

به نام خدا

گزارش تمرین سری هفتم درس بینایی کامپیوتر

نام مدرس: دکتر محمدرضا محمدی

فرزان رحمانی ۹۹۵۲۱۲۷۱

سوال ۱

منابع:

https://machinelearningprojects.net/train-yolov7-on-the-custom-dataset/#Step_3_%E2%80%93_Let%E2%80%99s_Prepere_the_data
<https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolov7-object-detection-on-custom-data.ipynb>

** فایل yml مورد استفاده در فایل زیر تمرین موجود است.

```
1 > ! dataset.yaml
1 train: /content/dataset/images/training/
2 val: /content/dataset/images/validation/
3
4 # number of classes
5 nc: 1
6
7 # class names
8 names: ['SolarPanel']
```

با مطالعه لینک های بالا و ریپو اصلی yolo و بخصوص readme آن به پیاده سازی این سوال پرداختیم.

توضیح کد:

کد پیاده سازی شده در واقع یک برنامه است که برای آموزش به کمک مدل YOLOV7 بر روی مجموعه داده های شامل تصاویر شبکه های خورشیدی استفاده می شود. این کد به ترتیب زیر اقدامات را انجام می دهد:

۱. Import کتابخانه ها و بسته های مورد نیاز: این بخش برای استفاده از کتابخانه ها و بسته های مورد نیاز در ادامه برنامه است. این کتابخانه ها شامل numpy، shutil، zipfile، matplotlib.pyplot، os، PIL.Image، cv2 و glob است.

۲. Mount کردن: Google Drive این بخش به منظور اتصال به گوگل درایو برای دسترسی به داده‌های مورد نیاز برنامه است. در این بخش گوگل درایو به کمک تابع `mount()` `drive.mount()` می‌شود تا به فایل‌های موجود در گوگل درایو دسترسی داشته باشیم.

۳. دانلود و آماده‌سازی مجموعه داده: این بخش شامل کدهایی است که برای دانلود و استخراج مجموعه داده از لینک داده شده استفاده می‌شود. مجموعه داده به صورت فایل فشرده دریافت می‌شود و سپس در مسیر مناسب استخراج می‌شود. در این بخش یک فایل فشرده شامل مجموعه داده‌های شامل تصاویر خورشیدی را از لینک داده شده دریافت می‌کنیم و سپس آن را در مسیر مورد نظر استخراج می‌کنیم.

۴. آماده‌سازی داده‌ها: این بخش شامل کدهایی است که برای آماده‌سازی داده‌ها قبل از آموزش مدل استفاده می‌شود. ابتدا مسیرهای مورد نیاز برای ذخیره تصاویر و لیبل‌ها تعریف می‌شوند و سپس تمام تصاویر و لیبل‌ها در مسیرهای مشخص شده کپی می‌شوند. سپس تصاویر و لیبل‌ها تغییر اندازه داده می‌شوند و در مسیرهای جدید ذخیره می‌شوند. در این بخش تصاویر و لیبل‌ها از مسیر منبع به مسیر آماده‌سازی منتقل می‌شوند و اندازه تصاویر تغییر می‌کند و در مسیرهای جدید ذخیره می‌شوند.

۵. ایجاد برچسب‌های YOLO برای هر تصویر: این بخش شامل کدهایی است که برای ایجاد برچسب‌های YOLO برای هر تصویر بر اساس مستطیلی که توسط مرز تمام شدن منحنی‌های تشخیص داده شده در تصویر ایجاد می‌شود. این برچسب‌ها به صورت فایل‌های متنی با پسوند `.txt` در کنار هر تصویر ذخیره می‌شوند. این بخش شامل کدهایی است که برای ایجاد برچسب‌های YOLO برای هر تصویر از مستطیل‌هایی که توسط مرزهای تشخیص داده شده در تصویر ایجاد می‌شوند، استفاده می‌کند.

۶. نمایش برچسب‌های YOLO بر روی تصاویر: این بخش شامل کدهایی است که برای رسم مستطیل‌های برچسب‌های YOLO بر روی تصاویر و نمایش تصاویر نهایی با برچسب‌ها استفاده می‌شود.

۷. تقسیم داده‌ها به دو بخش آموزش و اعتبارسنجی: این بخش شامل کدهایی است که برای تقسیم مجموعه داده به دو بخش آموزش و اعتبارسنجی به نسبت مشخص شده استفاده می‌شود. تصاویر و لیبل‌ها به دو بخش تقسیم شده و در مسیرهای جداگانه ذخیره می‌شوند. این بخش شامل کدهایی است که برای تقسیم مجموعه داده به دو بخش آموزش و اعتبارسنجی به نسبت مشخص شده استفاده می‌شود. تصاویر و لیبل‌ها به دو بخش تقسیم شده و در مسیرهای جداگانه ذخیره می‌شوند.

۸. Clone کردن مخزن: YOLOv7 این بخش از طریق دستور `git clone` مخزن YOLOv7 را دریافت می‌کند تا بتوانیم از آن برای آموزش و استفاده در ادامه برنامه استفاده کنیم. در این بخش مخزن YOLOv7 از طریق دستور `git clone` دریافت می‌شود.

۹. نصب و آماده‌سازی موارد مورد نیاز: این بخش شامل کدهایی است که برای نصب و آماده‌سازی موارد مورد نیاز برای آموزش YOLOv7 استفاده می‌شود. این کدها با استفاده از فرمان `pip install` موارد مورد نیاز را نصب می‌کنند.

۱۰. آموزش مدل: این بخش شامل کدی است که با استفاده از فرمان `python train.py` و با تنظیمات مشخص شده (مانند اندازه تصاویر، تعداد دسته‌ها، تعداد دوره‌ها، مسیر داده‌ها و وزن‌های مدل) مدل YOLOv7 را آموزش می‌دهد. در این بخش مدل YOLOv7 آموزش داده می‌شود با تنظیمات مشخص شده مانند اندازه تصاویر، تعداد دسته‌ها، تعداد دوره‌ها و مسیر داده‌ها.

۱۱. اجرای تشخیص بر روی تصاویر: این بخش شامل کدی است که با استفاده از فرمان `python detect.py` با استفرض کنید که برای آموزش مدل YOLOv7 بر روی مجموعه داده‌ای شامل تصاویر خورشیدی استفاده می‌کنیم. در این بخش مدل آموزش دیده را روی تصاویر تست اعمال می‌کند و نتایج تشخیص را نمایش می‌دهد.

ردیابی با تشخیص

- این خانواده از روش‌ها با معرفی روش SORT مطرح شدند
- این روش (و دیگر روش‌های مبتنی بر همین رویکرد) از سه گام اساسی تشکیل می‌شوند:

- تشخیص اشیاء

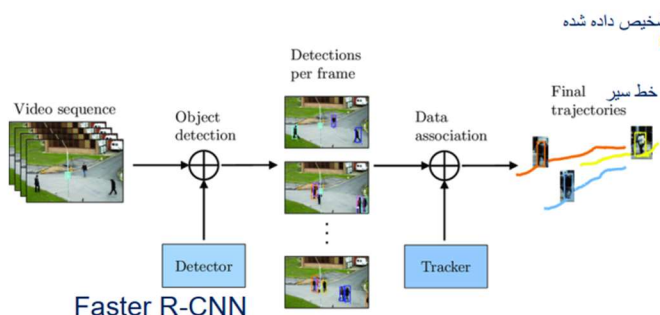
▪ Faster R-CNN

- قرابت‌سنجی (مبتنی بر ویژگی‌های مکانی و بصری)

▪ فقط از Kalman و IoU استفاده می‌کند

- انتساب assignment

▪ Hungarian



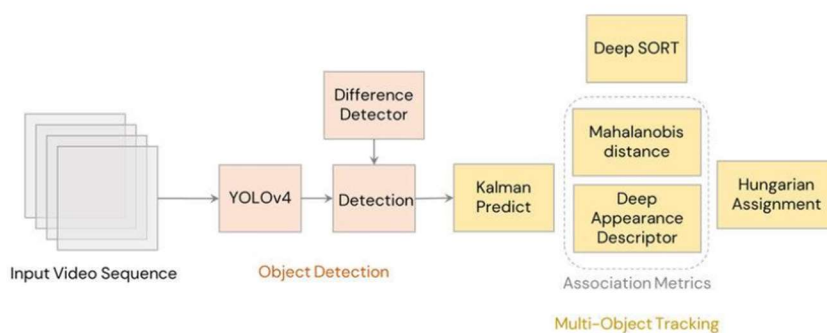
۲۶

Bewley, Alex, et al. "Simple Online and Realtime Tracking." ICIP, 2016.

DeepSORT

instead of Kalman and IoU

- در DeepSORT از یک شبکه عمیق برای سنجش قرابت بصری اشیاء استفاده می‌شود
- در برخی از مقالات بعد از DeepSORT فقط به استخراج توصیفگرهای ظاهری بهتر پرداخته شده است



۲۷

Wojke, Nicolai, Alex Bewley, and Dietrich Paulus. "Simple online and realtime tracking with a deep association metric." ICIP, 2017.

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{u}, \quad \mathbf{u} \sim \mathcal{N}(0, \mathbf{P})$$

$$\mathbf{y}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}, \quad \mathbf{v} \sim \mathcal{N}(0, \mathbf{Q})$$

Prediction Step

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{u}, \quad \mathbf{u} \sim \mathcal{N}(0, \mathbf{P})$$

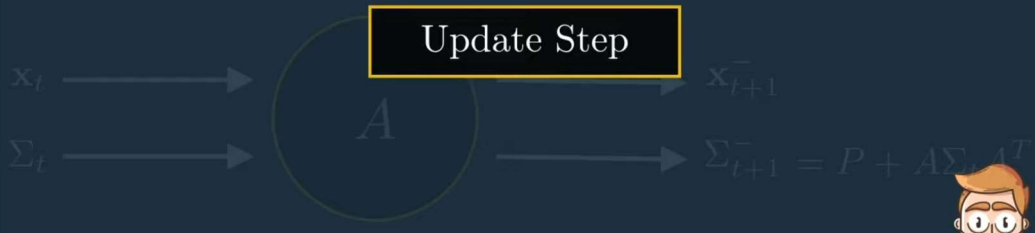
$$\mathbf{y}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}, \quad \mathbf{v} \sim \mathcal{N}(0, \mathbf{Q})$$

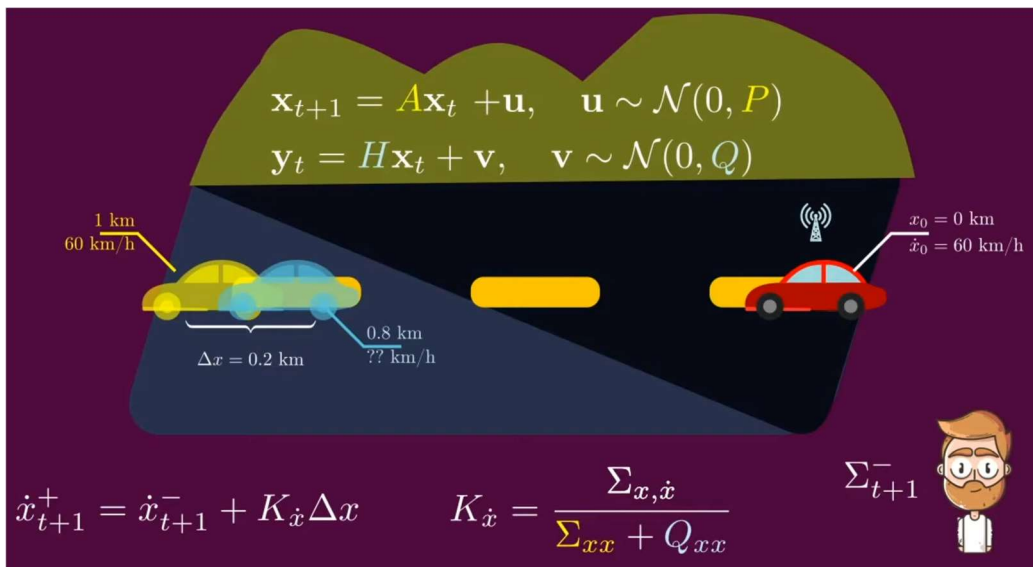
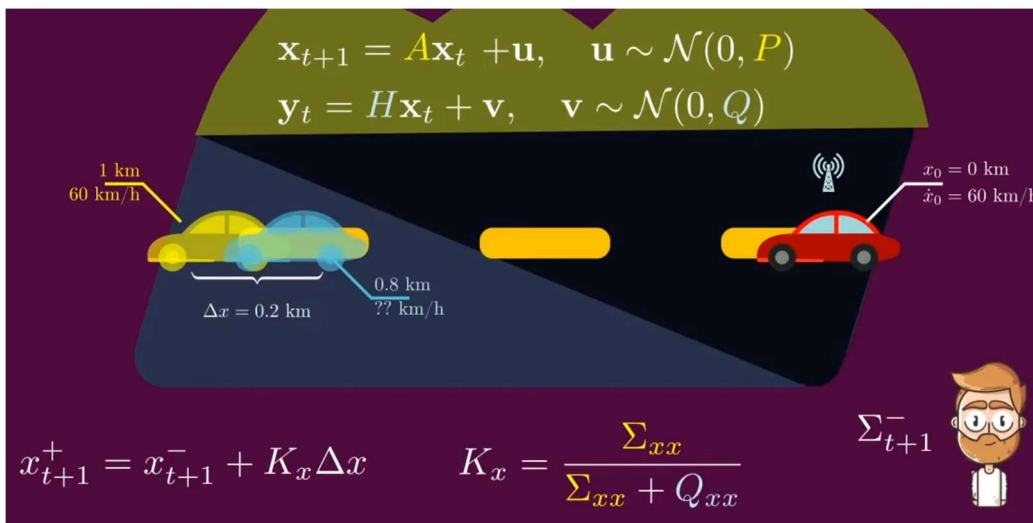
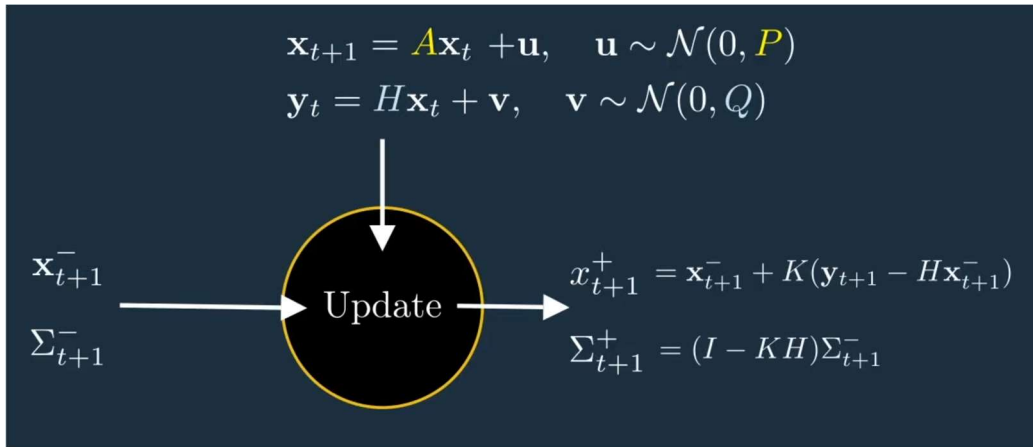


$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{u}, \quad \mathbf{u} \sim \mathcal{N}(0, \mathbf{P})$$

$$\mathbf{y}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}, \quad \mathbf{v} \sim \mathcal{N}(0, \mathbf{Q})$$

Update Step





Details [\[edit \]](#)

The Kalman filter is a [recursive](#) estimator. This means that only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state. In contrast to batch estimation techniques, no history of observations and/or estimates is required. In what follows, the notation $\hat{\mathbf{x}}_{n|m}$ represents the estimate of \mathbf{x} at time n given observations up to and including at time $m \leq n$.

The state of the filter is represented by two variables:

- $\mathbf{x}_{k|k}$, the [a posteriori](#) state estimate mean at time k given observations up to and including at time k ;
- $\mathbf{P}_{k|k}$, the [a posteriori](#) estimate covariance matrix (a measure of the estimated [accuracy](#) of the state estimate).

The algorithm structure of the Kalman filter resembles that of [Alpha beta filter](#). The Kalman filter can be written as a single equation; however, it is most often conceptualized as two distinct phases: "Predict" and "Update". The predict phase uses the state estimate from the previous timestep to produce an estimate of the state at the current timestep. This predicted state estimate is also known as the [a priori](#) state estimate because, although it is an estimate of the state at the current timestep, it does not include observation information from the current timestep. In the update phase, the [innovation](#) (the pre-fit residual), i.e. the difference between the current [a priori](#) prediction and the current observation information, is multiplied by the optimal Kalman gain and combined with the previous state estimate to refine the state estimate. This improved estimate based on the current observation is termed the [a posteriori](#) state estimate.

Typically, the two phases alternate, with the prediction advancing the state until the next scheduled observation, and the update incorporating the observation. However, this is not necessary; if an observation is unavailable for some reason, the update may be skipped and multiple prediction procedures performed. Likewise, if multiple independent observations are available at the same time, multiple update procedures may be performed (typically with different observation matrices \mathbf{H}_k).^{[29][30]}

Predict [\[edit \]](#)

Predicted (a priori) state estimate	$\hat{\mathbf{x}}_{k k-1} = \mathbf{F}_k \mathbf{x}_{k-1 k-1} + \mathbf{B}_k \mathbf{u}_k$
Predicted (a priori) estimate covariance	$\hat{\mathbf{P}}_{k k-1} = \mathbf{F}_k \mathbf{P}_{k-1 k-1} \mathbf{F}_k^\top + \mathbf{Q}_k$

Update [\[edit \]](#)

Innovation or measurement pre-fit residual	$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k k-1}$
Innovation (or pre-fit residual) covariance	$\mathbf{S}_k = \mathbf{H}_k \hat{\mathbf{P}}_{k k-1} \mathbf{H}_k^\top + \mathbf{R}_k$
Optimal Kalman gain	$\mathbf{K}_k = \hat{\mathbf{P}}_{k k-1} \mathbf{H}_k^\top \mathbf{S}_k^{-1}$
Updated (a posteriori) state estimate	$\mathbf{x}_{k k} = \hat{\mathbf{x}}_{k k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$
Updated (a posteriori) estimate covariance	$\mathbf{P}_{k k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \hat{\mathbf{P}}_{k k-1}$
Measurement post-fit residual	$\tilde{\mathbf{y}}_{k k} = \mathbf{z}_k - \mathbf{H}_k \mathbf{x}_{k k}$

The formula for the updated ([a posteriori](#)) estimate covariance above is valid for the optimal \mathbf{K}_k gain that minimizes the residual error, in which form it is most widely used in applications. Proof of the formulae is found in the [derivations](#) section, where the formula valid for any \mathbf{K}_k is also shown.

A more intuitive way to express the updated state estimate ($\hat{\mathbf{x}}_{k|k}$) is:

$$\mathbf{x}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \mathbf{z}_k$$

This expression reminds us of a linear interpolation, $x = (1 - t)(a) + t(b)$ for t between $[0, 1]$. In our case:

- t is the Kalman gain (\mathbf{K}_k), a matrix that takes values from 0 (high error in the sensor) to \mathbf{I} (low error).
- a is the value estimated from the model.
- b is the value from the measurement.

This expression also resembles the [alpha beta filter](#) update step.

Invariants [edit]

If the model is accurate, and the values for $\hat{\mathbf{x}}_{0|0}$ and $\mathbf{P}_{0|0}$ accurately reflect the distribution of the initial state values, then the following invariants are preserved:

$$\begin{aligned}\mathbf{E}[\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}] &= \mathbf{E}[\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}] = 0 \\ \mathbf{E}[\tilde{\mathbf{y}}_k] &= 0\end{aligned}$$

where $\mathbf{E}[\xi]$ is the [expected value](#) of ξ . That is, all estimates have a mean error of zero.

Also:

$$\begin{aligned}\mathbf{P}_{k|k} &= \text{cov}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}) \\ \mathbf{P}_{k|k-1} &= \text{cov}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}) \\ \mathbf{S}_k &= \text{cov}(\tilde{\mathbf{y}}_k)\end{aligned}$$

so covariance matrices accurately reflect the covariance of estimates.

Estimation of the noise covariances \mathbf{Q}_k and \mathbf{R}_k [edit]

Practical implementation of a Kalman Filter is often difficult due to the difficulty of getting a good estimate of the noise covariance matrices \mathbf{Q}_k and \mathbf{R}_k . Extensive research has been done to estimate these covariances from data. One practical method of doing this is the *autocovariance least-squares (ALS)* technique that uses the time-lagged [autocovariances](#) of routine operating data to estimate the covariances.^{[31][32]} The [GNU Octave](#) and [Matlab](#) code used to calculate the noise covariance matrices using the ALS technique is available online using the [GNU General Public License](#).^[33] Field Kalman Filter (FKF), a Bayesian algorithm, which allows simultaneous estimation of the state, parameters and noise covariance has been proposed.^[34] The FKF algorithm has a recursive formulation, good observed convergence, and relatively low complexity, thus suggesting that the FKF algorithm may possibly be a worthwhile alternative to the Autocovariance Least-Squares methods.

منابع:

https://en.wikipedia.org/wiki/Kalman_filter

<https://chat.openai.com/>

<https://medium.com/@smallfishbigsea/notes-on-tracking-algorithms-sort-and-deepsort-d2684ced502f>

<https://www.mdpi.com/2076-3417/12/3/1319#:~:text=The%20SORT%20method%20associates%20objects,the%20object%20bounding%20box%20images.>

(الف)

اعمال ریاضی فیلتر Kalman برای پیش‌بینی ۱ پارامتر:

فیلتر کالمن یک الگوریتم بازخوردی است که دو مرحله اصلی دارد: پیش‌بینی و به‌روزرسانی. فرض کنید که ما یک پارامتر که به‌طور مداوم در حال تغییر است را دنبال می‌کنیم. ما به دو متغیر نیاز داریم: حالت فعلی سیستم (x) و عدم قطعیت در برآورد حالت (P).

مرحله پیش‌بینی: در این مرحله، ما پیش‌بینی می‌کنیم که حالت بعدی سیستم چه خواهد بود. این کار با استفاده از مدل حرکت سیستم انجام می‌شود. برای مثال، می‌توانیم از یک مدل ساده مانند حرکت یکنواخت برای پیش‌بینی استفاده کنیم.

$$x' = Fx + Bu$$

$$P' = FP(F^T) + Q$$

که در اینجا F ماتریس تبدیل حالت، B ماتریس کنترل و u نیروی کنترل است. همچنین، Q نمایانگر عدم قطعیت در مدل حرکت است.

مرحله به‌روزرسانی: در این مرحله، ما با استفاده از اطلاعات جدید (مشاهدات) برآورد حالت سیستم را به‌روز می‌کنیم.

$$K = P'(H^T)(HP'(H^T) + R)^{-1}$$

$$x = x' + K(y - Hx')$$

$$P = (I - KH)P'$$

که در اینجا K ضریب کالمن، H ماتریس تبدیل مشاهده، R عدم قطعیت در مشاهدات و y مشاهده جدید است.

The Kalman filter is a recursive algorithm that estimates the state of a dynamic system from a series of noisy measurements. It also can be used for tracking an object's position, velocity, or any other single parameter. It consists of two steps: prediction and update. The prediction step uses a state transition model to project the current state estimate to the next time step, along with an associated uncertainty. The update step incorporates a new measurement into the state estimate by using a weighted average of the predicted state and the measurement, where the weights are determined by the relative uncertainty of each source. (The Kalman filter assumes a linear Gaussian model, where the system dynamics can be represented by linear equations, and the measurement noise and process noise are assumed to be Gaussian.)

For a one-parameter system, such as estimating the position of an object along a line, the Kalman filter can be simplified as follows:

Let x be the state variable (position), P be the state uncertainty (variance), F be the state transition model (constant velocity), Q be the process noise (variance), z be the measurement, R be the measurement noise (variance), and K be the Kalman gain.

The prediction step is:

$$\begin{aligned}x' &= Fx + w \\P' &= FPF^T + Q\end{aligned}$$

where w is a random variable with zero mean and variance Q that models the process noise.

The update step is:

$$\begin{aligned}y &= z - Hx \\S &= HP'H^T + R \\K &= P'H^TS^{-1} \\x &= x' + Ky \\P &= (I - KH)P'\end{aligned}$$

where y is the measurement residual, S is the residual covariance, H is the measurement model (identity matrix), and v is a random variable with zero mean and variance R that models the measurement noise.

The Kalman filter can be applied iteratively to a sequence of measurements to obtain an optimal estimate of the state and its uncertainty at each time step. The Kalman gain determines how much to trust the prediction versus the measurement, and it adapts to the changing noise levels. The Kalman filter is optimal for linear systems with Gaussian noise, but it can also work well for some nonlinear or non-Gaussian systems with appropriate modifications.

(ب)

فرض کنید حالت سیستم شامل ۴ پارامتر باشد: x و y (مختصات مرکز object) و w و h (عرض و ارتفاع object). ما می‌خواهیم این پارامترها را برای هر object در فریم‌های بعدی پیش‌بینی کنیم. متغیر حالت x یک بردار ۲۸ بعدی (4 features x 7 nodes) خواهد بود و عدم قطعیت حالت P (state uncertainty) یک ماتریس 28x28 خواهد بود. ماتریس مدل انتقال حالت F (state transition model) یک ماتریس 28x28 خواهد بود که چگونگی تکامل حالت را از یک مرحله زمانی به مرحله بعدی توضیح می‌دهد. نویز فرآیند Q (process noise) یک ماتریس 28x28 خواهد بود که عدم قطعیت در مدل انتقال حالت را توصیف می‌کند. مرحله پیش‌بینی بدین شکل خواهد بود:

$$\begin{aligned}x' &= Fx + w \\P' &= FPF^T + Q\end{aligned}$$

که در آن w یک متغیر تصادفی با میانگین صفر و ماتریس کواریانس Q است که نویز فرآیند (process noise) را مدل می کند.

اگر خیلی ساده بخواهیم توصیف کنیم:

$$x_{predicted} = F * x_{previous}$$

مرحله به روز رسانی یک اندازه گیری جدید z را شامل می شود که می تواند بردار به طول m باشد که m تعداد اندازه گیری های موجود در آن مرحله زمانی است. مدل اندازه گیری H (measurement model) یک ماتریس $mx28$ است که چگونگی ارتباط متغیر حالت x (state variable) با اندازه گیری z را توصیف می کند. نویز اندازه گیری R (measurement noise) یک ماتریس mxm است که عدم قطعیت در اندازه گیری ها را توصیف می کند. مرحله به روز رسانی به شکل زیر خواهد بود:

$$y = z - Hx$$

$$S = HP'H^T + R$$

$$K = P'H^TS^{-1}$$

$$x = x' + Ky$$

$$P = (I - KH)P'$$

که در آن y باقیمانده اندازه گیری (measurement residual)، S کواریانس باقیمانده (residual covariance)، و K بهره کالمن (Kalman gain) است.

اگر خیلی ساده بخواهیم توصیف کنیم:

$$x_{updated} = x_{predicted} + K * (z - H * x_{predicted})$$

همانند مورد یک پارامتری، فیلتر کالمن را می توان به طور مکرر در یک دنباله از اندازه گیری ها اعمال کرد تا تخمین بهینه ای از حالت و عدم قطعیت آن در هر مرحله زمانی بدست آید. بهره کالمن تعیین می کند که چقدر به پیش بینی در مقابل اندازه گیری اعتماد شود، و با سطوح نویز در حال تغییر سازگار می شود.

همچنین در زیر مثالی ساده از یک گره با چهار پارامتر آمده است:

برای این کار، می توانیم از یک مدل حرکت ساده مانند حرکت یکنواخت استفاده کنیم. در این مورد، ماتریس F به شکل زیر خواهد بود:

$$F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

این ماتریس باعث می‌شود که موقعیت x و y در هر گام زمانی با سرعت ثابت به‌روز شود.

(ج)

تفاوت در الگوریتم Deep SORT با الگوریتم SORT:

الگوریتم Deep SORT نسخه‌ای ارتقا یافته از الگوریتم SORT است. تفاوت اصلی این دو در نحوه متناظر کردن object ها در فریم‌های متوالی است. در الگوریتم SORT، این متناظر سازی بر اساس فاصله اقلیدسی بین پیش‌بینی‌های فیلتر کالمن، IoU و مشاهدات جدید انجام می‌شود.

در حالی که در الگوریتم Deep SORT، علاوه بر فاصله مکانی، یک ویژگی جدید به نام "نمایه مشترک" (Appearance Embedding) استفاده می‌شود. برای محاسبه این ویژگی، از یک شبکه عصبی عمیق (مانند CNN) استفاده می‌شود که هر object را به یک بردار ویژگی کوچکتر تبدیل می‌کند. سپس با مقایسه این بردارهای ویژگی، میزان شباهت بین object های مختلف در فریم‌های متوالی بررسی می‌شود.

استفاده از این نمایه مشترک باعث می‌شود تا الگوریتم Deep SORT در مواقعی که تغییرات شدیدی در حرکت یا شکل object ها وجود دارد، عملکرد بهتری نسبت به الگوریتم SORT داشته باشد.

The main difference between the Deep SORT and SORT algorithms is that Deep SORT uses a deep neural network to compute appearance descriptors for the detected objects. These appearance descriptors are used to compute the similarity between objects in consecutive frames, which helps to associate detections over time.

In the original SORT algorithm, the association of detections between frames is done using only the motion information provided by a Kalman filter. However, in some cases, motion information alone may not be sufficient to accurately associate detections, especially in crowded scenes where multiple objects may have similar motion patterns.

Deep SORT addresses this issue by incorporating appearance information into the data association step. The deep neural network is used to extract features from the detected objects, which are then used to compute a similarity metric between objects in consecutive frames. This similarity metric takes into account both the motion and appearance of the objects, which can improve the accuracy of the data association and reduce the number of identity switches.

In fact, The Deep SORT algorithm extends the SORT algorithm by incorporating a deep neural network for object detection and feature extraction. While the SORT algorithm focuses on object tracking using bounding box information, the Deep SORT algorithm adds the capability of deep feature representation. The deep network is used to extract discriminative features from the objects, enabling better association between detections in consecutive frames. These features help in addressing the challenges of appearance variations, occlusions, and complex

tracking scenarios. The Deep SORT algorithm combines the benefits of both deep learning-based object detection and the SORT algorithm's tracking and data association techniques, resulting in improved tracking performance.

In summary, the deep neural network in Deep SORT is used to compute appearance descriptors for detected objects, which are then used to improve the data association between frames.

AND ALSO:

Essentially, sort uses kalman filter for object tracking without using ego-motion information. The correction information is from associating objects in two adjacent frames. For example, A is detected within frame t , B is detected within frame $t+1$, and A and B are associated as the same object using some criteria like IOU (intersection over union). So the Kalman filter can use B's location in frame $t+1$ as a new measurement for A to update the states.

From the above process, you can see association plays a critical role. Not associated means lost. IOU is not a good criterion, as it lays a big burden on the initial states of kalman filter, especially when the objects are small.

DeepSort adds a pre-trained neural net to generate features for objects, so the association can be made based on feature similarity besides overlap. The cascaded matched mentioned in the paper means trying to associating objects in multiple previous frames. It can alleviate the occlusion issue and re-discover objects.

On the other hand, if there are lots of landmarks/objects in frames, a brute-force match to maximize the number of matched landmarks could work better in practice.

<https://towardsdatascience.com/understanding-domain-adaptation-5baa723ac71f>
<https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/>
<https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>
<https://chat.openai.com/>

(الف)

برای برچسب گذاری فریم های قفس و ایجاد مجموعه داده آموزشی، این مراحل را دنبال می کنیم:

۱. نرم افزار CVAT را از لینک ارائه شده (<https://www.cvat.ai>) دانلود و نصب می کنیم. میتوانیم از نسخه web app نیز استفاده کنیم.
۲. CVAT را باز می کنیم و یک task/project جدید ایجاد می کنیم.
۳. فریم های قفس را با آپلود در پلتفرم به CVAT وارد می کنیم.
۴. برچسب زدن frame ها را با کشیدن bounding boxes در اطراف پرندگان در قفس شروع می کنیم. برچسب مناسب را برای هر جعبه مرزی (به عنوان مثال، "bird") اختصاص می دهیم.
۵. پس از اینکه همه فریم ها را برچسب زدیم، برچسب ها را در قالب YOLOv5 استخراج می کنیم. CVAT گزینه ای برای استخراج فرمت های مختلف از جمله YOLOv5 فراهم می کند.
۶. فایل های صادر شده YOLOv5 را دانلود می کنیم.
۷. اکنون می توانیم از داده های برچسب زده شده برای آموزش مدل YOLOv5s استفاده کنیم. اندازه دسته (batch size) را در حین آموزش روی ۲ و تعداد دوره ها (epoch) را ۵۰ تنظیم می کنیم.

To train a YOLOv5 model, you will need to prepare your training data in the appropriate format. This typically involves organizing your images and annotations into separate directories and creating a YAML file that specifies the location of your training and validation data, as well as other training parameters such as the number of classes and the names of the classes.

Once your data is prepared, you can use the train.py script provided by the YOLOv5 repository to train your model. You will need to specify the path to your YAML file, as well as other training parameters such as the batch size and the number of epochs. The script will then automatically train your model using the specified parameters.

After training is complete, you can evaluate your model on your validation data to see how well it performs. You can also use the trained model to make predictions on new images by using the detect.py script provided by the YOLOv5 repository.

توضیح کد پیاده سازی شده در نوتبوک:

****** از لینک ها و مفاهیمی که در سوال ۱ یاد گرفتیم در این سوال نیز استفاده میکنیم.

****** فایل yml مورد استفاده در فایل زیر تمرین موجود است.

```
> 3 > ! dataset.yml
1  train: ../data/images/training/
2  val:  ../data/images/validation/
3
4  # number of classes
5  nc: 1
6
7  # class names
8  names: ['bird']
```

کد پیاده سازی شده در واقع یک برنامه است که برای آموزش مدل YOLOv5 بر روی داده‌های تصویری مورد استفاده قرار می‌گیرد. توضیحاتی که در کد آمده است را بطور خلاصه می‌توان به شرح زیر تفسیر کرد:

۱. ایمپورت کتابخانه‌های مورد نیاز:

- `numpy` برای کار با آرایه‌ها و محاسبات علمی

- `shutil` برای انجام عملیات فایل و پوشه

- `ZipFile` برای استخراج فایل‌های فشرده

- `matplotlib.pyplot` برای نمایش تصاویر و رسم نمودارها

- `os` برای انجام عملیات سیستمی

- `PIL.Image` برای کار با تصاویر

- `cv2` برای پردازش تصاویر

- `glob` برای پیدا کردن فایل‌ها با الگوی خاص

۲. Mount کردن گوگل درایو:

این بخش ارتباط مستقیم با گوگل درایو را برقرار می‌کند تا بتوانیم به فایل‌های موجود در آن دسترسی داشته باشیم.

۳. ساختن پوشه مورد نظر برای داده‌ها:

این بخش پوشه‌ای با نام `data_root` در مسیر `content/data/` می‌سازد تا در آن داده‌های مورد نیاز برای آموزش قرار بگیرند.

۴. خواندن لیبل‌ها و فیلم مورد نظر:

در این قسمت لیبیل‌های موجود در گوگل در ایو خوانده می‌شوند و همچنین فیلم مربوطه خوانده شده و فریم‌های آن جدا می‌شوند و سپس در محل مناسب بر اساس مدل YOLO قرار می‌گیرند.

۵. آماده‌سازی ساختار پوشه‌ها:

در این قسمت ساختار پوشه‌های مورد نیاز برای ذخیره تصاویر و لیبیل‌ها تعریف می‌شود و پوشه‌ها ساخته می‌شوند.

۶. کپی کردن داده‌ها:

در این بخش، فایل‌های تصویر و لیبیل‌ها به صورت تصادفی بر اساس درصدی که تعیین شده است، به پوشه‌های آموزش و اعتبارسنجی کپی می‌شوند.

۷. Clone کردن مخزن YOLOv5:

این بخش مخزن YOLOv5 را از طریق Git clone می‌کند تا بتوانیم از آن برای آموزش استفاده کنیم.

۸. نصب موارد مورد نیاز:

با استفاده از فرمان `!pip install -r requirements.txt` کتابخانه‌های مورد نیاز برای آموزش YOLOv5 نصب می‌شوند.

۹. تنظیمات و آموزش مدل:

با استفاده از دستور `python train.py` و با تنظیمات مشخص شده (مانند اندازه تصاویر، تعداد دسته‌ها، تعداد دوره‌ها، مسیر داده‌ها و وزن‌های مدل) مدل YOLOv5 آموزش داده می‌شود.

این کد از امکانات Google Colab برای استفاده از منابع سخت‌افزاری قدرتمند و امکانات موجود در آن مانند GPU بهره می‌برد.

(ب)

این سوال خوبی است که مفهوم تطبیق دامنه (domain adaptation) را در بر می‌گیرد. تطبیق دامنه فرآیند تطبیق یک مدل آموزش‌دیده در یک حوزه (مانند تصاویر پرندگان در قفس) با دامنه دیگر (مانند تصاویر پرندگان در طبیعت) است. این می‌تواند زمانی مفید باشد که دامنه هدف دارای داده‌های برچسب‌دار محدود یا بدون داده در دسترس باشد، اما دامنه منبع دارای داده‌های برچسب‌دار فراوان باشد.

با این حال، تطبیق دامنه همیشه آسان یا موثر نیست. ممکن است تفاوت‌های قابل توجهی بین دامنه منبع و هدف وجود داشته باشد که باعث می‌شود مدل در دامنه جدید عملکرد ضعیفی داشته باشد. برای مثال، ظاهر، ژست، مقیاس و پس‌زمینه پرندگان ممکن است بین دو حوزه بسیار متفاوت باشد که می‌تواند مدل را به اشتباه بیندازد و دقت آن را کاهش دهد.

بنابراین، برای پاسخ به سؤال، ممکن است نتوان یک مدل خوب را با آموزش آن بر روی مجموعه داده ای که پرندگان در قفس نیستند، ساخت، مگر اینکه برخی از تکنیک‌های تطبیق دامنه (domain adaptation techniques) به کار گرفته شوند. این تکنیک‌ها می‌تواند شامل موارد زیر باشد:

- داده افزایی (Data augmentation): این شامل اعمال تبدیل هایی مانند برش، چرخش، فلیپ کردن یا اضافه کردن نویز (cropping, flipping, rotating, or adding noise) به داده های منبع (source data) است تا متنوع تر و شبیه به داده های هدف (target data) شود.
- تراز کردن ویژگی (Feature alignment): این شامل یادگیری یک نمایش ویژگی مشترک برای هر دو حوزه (domain) است که اختلاف توزیع (distribution discrepancy) بین آنها را به حداقل می رساند. این را می توان با استفاده از روش هایی مانند یادگیری مخالف یا حداکثر اختلاف میانگین (adversarial learning or maximum mean discrepancy) انجام داد.
- سازگاری مدل (Model adaptation): این شامل تنظیم دقیق یا به روز رسانی (fine-tuning or updating) پارامترهای مدل بر روی مقدار کمی از داده های هدف برچسب دار یا بدون برچسب (small amount of labeled or unlabeled target data) است تا برای دامنه جدید مناسب تر شود. این کار را می توان با استفاده از روش هایی مانند خودآموزی یا آموزش مشترک (self-training or co-training) انجام داد.

به طور خلاصه، آموزش یک مدل بر روی یک مجموعه داده که در آن پرندگان در قفس نیستند ممکن است به مدل خوبی برای تشخیص پرندگان در قفس منجر نشود (may not result in a good model)، مگر اینکه برخی از تکنیک های انطباق دامنه (domain adaptation) برای پر کردن شکاف بین دو حوزه اعمال شوند (bridge the gap between the two domains).

It is not ideal to train a model for object detection using a dataset where the birds are not in cages. The reason is that the model needs to learn the specific visual characteristics and context of birds within a cage in order to accurately detect and localize them. Training the model on images without cages would not provide the necessary information about the unique appearance and spatial arrangement of birds within a cage. As a result, the model's performance may be poor when applied to the task of detecting birds specifically in a cage.

(ج)

یکی از راه حل های این مشکل استفاده از انتقال یادگیری (transfer learning) است که شامل گرفتن یک مدل از پیش آموزش دیده [pre-trained model] (مانند مدل آموزش داده شده بر روی مجموعه داده COCO) و تنظیم دقیق (fine-tuning) آن بر روی مجموعه داده کوچکتري از پرندگان در قفس است. این می تواند دانش آموخته شده توسط مدل از پیش آموزش دیده از مجموعه داده های بزرگ را به این مسئله منتقل کند و آن را با وظیفه خاص تشخیص پرندگان در قفس تطبیق دهد.

در اینجا مراحل وجود دارد که می توانید برای پیاده سازی این راه حل دنبال کنید:

1. Obtain a pre-trained object detection model: می توانید یک مدل از پیش آموزش دیده را دانلود کنید که بر روی یک مجموعه داده بزرگ مانند COCO آموزش دیده است. بسیاری از مدل های از قبل آموزش دیده به صورت آنلاین موجود هستند، مانند YOLOv5، SSD و Faster R-CNN.

۲. Prepare your data: مجموعه داده کوچکی از تصاویر پرندگان در قفس را جمع آوری کنید و آنها را با جعبه های محدود در اطراف پرندگان حاشیه نویسی کنید. برای این کار می توانید از ابزار حاشیه نویسی مانند CVAT استفاده کنید.

۳. Fine-tune the model: مدل از پیش آموزش دیده را بارگذاری کنید و آن را روی مجموعه داده پرندگان در قفس تنظیم کنید (fine-tune). این شامل به روز رسانی وزن های مدل با استفاده از پس انتشار (backpropagation) برای به حداقل رساندن اتلاف (loss) در مجموعه داده شما است. می توانید از نرخ یادگیری کمتر و دوره های کمتری (smaller learning rate and fewer epochs) نسبت به زمانی که از ابتدا آموزش می دهید استفاده کنید، زیرا مدل قبلاً تا حدی آموزش دیده است.

۴. Evaluate the model: پس از تنظیم دقیق، عملکرد مدل را روی مجموعه ای از پرندگان در قفس ارزیابی کنید تا ببینید چقدر خوب عمل می کند. برای اندازه گیری دقت مدل می توانید از معیارهایی مانند mean average precision (mAP) استفاده کنید.

با استفاده از انتقال یادگیری، می توانید از یک مجموعه داده بزرگ از قبل موجود (مانند COCO) برای آموزش مدلی استفاده کنید که بتواند پرندگان در قفس را به خوبی تشخیص دهد، حتی اگر فقط مقدار کمی داده برجسته گذاری شده برای این کار خاص داشته باشید.

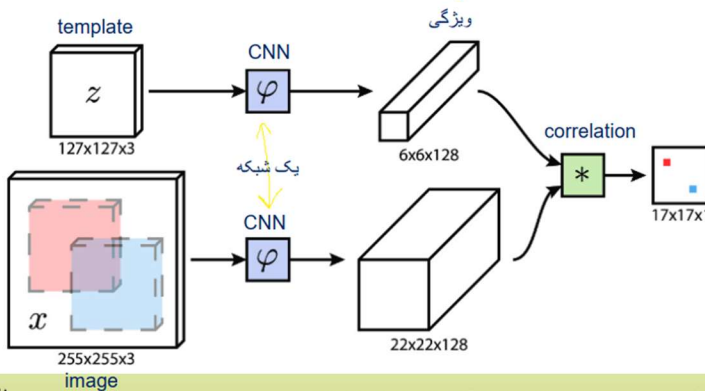
If it is difficult to provide a dataset with labeled images of birds in cages, one possible solution is to use a pre-existing dataset like COCO (Common Objects in Context) and fine-tune a pre-trained model on this dataset. Although COCO does not specifically focus on birds in cages, it contains a wide variety of object classes, including birds in natural contexts. By fine-tuning the model on COCO and then further training it on a smaller dataset containing images of birds in cages, the model can learn to recognize birds specifically in a cage environment. This approach leverages the pre-trained model's ability to extract generic visual features and then adapts it to the target domain of birds in cages, thus reducing the need for a large manually labeled dataset.

ابتدا خلاصه ای از مطالبی که در این موارد در اسلاید های درس موجود است آورده شده است:

شبیه تطبیق کلیشه
template matching

SiamFC

- برای مقایسه دقیق تر، می توان با استفاده از شبکه های عمیق بازنمایی معنایی سطح بالا از تصاویر استخراج کرد و سپس مقایسه را انجام داد



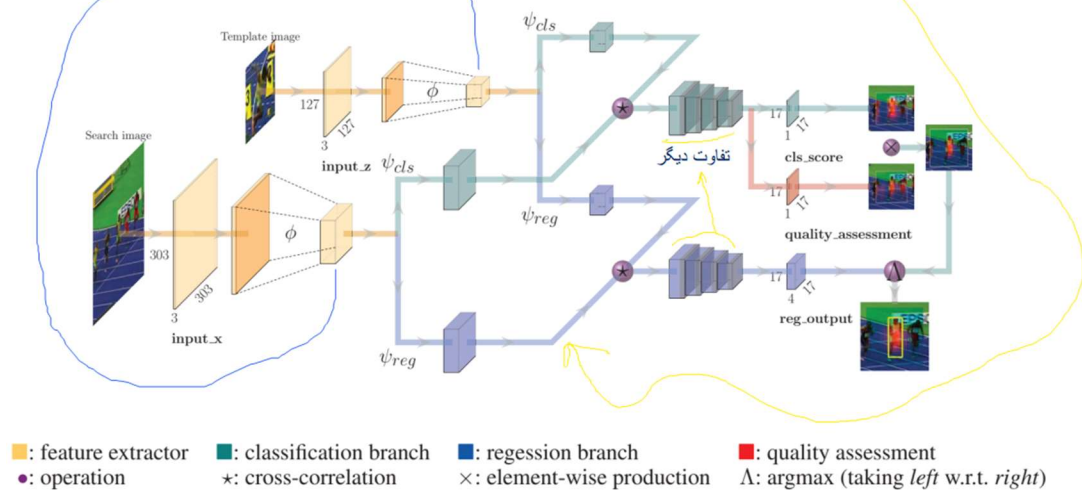
۲۱

Bertinetto, Luca, et al. "Fully-convolutional siamese networks for object tracking." *European conference on computer vision*. Springer, Cham, 2016.

بازنمایی معنایی سطح بالا از تصاویر استخراج می شود

تفاوت اصلی ایجاد شاخه مکان یابی است
regression(bounding box)

SiamFC++



۲۳

Xu, Yinda, et al. "Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 07. 2020.

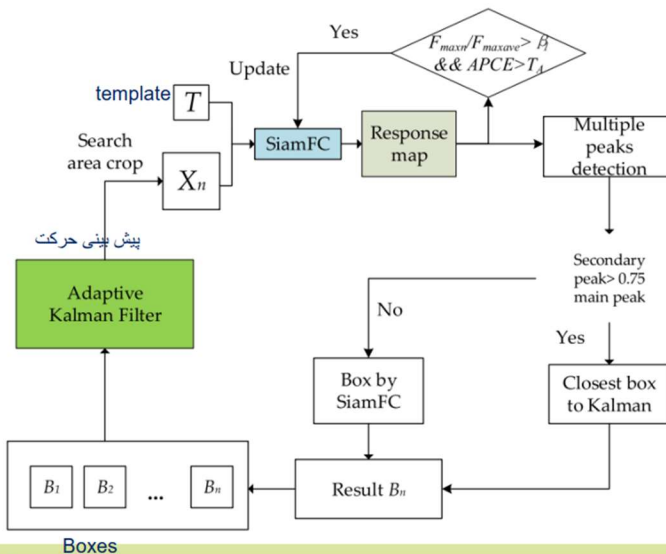
ردیابی با پیش‌بینی حرکت

- استفاده از پیش‌بینی حرکت، به خصوص در حضور اشیاء مشابه، می‌تواند کمک قابل توجهی به ردیابی نماید

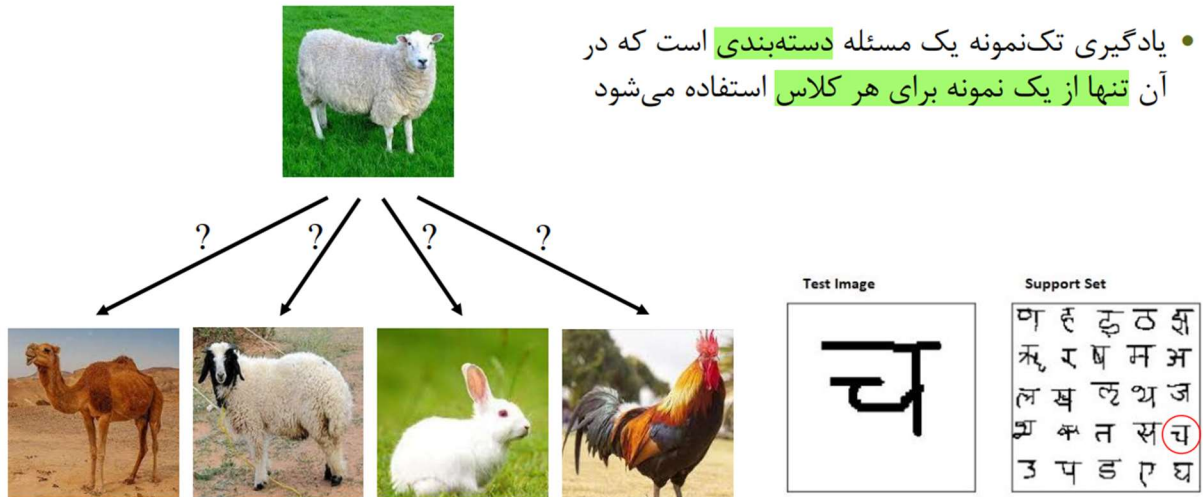


- می‌توان برای افزایش سرعت، حتی ناحیه جستجو را محدود کرد
- از جمله الگوریتم‌های پرکاربرد برای پیش‌بینی محل شیئی می‌توان به فیلتر Kalman اشاره کرد

ردیابی با پیش‌بینی حرکت



یادگیری تک‌نمونه (One Shot Learning)



۲۹

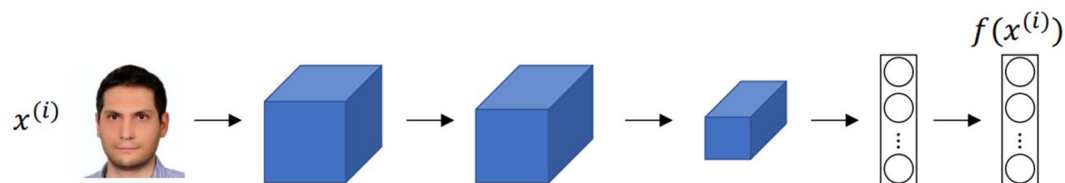
<https://towardsdatascience.com/one-shot-learning-with-siamese-networks-using-keras-17f34e75bb3d>

یادگیری تابع شباهت

- $d(img1, img2)$: درجه تفاوت بین دو تصویر

$$d(x^{(1)}, x^{(2)}) = \|f(x^{(1)}) - f(x^{(2)})\|_2^2$$

- پارامترهای شبکه آموزش می‌بینند تا:
 - اگر $x^{(i)}$ و $x^{(j)}$ مربوط به یک نفر باشند، $d(x^{(i)}, x^{(j)})$ عدد کوچکی باشد
 - اگر $x^{(i)}$ و $x^{(j)}$ مربوط به افراد متفاوتی باشند، $d(x^{(i)}, x^{(j)})$ عدد بزرگی باشد











از یک شبکه چند بار استفاده می کند

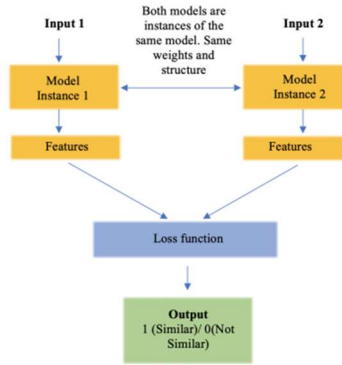
مناسب برای کاشی

شبکه Siamese

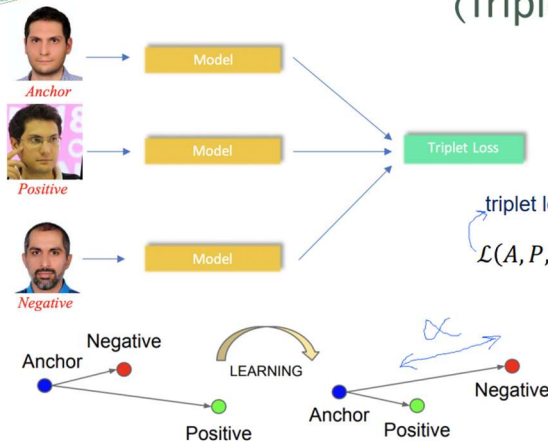
for training

$x^{(1)}$	$x^{(2)}$	y
		0
		1
		0
		1

- یک شبکه Siamese کلاسی از شبکه های عصبی است که شامل یک یا چند شبکه یکسان است



تابع ضرر سه تایی (Triplet Loss)



$$d(A, P) + \alpha \leq d(A, N)$$

$$d(A, P) - d(A, N) + \alpha \leq 0$$

$$\mathcal{L}(A, P, N) = \max(d(A, P) - d(A, N) + \alpha, 0)$$

$$J = \sum_{i=1}^M \mathcal{L}(A^{(i)}, P^{(i)}, N^{(i)})$$

اگر منفی باشد درست گفته و ضرری نداریم

۳۵

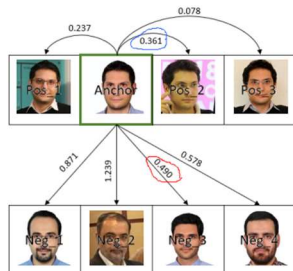
انتخاب سه تایی ها

- سه تایی هایی که برای آموزش مدل استفاده می شوند باید با دقت انتخاب شوند
- اگر به صورت تصادفی انتخاب شوند

- شرط $d(A, P) + \alpha \leq d(A, N)$ به راحتی برآورده می شود
- تابع ضرر کوچک خواهد بود و به روزرسانی مدل به کندی انجام خواهد شد

- سه تایی ها به صورت آنلاین و برحسب فاصله فعلی با یکدیگر تولید می شوند

- به آنها مثبت سخت (hard positive) و منفی سخت (hard negative) گفته می شود



SiamFC (Siamese Fully Convolutional) یک معماری شبکه است که برای کارهای ردیابی اشیاء بصری (visual object tracking tasks) طراحی شده است. این یک ساختار شبکه سیامی (Siamese) را معرفی می کند که امکان ردیابی کارآمد و دقیق اشیاء در فیلم ها را فراهم می کند.

SiamFC is a fully convolutional Siamese neural network that has achieved outstanding results in addressing a broad scope of computer vision problems, including object tracking. It models visual tracking as a similarity-learning problem. The SiamFC tracks the target by the similarity measurement method. It uses the similarity function $f(z, x)$ measuring the search area x and the template area z . The position with high similarity is given a high score in the response map, and the target position should be where the score is the highest in the response map.

Despite these improvements, fully convolutional Siamese neural networks still hardly adapt to complex scenes, such as appearance change, scale change, similar objects interference, etc. To improve the deficient tracking ability of fully-convolutional Siamese networks (SiamFC) in complex scenes, an object tracking framework with Siamese network and re-detection mechanism (Siam-RM) has been proposed. The mechanism adopts the Siamese instance search tracker (SINT) as the re-detection network.

The concept of Siamese architecture extends beyond object tracking and can benefit other computer vision tasks. For example, One-Shot Learning is one of the areas used in this network.

در زیر پاسخ سوالات آمده است:
(الف)

مزایا و محدودیت های SiamFC در وظایف بینایی کامپیوتر:

مزایا:

- SiamFC با به اشتراک گذاشتن ویژگی های کانولوشنی در فریم ها به عملکرد بلادرنگ (real-time performance) دست می یابد و نیاز به پردازش فریم به فریم را از بین می برد.

- نیازی به تنظیم دقیق آنلاین یا افزایش داده ندارد (online fine-tuning or data augmentation)، و از نظر محاسباتی کارآمد و آسان است.

- SiamFC به دلیل توانایی خود در یادگیری ویژگی های متمایز، در برابر تغییرات در مقیاس، چرخش و انسداد (variations in scale, rotation, and occlusion) استحکام نشان می دهد و قوی عمل میکند.

محدودیت ها:

- SiamFC ممکن است با تغییرات ظاهری شدید یا تغییر شکل اجسام (extreme appearance changes or deformations of object) دست و پنجه نرم کند زیرا به تطابق ظاهری متکی است.
- ممکن است در سناریوهایی با اشیاء مشابه یا پس زمینه‌های به هم ریخته با چالش‌هایی روبرو شود، جایی که تشخیص بین اشیاء دشوار می‌شود.

The SiamFC network architecture has several advantages in computer vision tasks. It is a **fully-convolutional Siamese network** trained end-to-end on the **ILSVRC15 dataset** for object detection in video. The tracker operates at frame-rates beyond real-time and, despite its extreme simplicity, achieves state-of-the-art performance in multiple benchmarks. This is because the online-only approach of traditional object tracking methods inherently limits the richness of the model they can learn. SiamFC overcomes this limitation by exploiting the expressive power of deep convolutional networks.

(ب)

عملکرد و اجزای اصلی SiamFC در ردیابی اشیاء:

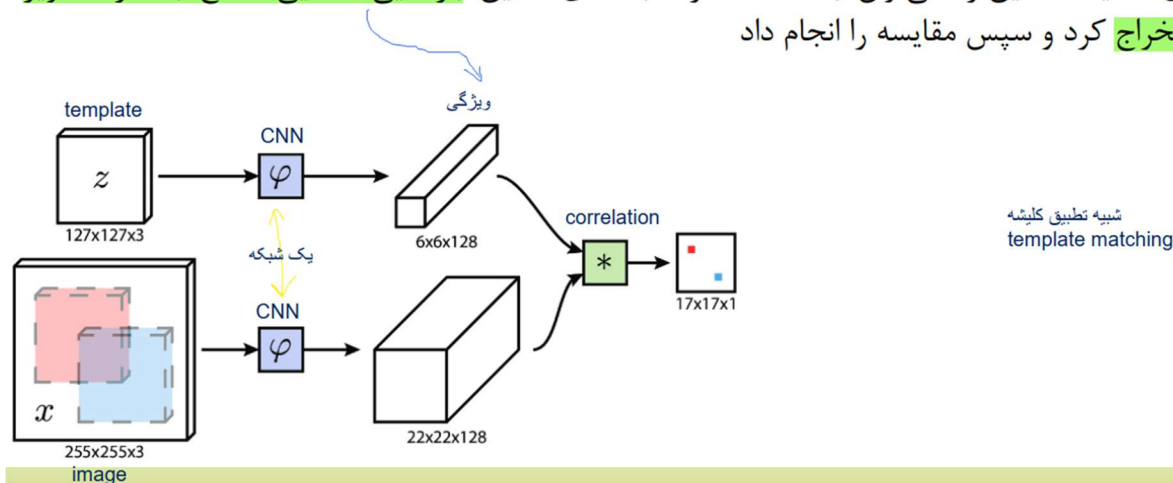
- SiamFC با یادگیری یک متریک شباهت بین شیء هدف و مناطق کاندید (target object and candidate regions) در فریم‌های بعدی کار می‌کند. اجزای اصلی عبارتند از:
 - شاخه الگو (Template Branch): ویژگی‌های یک تصویر الگو (template image) حاوی شیء مورد نظر را استخراج می‌کند.
 - شاخه جستجو (Search Branch): ویژگی‌هایی را از تصویر جستجو استخراج می‌کند، که فریم بعدی است که هدف قرار است در آن قرار گیرد.
 - همبستگی متقابل (Cross-correlation): همبستگی متقابل ویژگی‌های شاخه‌های الگو و جستجو برای اندازه‌گیری شباهت بین مناطق هدف و کاندید محاسبه می‌شود. (features from the template and search branches are cross-correlated to measure the similarity)
 - محلی‌سازی (Localization): بالاترین امتیاز شباهت نشان دهنده منطقه‌ای است که به احتمال زیاد شیء مورد نظر را در بر می‌گیرد.

SiamFC works by converting the visual tracking process into a **template matching problem**. A template and detection image are mapped into a newly embedded space by the same convolutional neural network (CNN), and they compare the similarity with each other by

correlation. SiamFC performs a multi-scale search on the detection image and determines the location and scale of the target based on those of the maximum score of the response map.

Siamese Fully Convolutional SiamFC

- برای مقایسه دقیق‌تر، می‌توان با استفاده از شبکه‌های عمیق بازنمایی معنایی سطح بالا از تصاویر استخراج کرد و سپس مقایسه را انجام داد



(ج)

چالش‌های ردیابی اشیاء

- اشیاء کوچک دارای ویژگی‌های دیداری اندک
- حرکت‌های بسیار سریع و نامنظم
- پس‌زمینه پیچیده
- انسداد، خروج از دید دوربین و بازگشت
- وجود اشیاء مشابه در همسایگی
- ردیابی همزمان چندین شیء متحرک
- ردیابی میان دوربین‌های مختلف



چالش‌های موجود در ردیابی اشیاء و نحوه برخورد SiamFC با آنها:

چالش‌های ردیابی اشیاء شامل تغییرات در ظاهر، تغییرات مقیاس، انسداد و حرکت سریع است. SiamFC با این چالش‌ها از طریق‌های زیر مقابله می‌کند:

- یادگیری ظاهری (Appearance Learning): ویژگی‌های ظاهری متمایز را می‌آموزد تا تغییرات ظاهری اشیاء را مدیریت کند و با تغییرات در مقیاس مقابله کند.

- منظم‌سازی فضایی (Spatial Regularization): SiamFC از یک اصطلاح منظم‌دهی فضایی برای توضیح حرکت اجسام و اطمینان از استحکام برای اجسام با حرکت سریع استفاده می‌کند.

- آموزش آنلاین (Online Learning): در حالی که SiamFC نیازی به تنظیم دقیق آنلاین ندارد (does not require online fine-tuning)، همچنان می‌تواند با به روز رسانی دوره ای الگو (updating the template periodically)، با تغییرات ظاهری در طول زمان سازگار شود.

Object tracking is a fundamental problem in computer vision and is widely used in unmanned vehicles, video surveillance, human-computer interaction, and so on. Given the position and scale of the target in the first frame, the tracker tracks the target and predicts its position and scale information in subsequent frames. During the tracking process, the tracker is often affected by occlusion, illumination changing, motion blur, and out of view making visual tracking a challenging problem.

SiamFC converts the visual tracking process into a template matching problem. A template and detection image are mapped into a newly embedded space by the same convolutional neural network (CNN), and they compare the similarity with each other by correlation. Thanks to CNN's powerful appearance representation capability and the simplicity of Siamese structure, SiamFC shows well tracking performance while reaching up to 120 frames per second (FPS).

(د)

گسترش معماری سیامی فراتر از ردیابی اشیاء:

معماری سیامی استفاده شده در SiamFC را می‌توان برای کارهای مختلف بینایی کامپیوتری فراتر از ردیابی اشیاء اعمال کرد. یک مثال آموزش تک شات (One-Shot Learning) است که هدف آن شناسایی اشیاء جدید از یک مثال آموزشی (a single training example) است. با استفاده از شبکه‌های سیامی، مدل می‌تواند متریک شباهت بین تصویر پرس و جو و تصاویر مرجع اشیاء شناخته شده (similarity metric between the query image and the reference images of known objects) را بیاموزد، که امکان تشخیص دقیق با داده‌های آموزشی محدود را فراهم می‌کند. شبکه‌های سیامی همچنین می‌توانند برای کارهایی مانند بازیابی تصویر، تطبیق تصویر، و تشخیص چهره (image retrieval, image matching, and face recognition) استفاده شوند، جایی که مقایسه‌های مبتنی بر شباهت مورد نیاز است.

The concept of Siamese architecture extends beyond object tracking and can be used in other computer vision tasks that require comparing two inputs. For example, Siamese networks have been used for **one-shot learning**, where the goal is to learn to recognize new classes based on only one or very few examples. In this case, the Siamese network is trained to learn a similarity

metric between pairs of images, so that it can determine whether two images belong to the same class or not.

Siamese networks have also been used for **face verification**, where the goal is to determine whether two face images belong to the same person or not. In this case, the Siamese network is trained to learn a similarity metric between pairs of face images.

منابع:

- (1) Enhancement: SiamFC Tracker Algorithm Performance Based on ... - MDPI.
<https://www.mdpi.com/2227-7390/10/9/1527>
- (2) Visual Tracking Method Based on Siamese Network with Multi ... - Springer.
<https://link.springer.com/article/10.3103/S0146411622020080>
- (3) Object tracking framework with Siamese network and re-detection
<https://jwcn-urasipjournals.springeropen.com/articles/10.1186/s13638-019-1579-x>
- (4) Improved SiamFC Target Tracking Algorithm Based on Anti ... - Hindawi.
<https://www.hindawi.com/journals/js/2022/2804114/>
- (5) Sensors | Free Full-Text | HKSiamFC: Visual-Tracking Framework ... - MDPI.
<https://www.mdpi.com/1424-8220/20/7/2137>
- (6) Ensemble siamese networks for object tracking | SpringerLink.
<https://link.springer.com/article/10.1007/s00521-022-06911-4>
- (7) Fully Convolutional Single-Crop Siamese Networks for Real-Time Visual
<https://www.mdpi.com/2079-9292/8/10/1084>
- (8) Siamese anchor-free object tracking with multiscale spatial ... - Nature.
<https://www.nature.com/articles/s41598-021-02095-4>
- (9) Spiking SiamFC++: Deep Spiking Neural Network for Object Tracking.
<https://arxiv.org/pdf/2209.12010>
- (10) Combining Siamese Network and Correlation Filter for ... - Springer.
https://link.springer.com/chapter/10.1007/978-3-030-79457-6_13

(11) SiamOAN: Siamese object-aware network for real-time target tracking.
<https://www.sciencedirect.com/science/article/pii/S0925231221016696>

(12) ChatGPT

<https://chat.openai.com/>

پایان