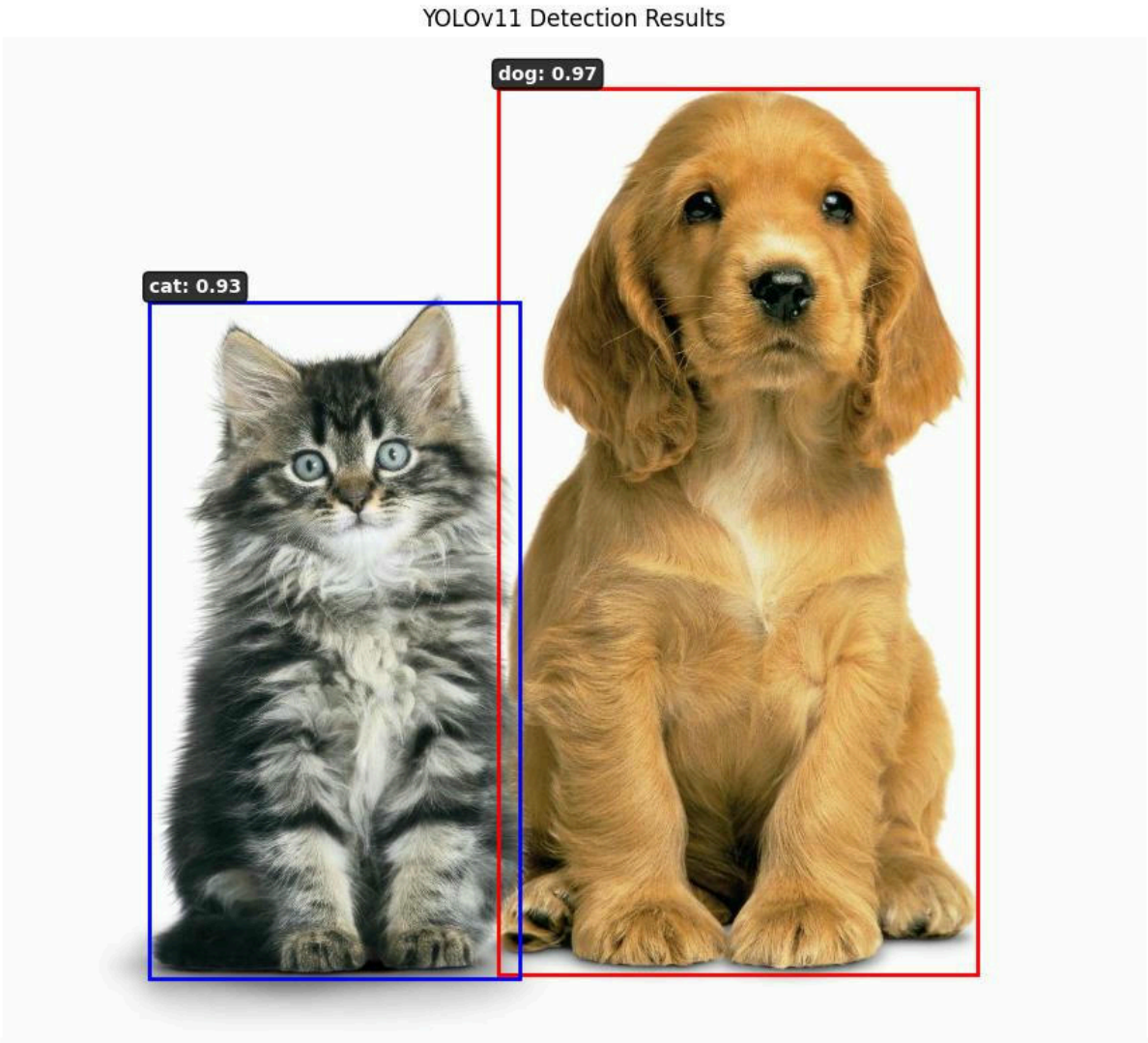
Deep Learning – Project_2

Pooya Hatami - 4031334011

2025-08-11

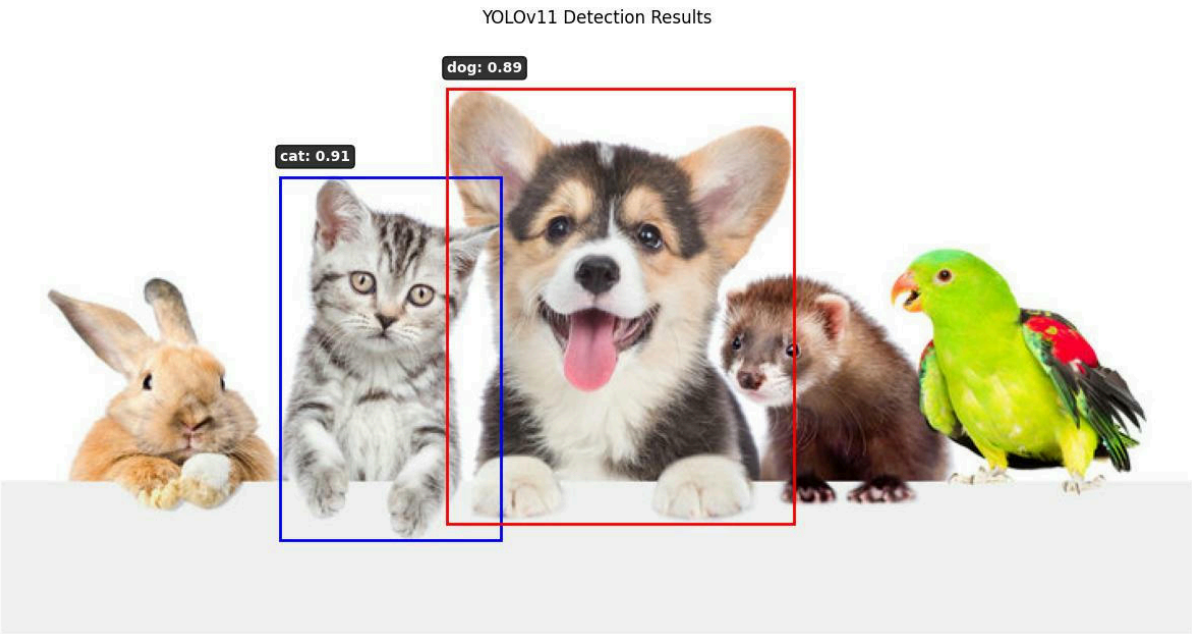
استفاده از مدل YOLO

در این روش از مدل YOLOv11L برای inference بدون هیچ گونه تغییر استفاده شده و همچنین مقدار حد نصاب اطمینان[۲] برای در نظر گرفتن جعبه فراگیر[۱] برابر با ۰.۷ در نظر گرفته شده است. با توجه به توزیع متفاوت دو داده تست، مدل سنگین و حدنصاب بالایی در نظر گرفته شده است تا به نتیجه مطلوب دست یابییم.



شکل ۱: نتیجه مدل YOLOv11L بر روی عکس cat_and_dog

[۱] Bounding Box
[۲] Confidence Threshold



شکل ۲: نتیجه مدل YOLOv11L بر روی عکس random

لینک نوت‌بوک YOLO در Colab

Open in Colab

استفاده از مدل سفارشی

در این قسمت از یک مدل سفارشی با الهام از معماری YOLOv2 و با پشتوانه ResNet50 استفاده می‌شود به اینصورت که وجود سگ و گربه و همچنین موقعیت سگ و گربه در یک مرحله پیش‌بینی می‌شود.

مجموعه‌داده

برای آموزش از کلاس سگ و گربه مجموعه‌داده Coco2017 استفاده شد. قسمت‌های آموزش و اعتبارسنجی با هم ادغام شده و سپس به صورت هوشمندانه‌تر جدا می‌شوند. کلیه کدهای مربوط به این قسمت در فایل `coco_downloader.py` موجود می‌باشد.

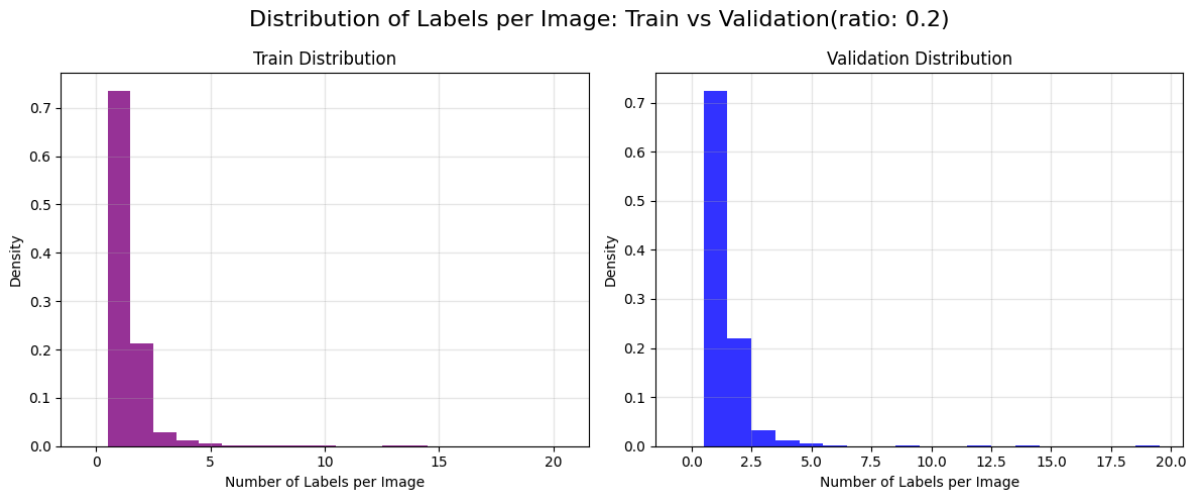
مجموعه‌داده	تعداد تصاویر شامل سگ و گربه	تعداد تصاویر شامل فقط سگ	تعداد تصاویر شامل فقط گربه	مجموع
Coco2017 (train+valid)	۲۲۰	۴۳۴۲	۴۰۷۸	**۸۶۴۰**

جدول ۱: مقایسه مجموعه‌داده‌های مورد استفاده

برای آموزش و اعتبارسنجی تمامی تصاویر شامل سگ و گربه(به علت تعداد کم) انتخاب می‌شوند سپس هر داده‌های هر کلاس بر اساس تعداد Bounding Box مرتب شده و ۲۵۰۰ داده اول از هرکلاس انتخاب می‌شوند. با اینکار از تمام عکس‌های شامل چندین سگ یا گربه(که تعداد کمی هم دارند) استفاده می‌شود.

آماده‌سازی داده جهت آموزش

در این قسمت داده‌ها به دو قسمت آموزش و ارزیابی که نسبت آن توسط پارامتر `val_ratio` تعیین می‌شود، تقسیم می‌شوند.



شکل ۳: توزیع تعداد لیبل ها در داده‌های آموزش و ارزیابی(نسبت ۰.۲)

پیش‌پردازش ثابت انجام‌شده روی همه داده‌ها عبارتند از:

- تغییر ابعاد عکس‌ها به (۲۲۴، ۲۲۴) متناظر ResNet50 از پیش آموزش دیده[۳] برروی ImageNet
- نرمال‌سازی عکس‌ها با میانگین و انحراف از معیار مدل ResNet50

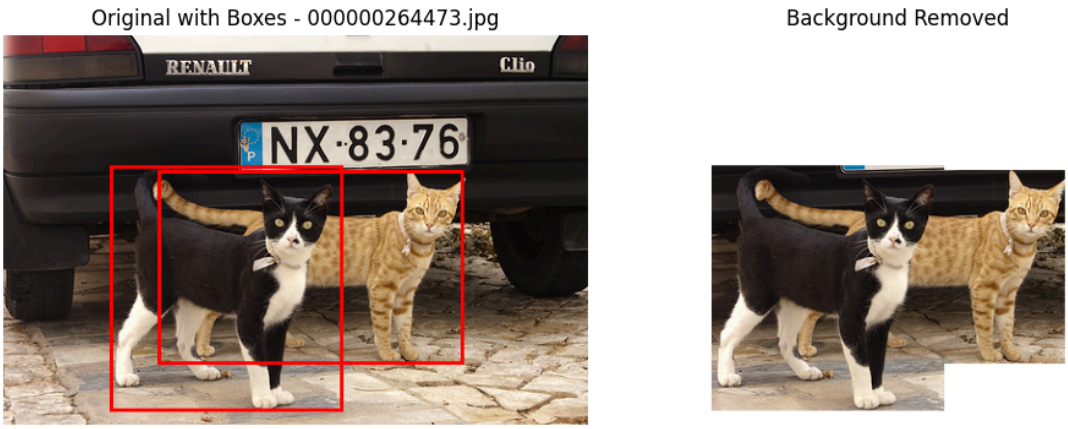
همچنین برای جلوگیری از بیش‌برازش[۴] از Augmentation های مختلف نیز استفاده شده است که استفاده و احتمال اعمال آن‌ها توسط پارامتر قابل تنظیم می‌باشد.

```
def get_train_augmentations(target_size=(224, 224), p=0.5):
    transforms_list = [
        A.HorizontalFlip(p=0.5),

        A.OneOf([
            A.RandomBrightnessContrast(
                brightness_limit=0.15, contrast_limit=0.15, p=1.0),
            A.HueSaturationValue(
                hue_shift_limit=10, sat_shift_limit=15,
                val_shift_limit=10, p=1.0),
        ], p=p),

        A.OneOf([
            A.MotionBlur(blur_limit=3, p=1.0),
```

[۳] Pretrained
[۴] Overfitting

```
            A.GaussianBlur(blur_limit=3, p=1.0),
        ], p=p * 0.2),

        A.Affine(
            translate_percent={'x': (-0.05, 0.05), 'y': (-0.05, 0.05)},
            scale=(0.9, 1.1),
            rotate=(-10, 10),
            p=p * 0.7
        ),

        A.RandomSizedBBoxSafeCrop(
            height=target_size[0],
            width=target_size[1],
            erosion_rate=0.2,
            p=p * 0.3
        ),

        A.Normalize(mean=(0.485, 0.456, 0.406),
                    std=(0.229, 0.224, 0.225)),
        ToTensorV2(),
    ]

    return A.Compose(transforms_list,
        bbox_params=A.BboxParams(format='albumentations',
        label_fields=['class_labels']))

```

به دلیل اختلاف بین توزیع عکس‌های آموزش با دو عکس تست(عکس‌های آموزش عکس از حیوانات در محیط‌های طبیعی و با کیفیت کم و دو عکس تست در استودیو و با کیفیت بالا می‌باشند)، با یک روش ساده، به صورت تصادفی و با احتمال قابل تنظیم، سعی می‌شود پس‌زمینه عکس‌ها به سفید تغییر یابد که در ادامه خروجی آن آورده شده است:



شکل ۴: حذف پس‌زمینه عکس‌ها

معماری مدل

معماری مدل سفارشی `SimpleResNetYOLO` شامل دو قسمت اصلی می‌باشد:

شبکه پشتیبان (Backbone)

از مدل ResNet50 پیش‌آموزش‌دیده بر روی مجموعه‌داده ImageNet به عنوان شبکه پشتیبان استفاده شده است. برای سازگاری با معماری YOLO، لایه‌های fully connected انتهایی(دو لایه آخر) حذف شده و فقط قسمت convolutional باقی مانده است.

برای پشتیبانی از grid size های مختلف، تنظیمات خاصی روی لایه چهارم ResNet50 انجام شده است:

- برای grid size ۱۴×۱۴ با stride: ۲۲۴ target size تبدیل به ۱ و dilation به ۲ تغییر می‌یابد
- برای grid size ۲۸×۲۸ با target size ۴۴۸: تنظیمات مشابه اعمال می‌شود

این تنظیمات باعث می‌شود که feature map خروجی ResNet50 با اندازه مطلوب تولید شود.

سر پیش‌بینی (Prediction Head)

سر پیش‌بینی شامل چهار لایه convolutional است که به صورت متوالی عمل می‌کنند:

۱. **لایه اول**: کاهش کانال‌ها از ۲۰۴۸ به ۲۵۶ با kernel سایز ۱
۲. **لایه دوم**: convolution با kernel سایز ۳، padding برابر ۲ و dilation برابر ۲
۳. **لایه سوم**: convolution با kernel سایز ۱ برای ترکیب feature ها
۴. **لایه آخر**: تولید خروجی نهایی بدون activation

سه لایه اول دارای ReLU، Batch Normalization و Dropout با نرخ ۰.۴ هستند. لایه آخر bias خود را به گونه‌ای مقداردهی اولیه می‌کند که احتمال اولیه objectness برابر ۰.۰۱ باشد.

خروجی نهایی برای هر anchor box شامل:

- ۴ مختصات bounding box (x, y, w, h)
- ۱ امتیاز اطمینان (objectness score)
- ۲ امتیاز کلاس (سگ و گربه)

با grid size ۱۴×۱۴ و ۵ anchor box، تعداد کل پیش‌بینی‌ها برابر با $۱۴ \times ۱۴ \times ۵ \times ۷ = ۶۸۶۰$ خواهد بود.

استراتژی آموزش

برای بهینه‌سازی فرآیند آموزش، از استراتژی "Progressive Unfreezing" استفاده شده است:

۱. **مرحله اول** (تا epoch ۱۲): تمام لایه‌های backbone منجمد، نرخ یادگیری ۱e-۳
۲. **مرحله دوم** (epoch ۱۲ تا ۲۴): دو لایه آخر backbone آزاد، نرخ یادگیری ۱e-۵
۳. **مرحله سوم** (epoch ۲۴ به بعد): تمام لایه‌های backbone آزاد، نرخ یادگیری ۱e-۶

این استراتژی باعث تطبیق تدریجی شبکه پشتیبان با داده‌های هدف می‌شود.

Anchor Box Generation

برای تولید anchor box ها از الگوریتم K-Means با ۳ خوشه بر روی ابعاد bounding box های داده‌های آموزش استفاده شده است. سپس دو anchor اضافی با نسبت‌های ۱۶:۹ و ۹:۱۶ بر اساس بزرگترین anchor محاسبه شده اضافه می‌شوند.

Anchor box های نهایی محاسبه شده:

```
[[ 2.27  2.54]
 [ 8.25  4.64]
 [ 4.64  8.25]
 [ 5.78  6.62]
 [10.18 10.93]]
```

تنظیمات آموزش

پارامترهای نهایی آموزش بر اساس آزمایش‌های انجام شده:

پارامتر	مقدار
Grid Size	۱۴×۱۴
Target Size	۴۴۸×۴۴۸
Batch Size	۸
Epochs	۳۰
Dropout Rate	۰.۴
Weight Decay	۱e-۳
Coord Weight	۱.۰
Augmentation Strength	۰.۵
Background Removal Probability	۰.۵

جدول ۲: پارامترهای نهایی آموزش

Loss Function

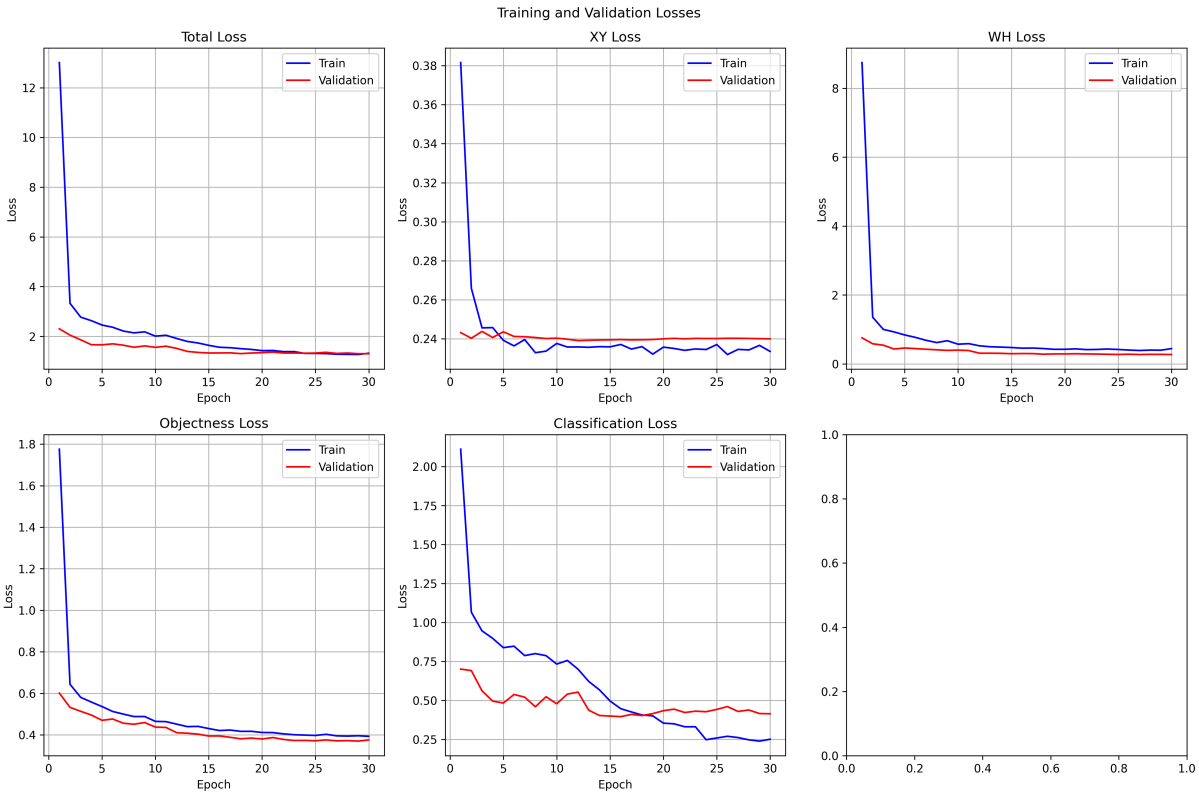
تابع هزینه ترکیبی شامل چهار بخش اصلی است:

۱. **XY Loss**: MSE loss برای مختصات مرکز bounding box
۲. **WH Loss**: MSE loss برای ابعاد bounding box (در فضای log)
۳. **Objectness Loss**: Focal Loss برای تشخیص وجود object با α=۰.۲۵ و γ=۲.۰
۴. **Classification Loss**: Binary Cross Entropy برای تشخیص نوع حیوان

```
total_loss = coord_weight * (xy_loss + wh_loss) + objectness_loss +
cls_loss
```

نتایج آموزش

مدل به مدت ۳۰ epoch آموزش داده شد. در طول آموزش، بهترین validation loss در epoch مناسب ذخیره شد:



شکل ۵: منحنی‌های loss در طول آموزش

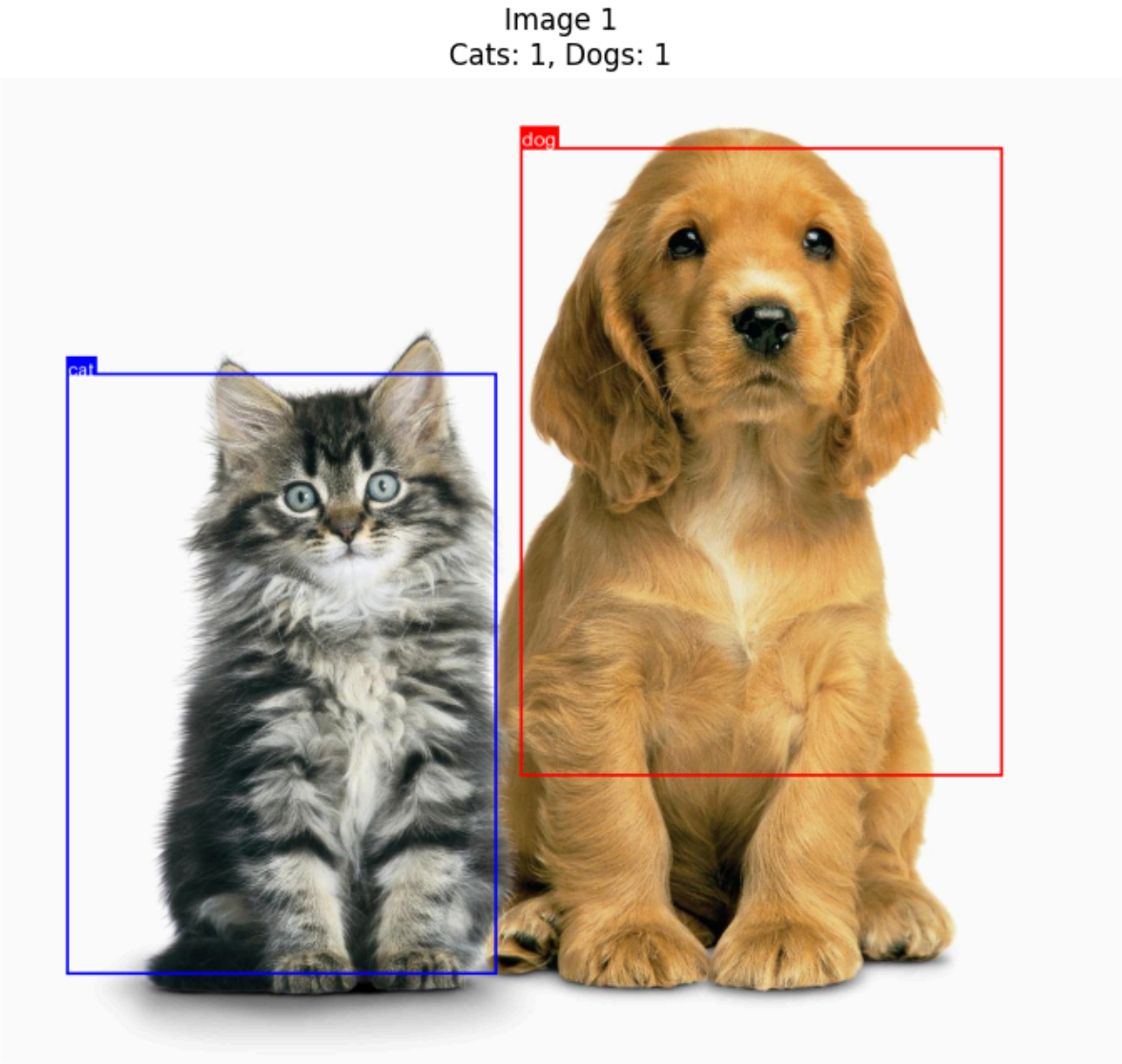
آنالیز عملکرد

جنبه	مدل سفارشی	YOLOv11L
دقت تشخیص	متوسط	بالا
سرعت inference	متوسط	سریع
تعداد پارامترها	۲۶M	۵۹M
اندازه مدل	کوچک	بزرگ
قابلیت تنظیم	بالا	پایین

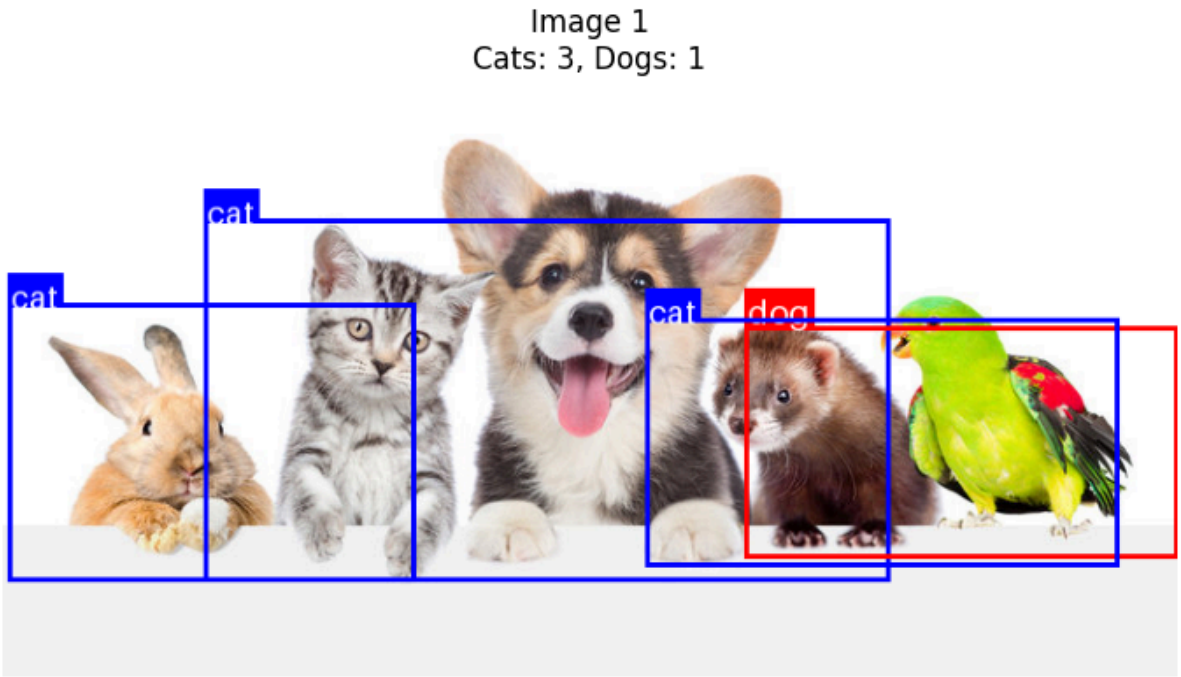
جدول ۳: مقایسه مدل سفارشی با YOLOv11L

ارزیابی بصری نتایج

نتایج حاصل از اعمال مدل سفارشی بر روی تصاویر تست:



شکل ۶: نتیجه مدل سفارشی روی عکس cat_and_dog



شکل ۷: نتیجه مدل سفارشی روی عکس random

شکل ۸: نتیجه مدل سفارشی روی دو عکس نمونه از مجموعه داده

تحلیل محدودیت‌ها و بهبودهای ممکن

مدل سفارشی با وجود پیاده‌سازی مناسب، محدودیت‌هایی نسبت به مدل‌های پیشرفته‌تر دارد:

محدودیت‌های فعلی

۱. **تعداد anchor box محدود**: استفاده از ۵ anchor box که ممکن است برای اشکال متنوع کافی نباشد
۲. **معماری ساده**: نبود feature pyramid یا multi-scale detection
۳. **تعداد کلاس محدود**: فقط دو کلاس سگ و گربه
۴. **عدم استفاده از تکنیک‌های پیشرفته**: مانند attention mechanisms یا transformer blocks

راه‌های بهبود

۱. افزایش تعداد anchor box ها با تحلیل دقیق‌تر داده‌ها
۲. پیاده‌سازی Feature Pyramid Network (FPN) برای multi-scale detection
۳. استفاده از data augmentation پیشرفته‌تر
۴. بهینه‌سازی تابع loss با وزن‌دهی هوشمندتر
۵. اعمال تکنیک‌های regularization اضافی

نتیجه‌گیری

در این پروژه، دو روش مختلف برای تشخیص سگ و گربه پیاده‌سازی شد که مدل سفارشی عملکرد بدتری را نسبت به مدل YOLOv11L به نمایش گزاشت

لینک مخزن گیتهاب

Link