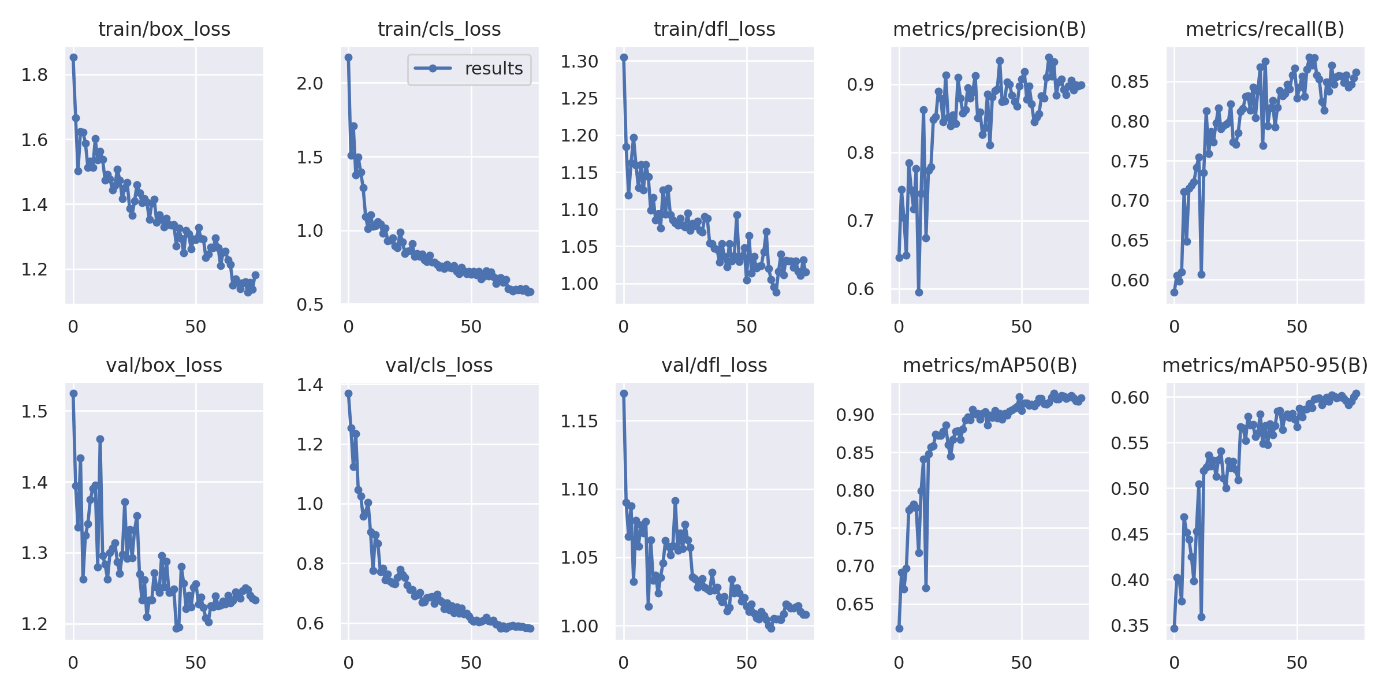
* Batch = 2, Epoch = 40



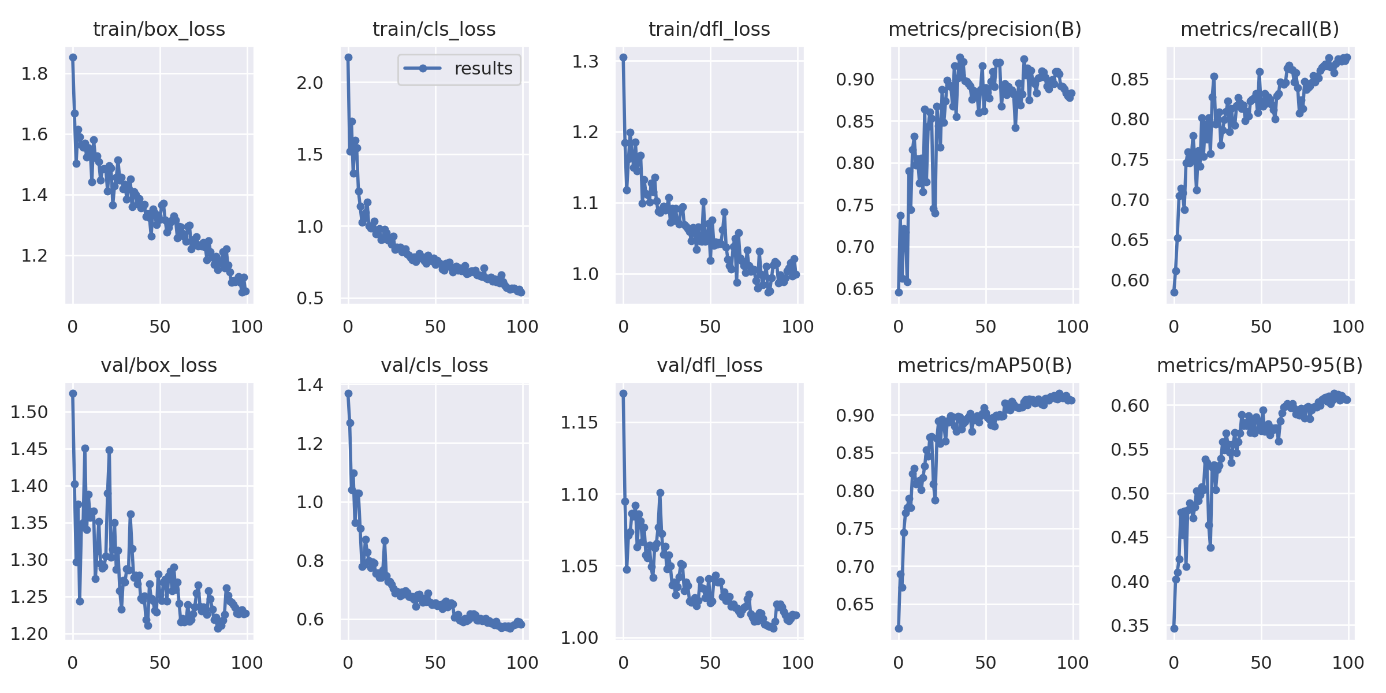
* Batch = 2, Epoch = 50



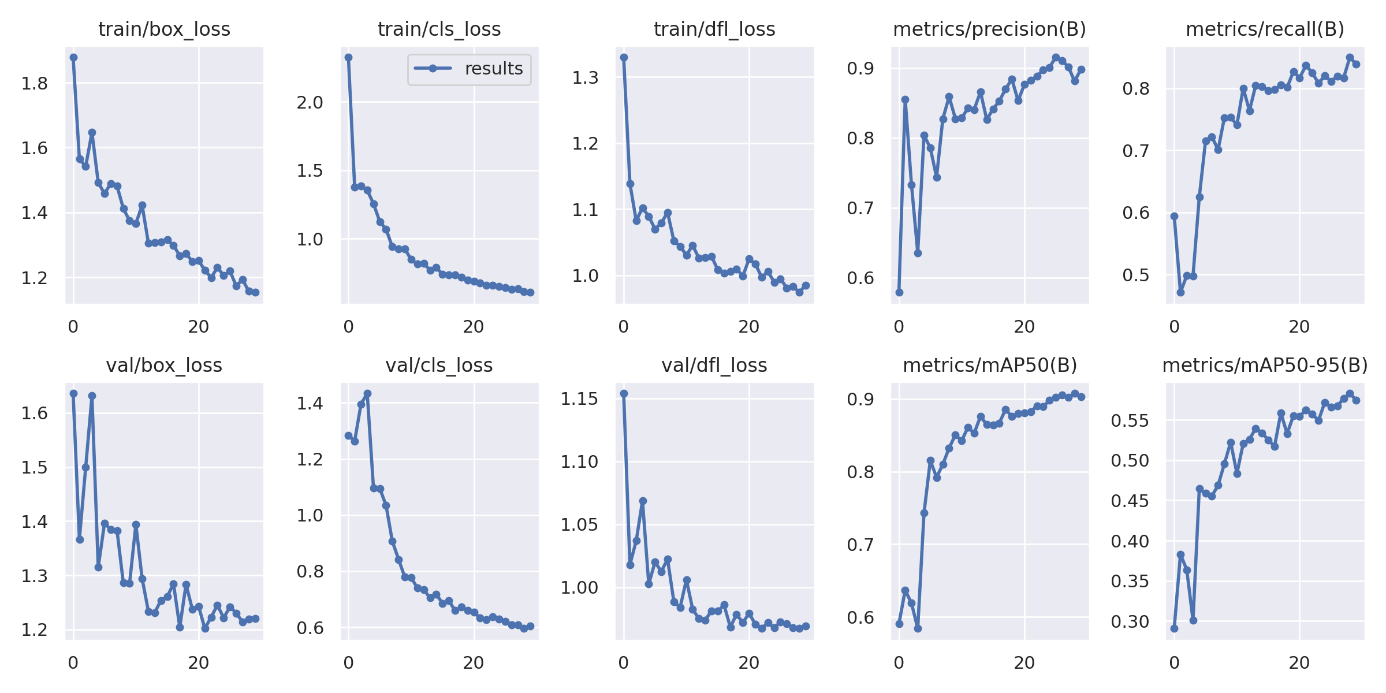
* Batch = 2, Epoch = 75



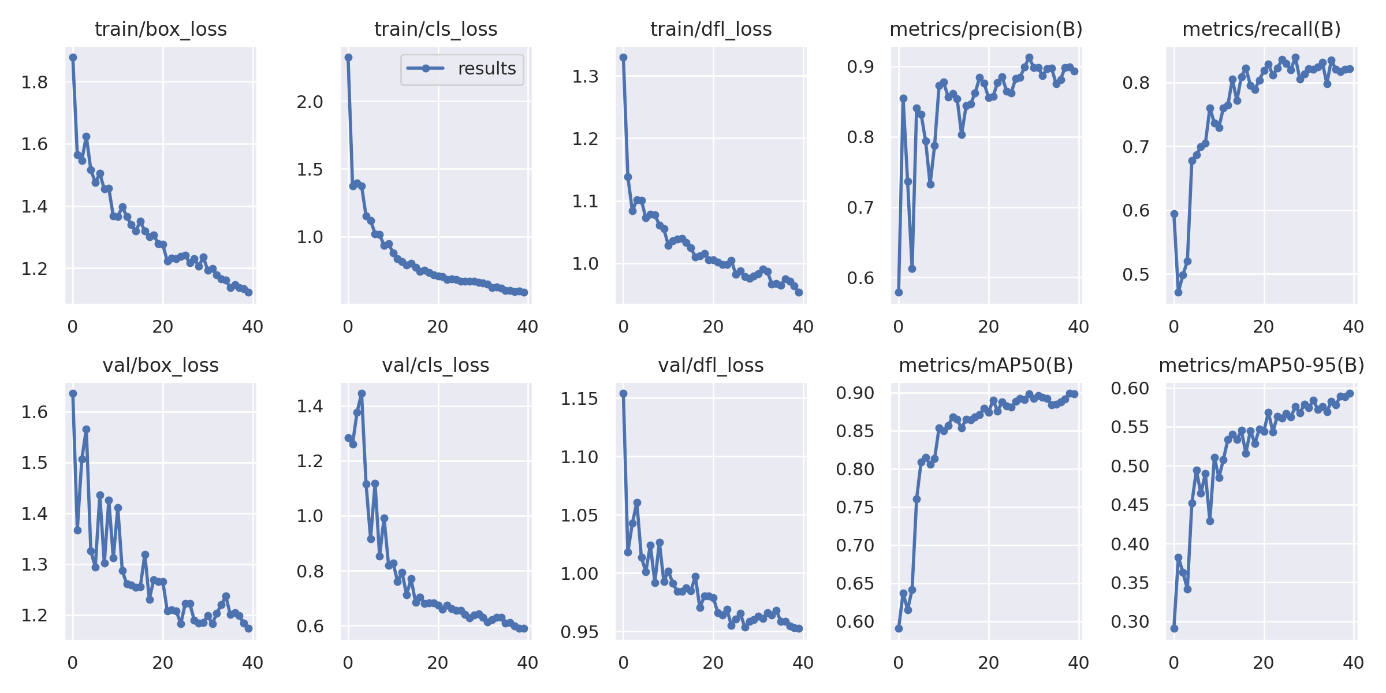
* Batch = 2, Epoch = 100



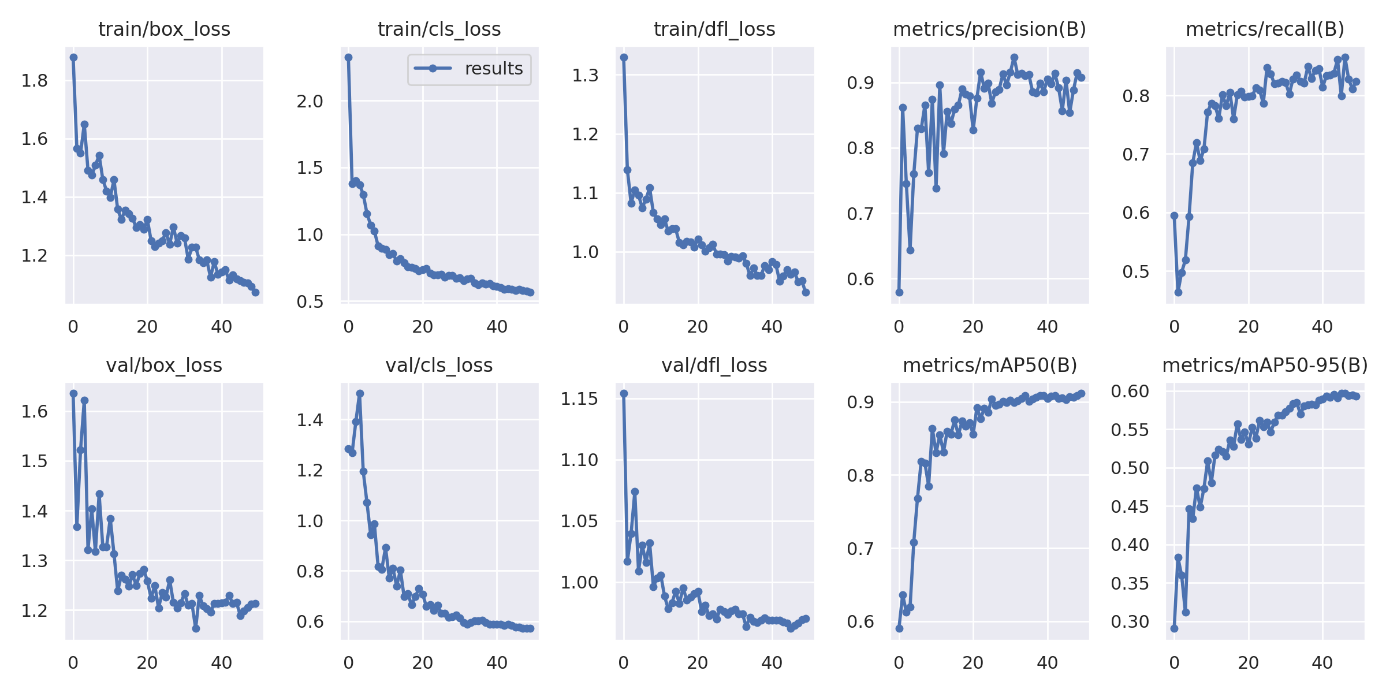
* Batch = 5, Epoch = 30



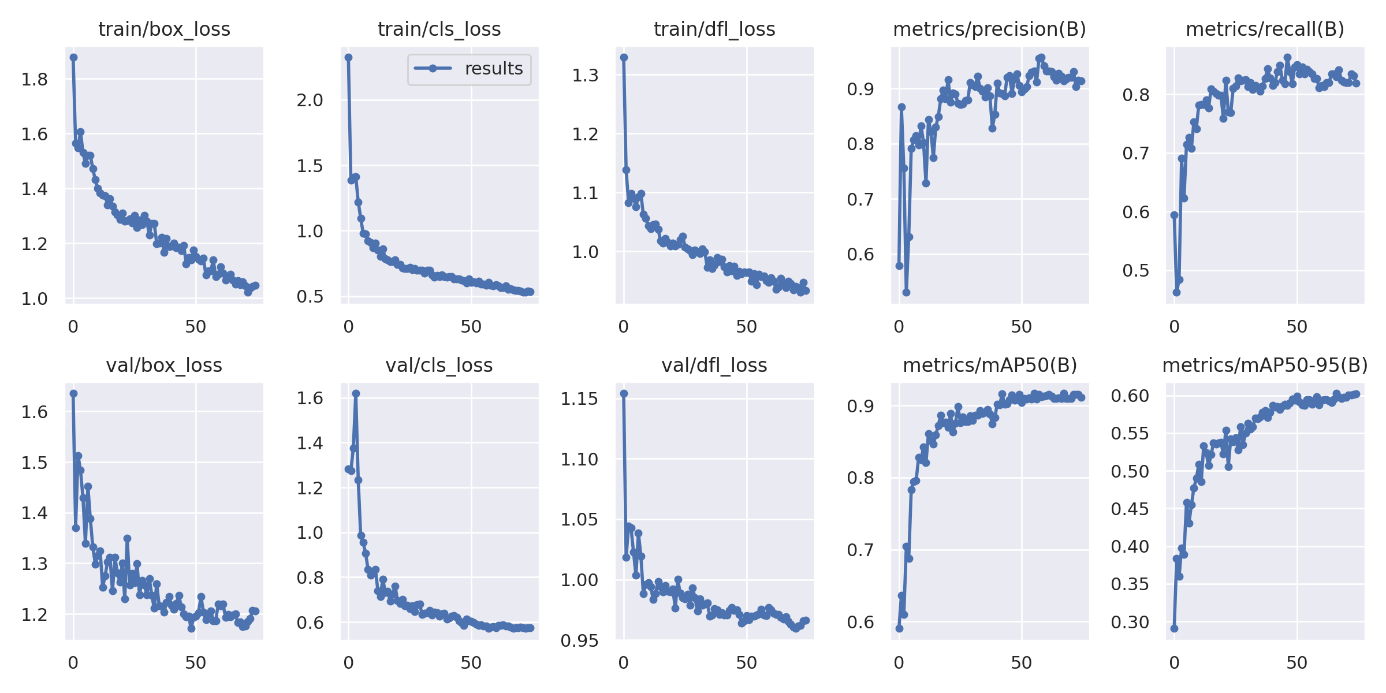
* Batch = 5, Epoch = 40



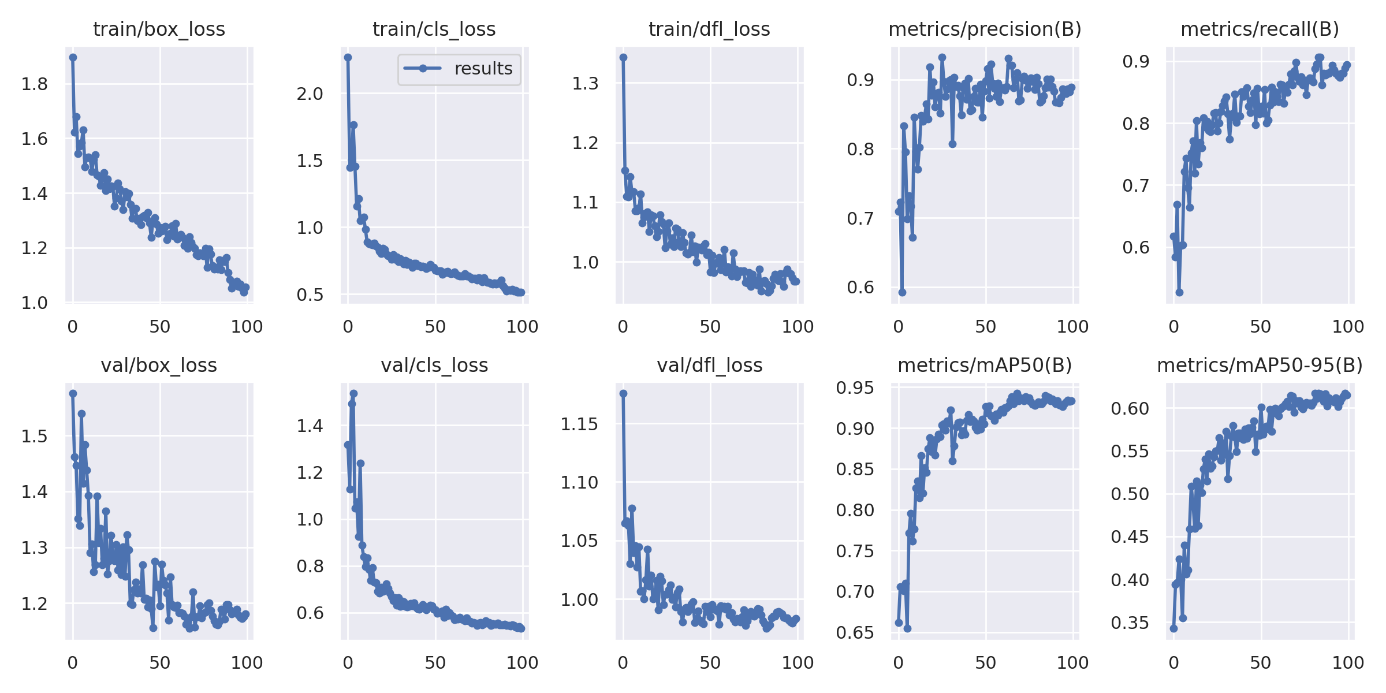
* Batch = 5, Epoch = 50



* Batch = 5, Epoch = 75



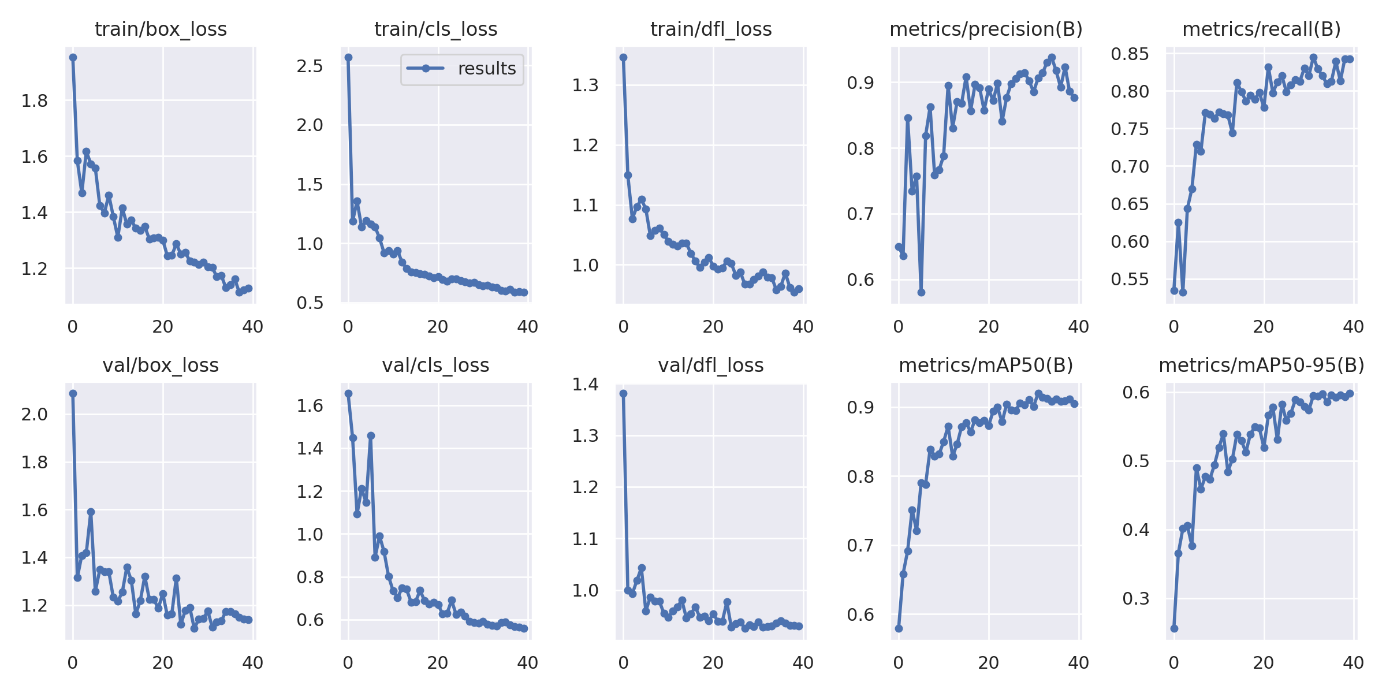
* Batch = 5, Epoch = 100



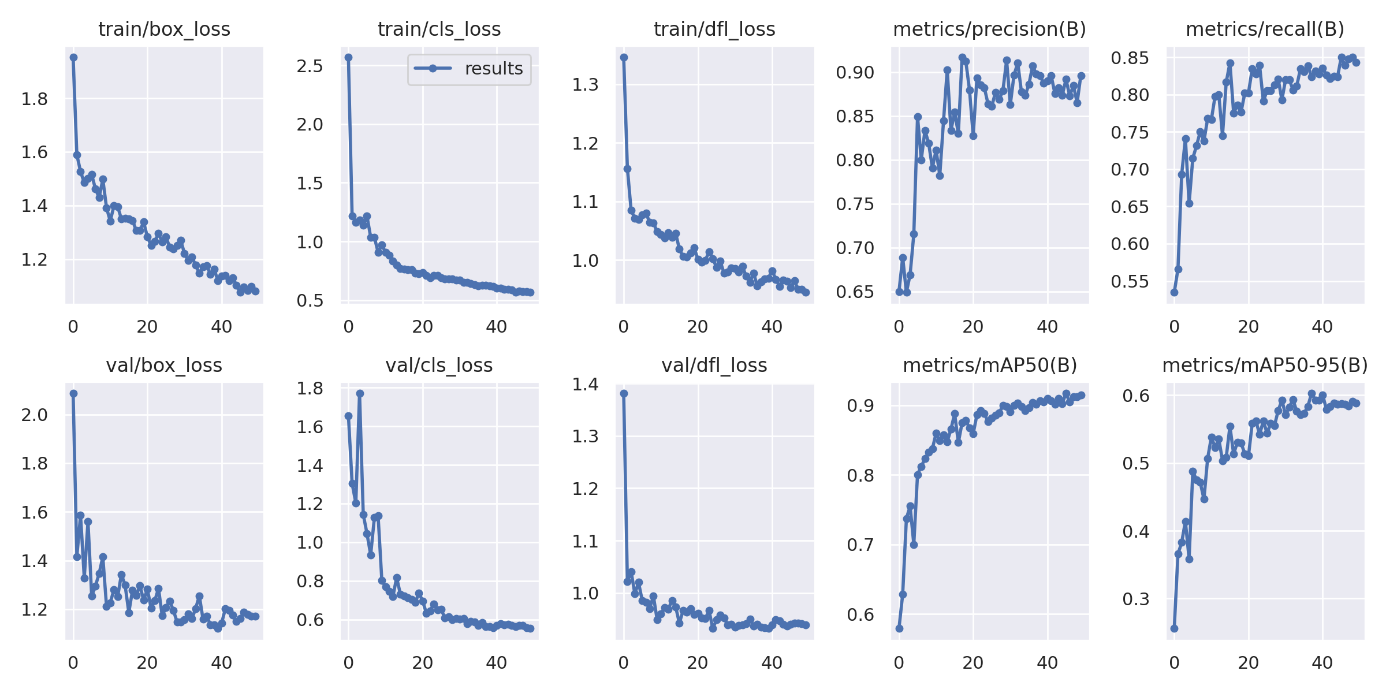
* Batch = 8, Epoch = 30



* Batch = 8, Epoch = 40



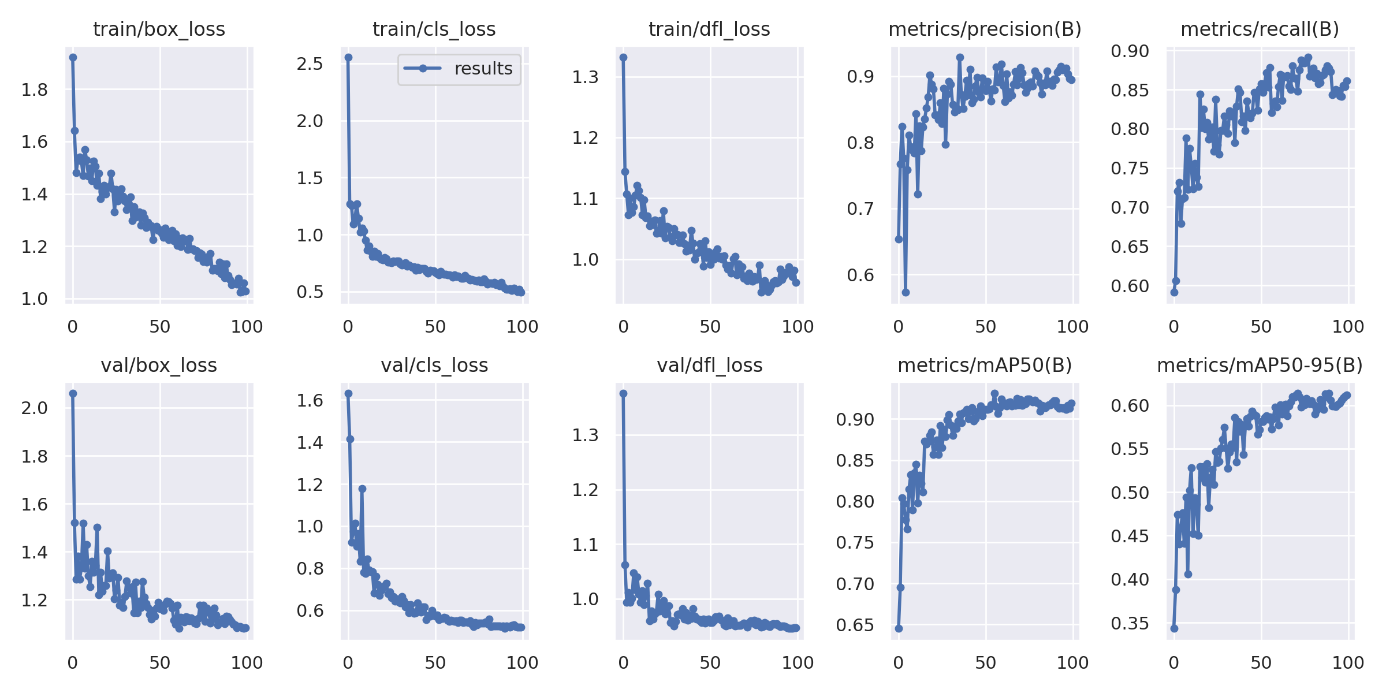
* Batch = 8, Epoch = 50



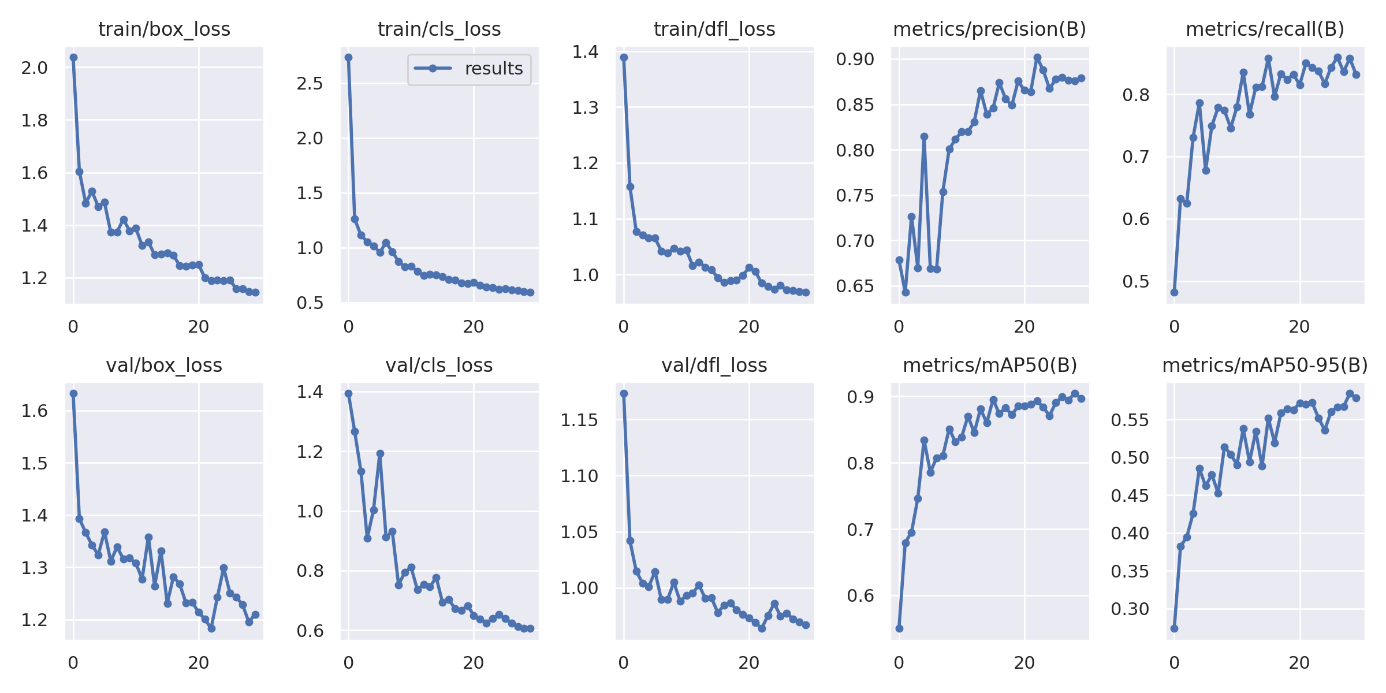
* Batch = 8, Epoch = 75



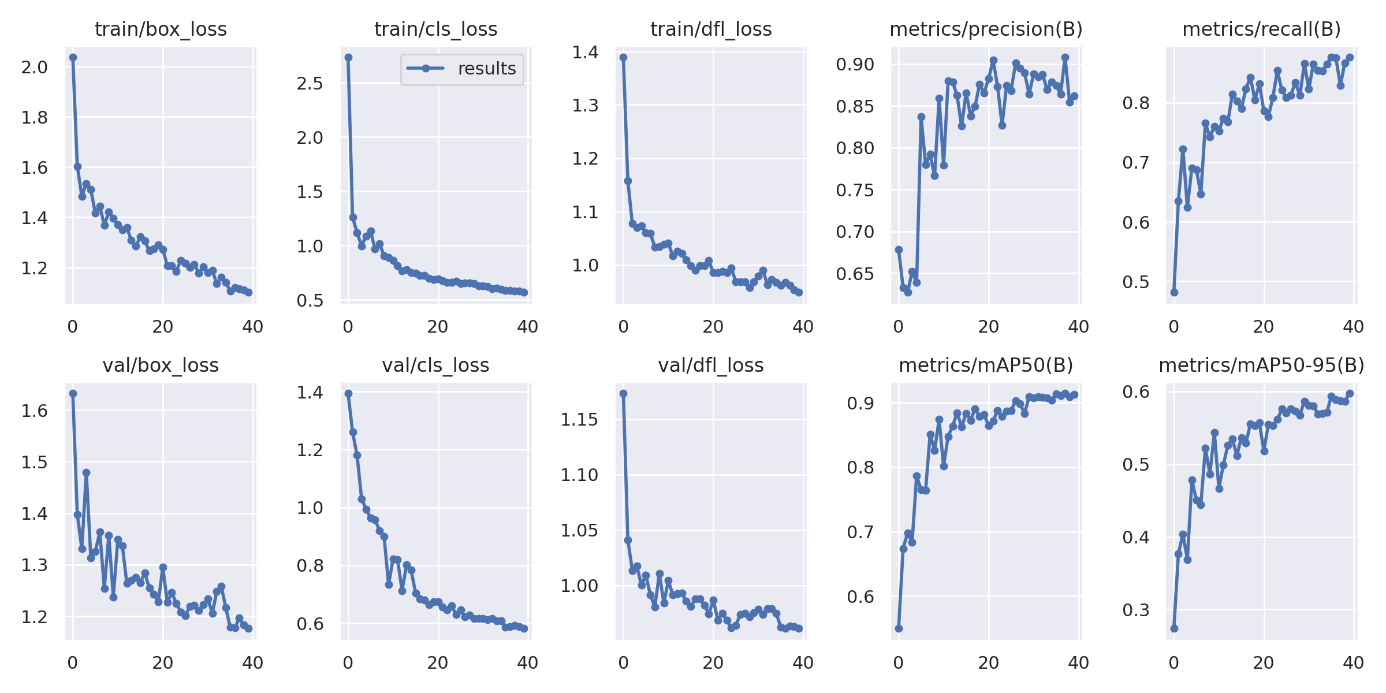
* Batch = 8, Epoch = 100



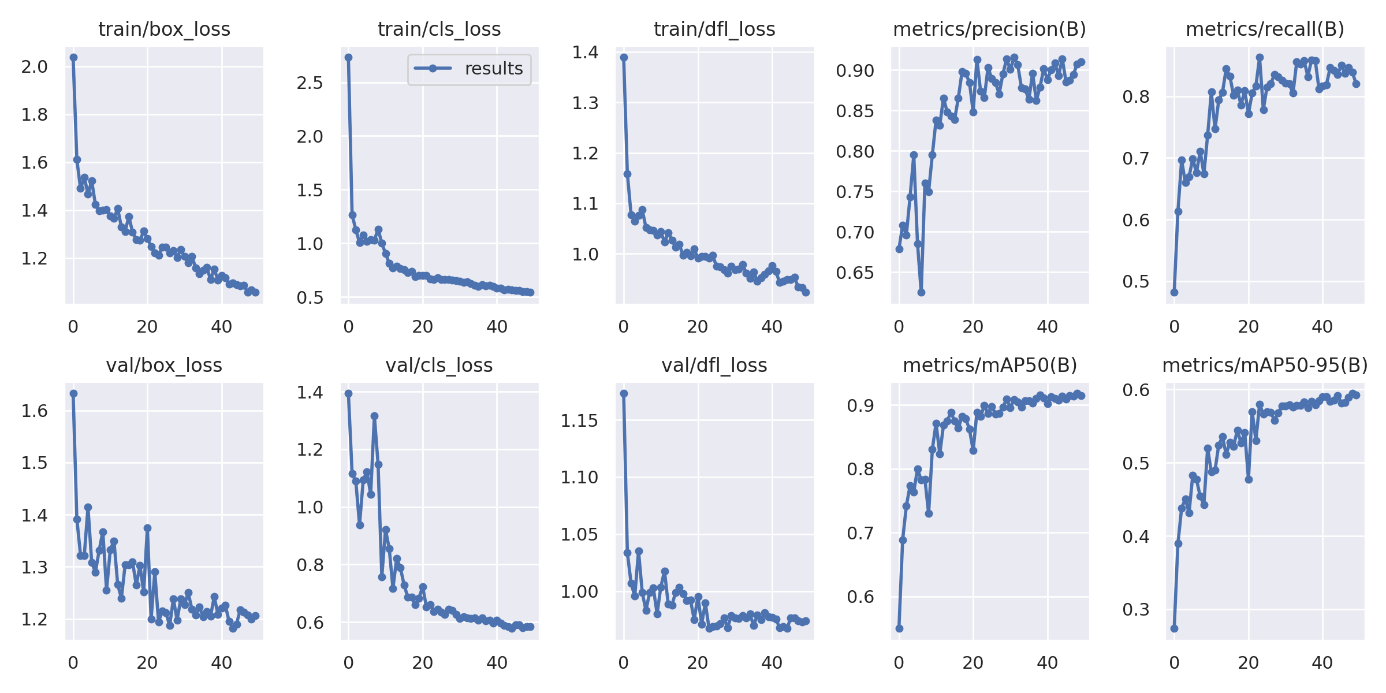
* Batch = 10, Epoch = 30



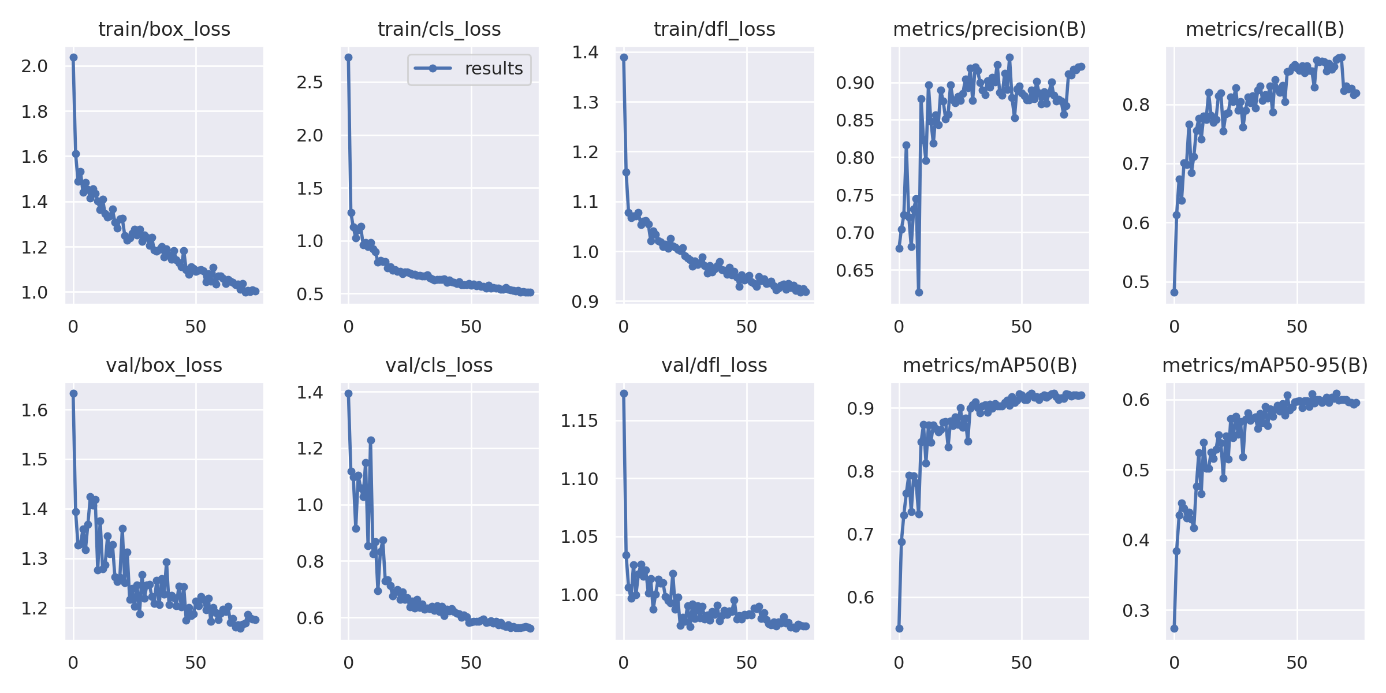
* Batch = 10, Epoch = 40



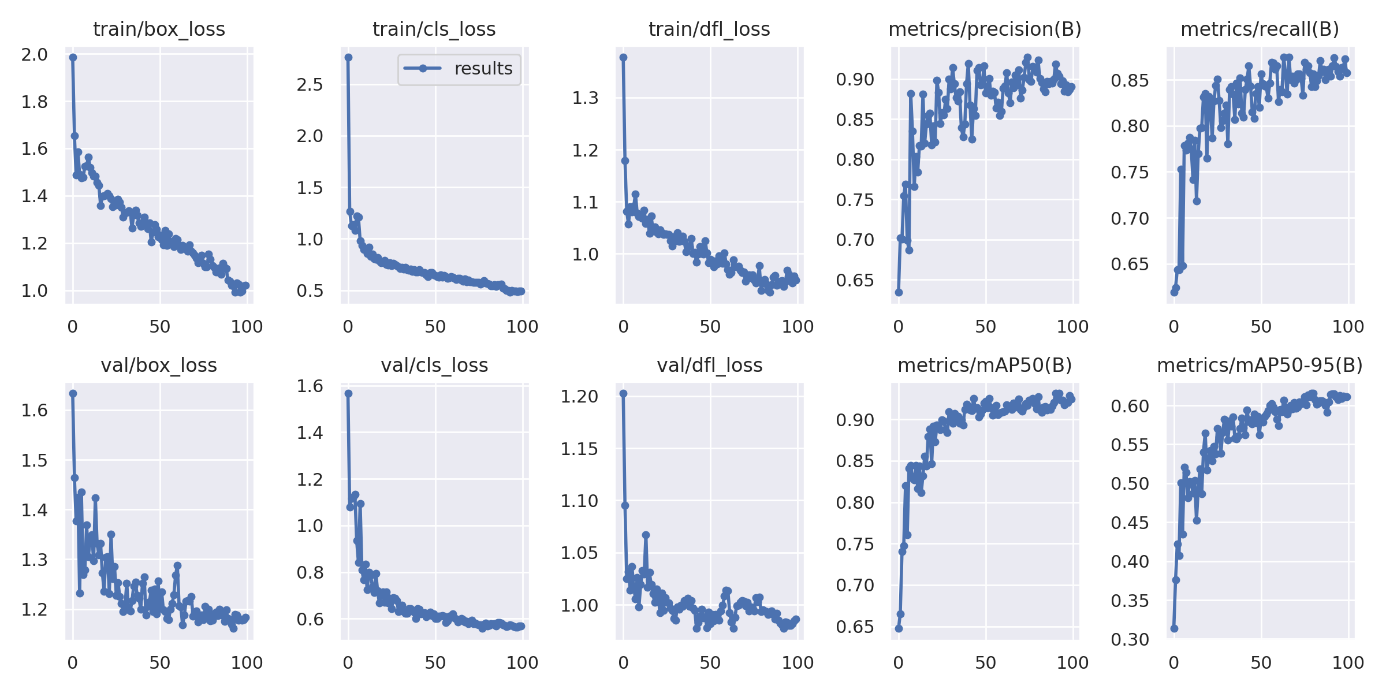
* Batch = 10, Epoch = 50



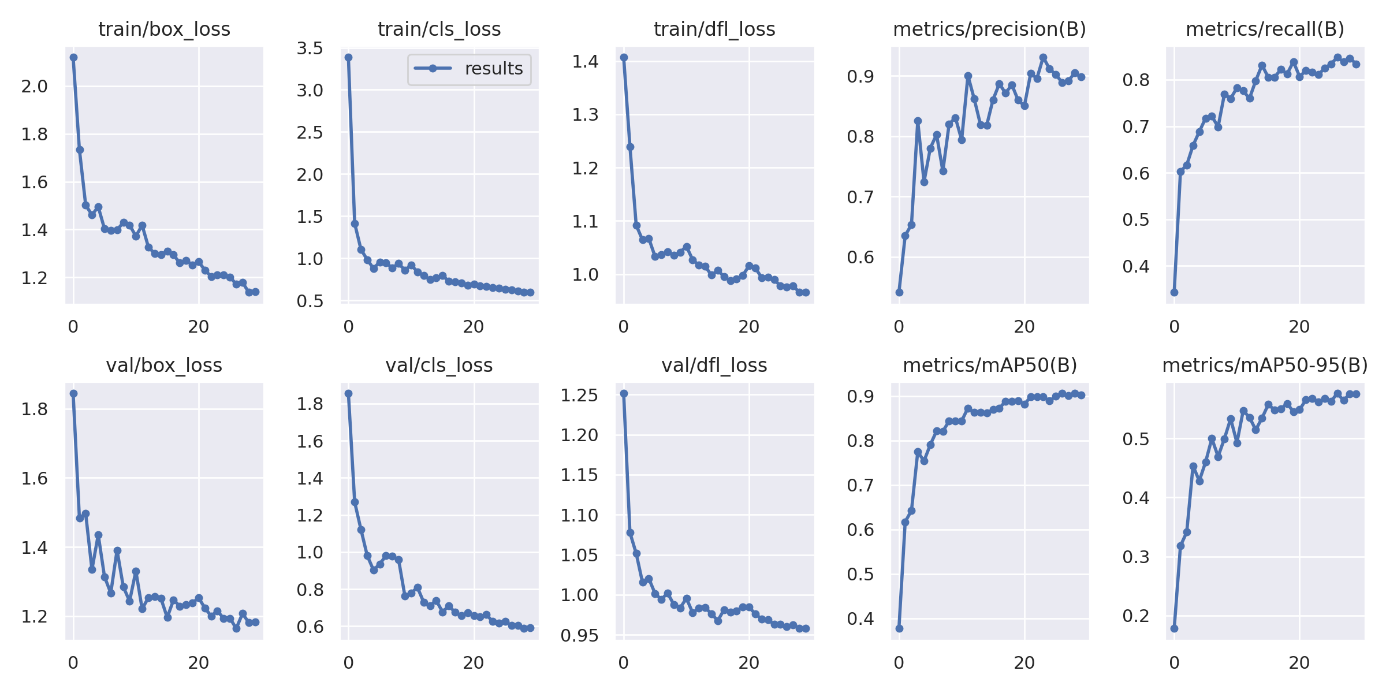
* Batch = 10, Epoch = 75



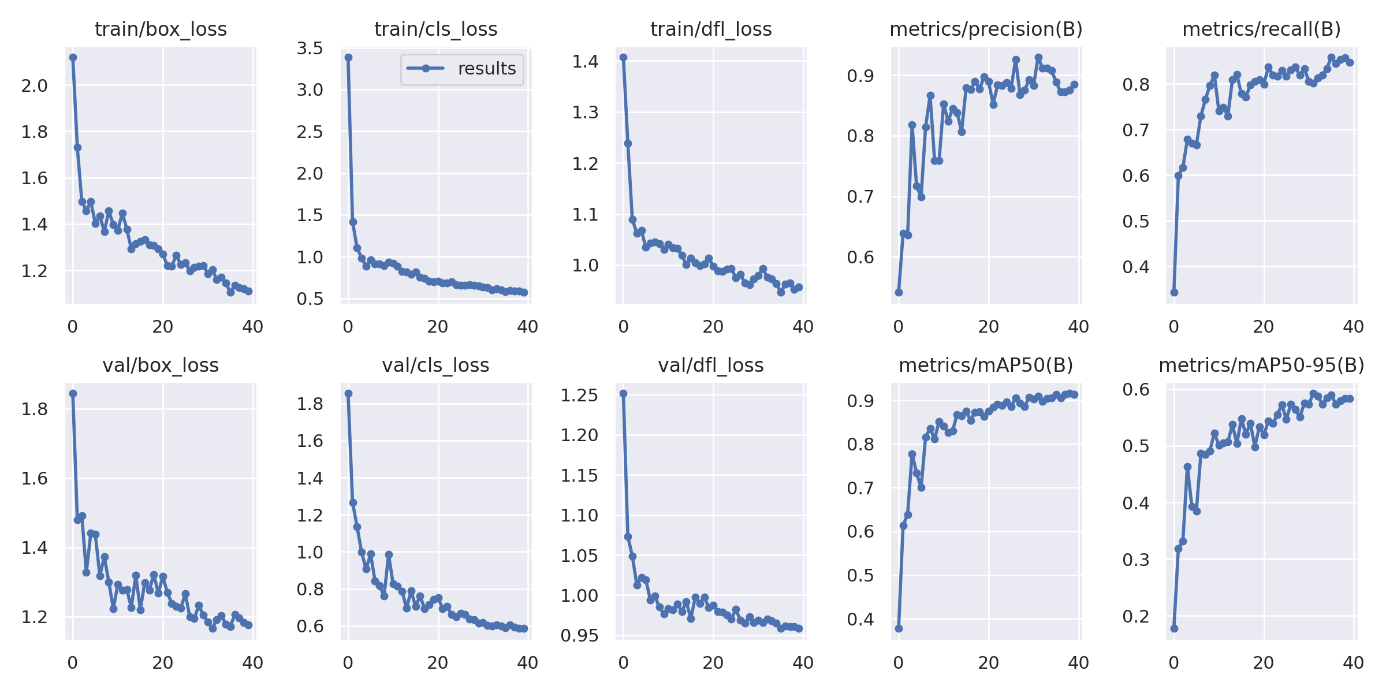
* Batch = 10, Epoch = 100



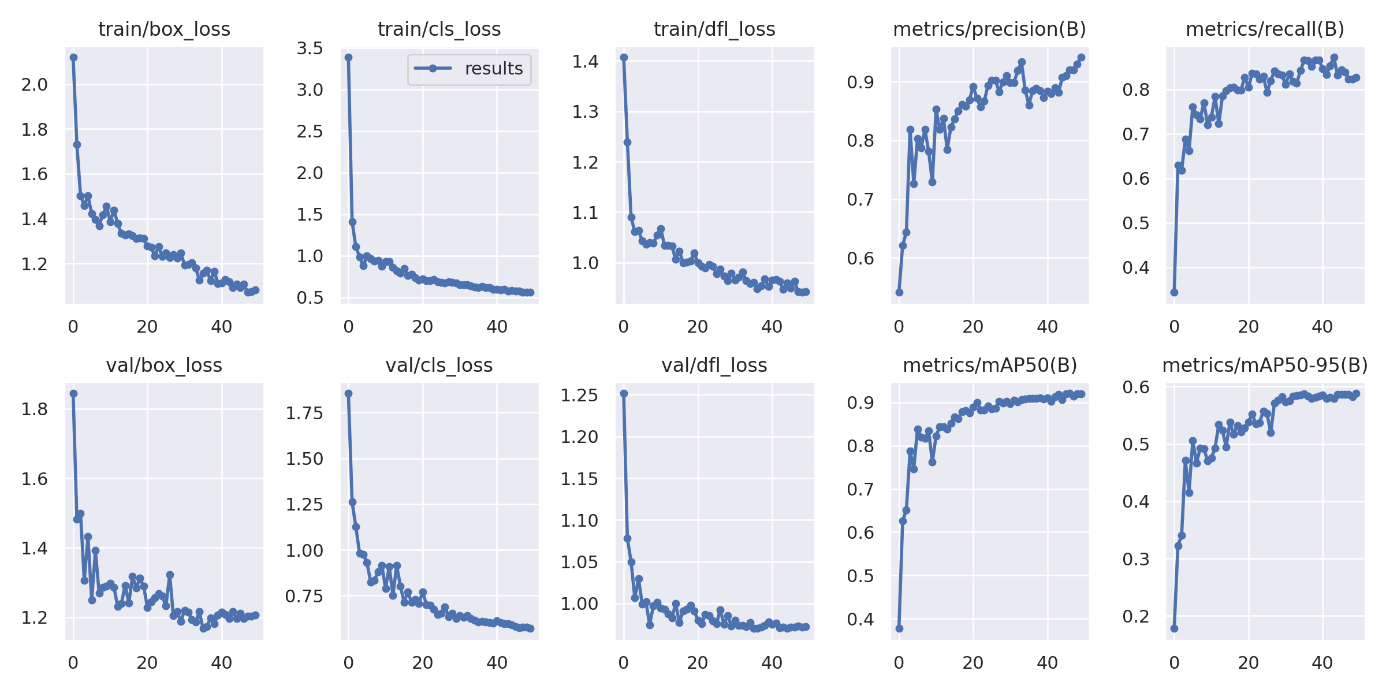
* Batch = 16, Epoch = 30



* Batch = 16, Epoch = 40

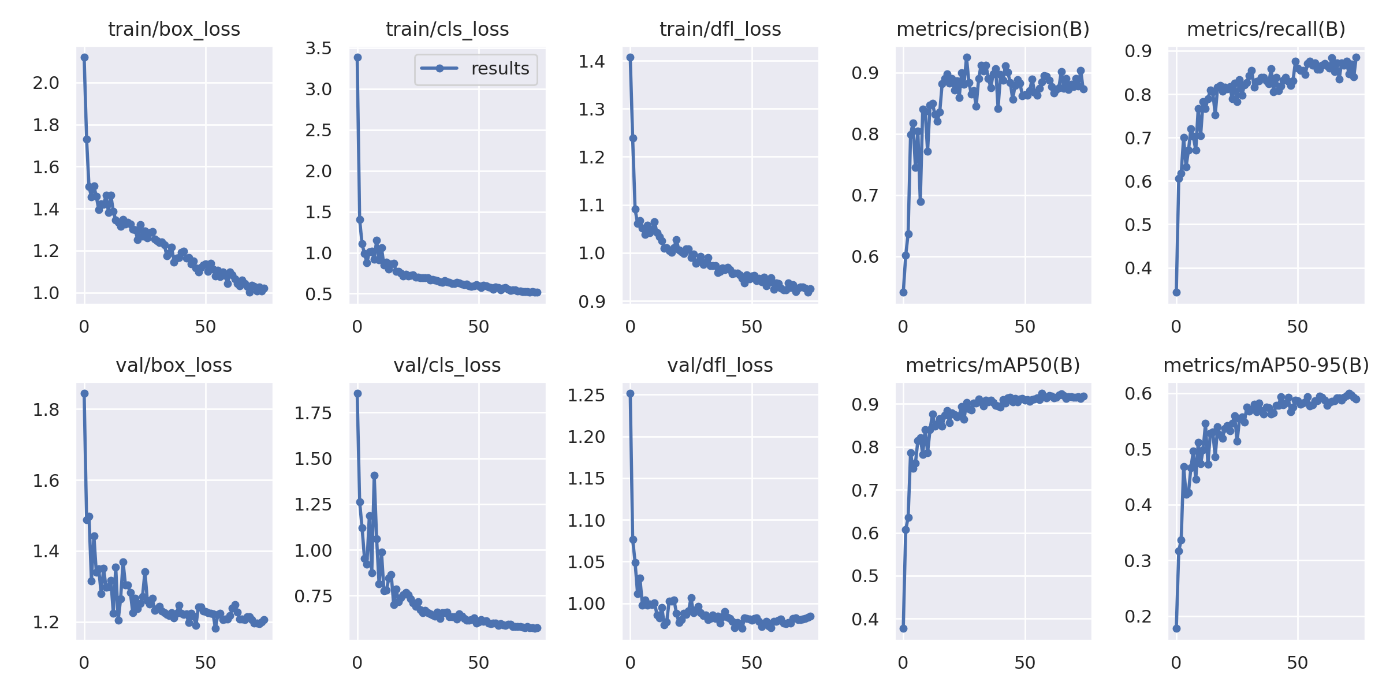


* Batch = 16, Epoch = 50



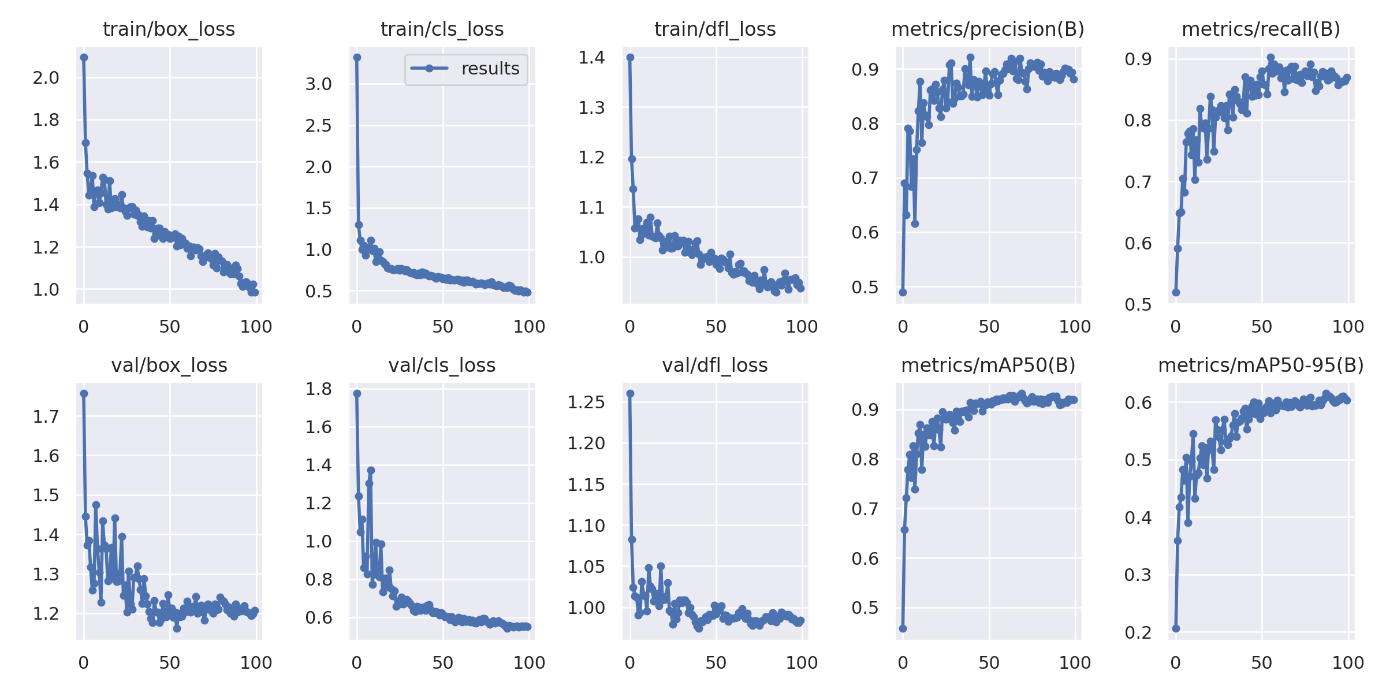


* Batch = 16, Epoch = 75





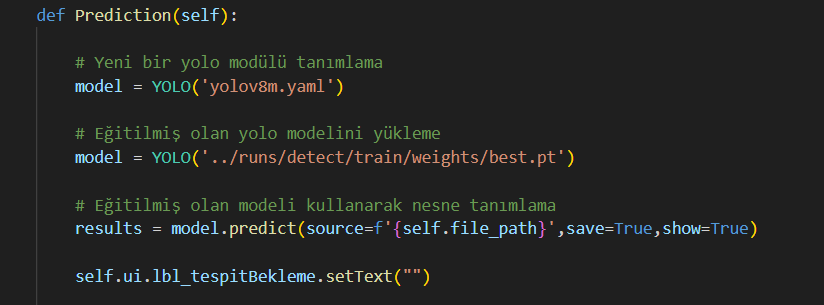
* Batch = 16, Epoch = 100





1. **EĞİTİLEN ALGORİTMAYLA NESNE TESPİTİ YAPMA**

**-** Yolo sürümümüzü yazıyoruz. Eğitilmiş olan modelimizin dosya yolunu giriyoruz. model.predict() kısmında nesne tespiti yapılacak olan dosyanın yolunu giriyoruz ve kayıt olmasını sağlamak için save = true, göstermek için de show = true yazıyoruz.



1. **NESNE TESPİT UYGULAMASI TASARLAMA**

**-**Kütüphanelerimizi tanımlıyoruz.

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**-**PyQt5 in QT designer uygulaması aracılığıyla arayüzümüzün tasarımını yapıyoruz. Tasarım bittikten sonra çalıştığımız dosyanın içine ui uzantılı çıktımızı atıyoruz.

**metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

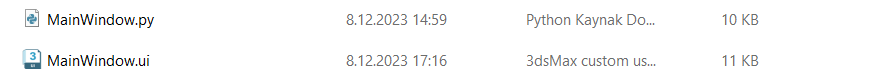
Açıklama otomatik olarak oluşturuldu**

-Dosyadaki görünümü

****

**-**ui uzantılı dosyayı py uzantılı dosyaya terminal üzerinden dönüştürüyoruz.





-Uygulamayı çağırmak için bir class tanımlıyoruz ve dönüştürdüğümüz py uzantılı dosyayı içinde tanımlıyoruz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

-Uygulamanın ve arayüzün temel elemanlarının tanımlandığı aynı zamanda fonksiyonların çağırıldığı init fonksiyonu bölümünü yazıyoruz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

- Çalıştığımız dosya yolunu tanımlıyoruz ve istenen formata uyarlıyoruz. Nesne tespiti yapacağımız dosyanın yolunu tanımlıyoruz. Default saniye, epoch, batch değerlerini belirliyoruz. Dosya yolu ve saniyenin texboxlarını değiştirilemez yapıyoruz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

-Araç ve insan sayısının texbox’ının içini boşaltıyoruz. dosya = QFileDialog() kısmında dosya seçme sayfasını açıyoruz ve seçilebilecek formatlarımızı belirliyoruz. Dosya yolunu kaydediyoruz. Dosya ismi değişkenine dosya yolunun son kısmını yani dosya ismini atıyoruz. Dosya ismini ekranda gösteriyoruz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

-Dosyamız mp4 formatlıysa saniye textbox’ını değiştirilebilir yapıyoruz. Dosya yolundaki videonun bilgilerini cv2 kütüphanesiyle dataya atıyoruz. Frame ve fps değerlerini kaydediyoruz. Frame’i fps e bölüp saniyeyi hesaplıyoruz. Sonra saniyeyi kaydedip textbox’a yazıyoruz. Eğer mp4 formatlı değilse textbox’ı değiştirilemez olarak bırakıyoruz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**-** Classın içine training adında bir fonksiyon açıyoruz. Fonksiyonun içinde daha önceden yapılmış olan bütün train dosyalarını siliyoruz.

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**-** Kullanıcıdan alınan epochs ve batch değerlerinin girilip girilmediğini kontrol ediyoruz. Kullanıcı eğer değer girmemişse sabit olan değerleri kulanıyoruz.

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

-Yolo modülümüz tanımlayıp model.train() kısmında data yolumuzu, epochs ve batch değerlerini alıp eğitimi başlatıyoruz. Validation işlemini yapıyoruz. Modeli onnx formatında kaydediyoruz.

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**-** Classın içine prediction adında bir fonksiyon açıyoruz. Uygulama başladığında nesne sayılarının text değerlerini sıfırlıyoruz. Kullanıcıdan saniye değeri alıyoruz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

-Uygulamadan ortalamasının hesaplanması istenilen nesneyi okuyoruz. Ve ekrana “nesne tespiti yapılıyor” yazısını yazdırıyoruz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

-Nesne tespitini başlatıyoruz. Nesne tespiti bittiğinde “nesneler tespit ediliyor” yazısını ekrandan siliyoruz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

- Nesne tespiti yapılacak dosya yolu mp4 ise seçilen nesnelerin saniye başına düşen nesne oranını hesaplıyoruz.

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

- Nesne tespiti yapılacak dosya yolu jpg ise seçilen nesnelerin sayısını buluyoruz.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

-Seçilen nesnelerin ortalamasını ekranda gösteriyoruz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

-Uygulamayı başlatmak için bir fonksiyon yazıyoruz ve yazılan bu fonksiyonu çağırıyoruz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu