DAFTAR PUSTKA

Riansyah, R. Mardiati, M. R. Effendi and N. Ismail, "Fish Feeding Automation and Aquaponics Monitoring System Base on *IoT*," 2020 6th International Conference on Wireless and Telematics (ICWT), 2020, pp. 1-4, doi: 10.1109/ICWT50448.2020.9243620.

Novianto P, Zahir Z dan M. Nizwar, "Sistem Estimasi Berat Kepiting Bakau Menggunakan Pengolahan Citra Digital." 2018 Universitas Hasanuddin.

Aisuwarya, R. and Suhendra, E.F., 2018, October. Development of Automatic Fish Feeding System based on Gasping Behavior. In 2018 International Conference on Information Technology Systems and Innovation (ICITSI) (pp. 470-473). IEEE.

Akila, I.S., Karthikeyan, P., Hari, H.M. and Hari, K.J., 2018, March. *IoT* Based Domestic Fish Feeder. In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 1306-1311). IEEE.

Altybayev, A., Naydenko, Y., Meskhi, B., Mozgovoy, A., Rudoy, D. and Olshevskaya, A., 2020. Creation of integrated system for feeding management activities automation in beef breeding. In E3S Web of Conferences (Vol. 175, p. 03019). EDP Sciences.

Burange, A.W. and Misalkar, H.D., 2015, March. Review of Internet of Things in development of smart cities with data management & privacy. In 2015 International Conference on Advances in Computer Engineering and Applications (pp. 189-195). IEEE.

Hasim, H.N.B., Ramalingam, M., Ernawan, F. and Puviarasi, R., 2017, February. Developing fish feeder system using Raspberry Pi. In 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB) (pp. 246-250). IEEE.

Harani, N.H., Sadiah, A.S. and Nurbasari, A., 2019, April. Smart Fish Feeder Using Arduino Uno With Fuzzy Logic Controller. In 2019 5th International Conference on Computing Engineering and Design (ICCED) (pp. 1-6). IEEE.

Huang, J., Kuang, S.R., Chang, Y.N., Hung, C.C., Tsai, C.R. and Feng, K.L., 2019, October. A*IoT*s for Smart Shrimp Farming. In 2019 International SoC Design Conference (ISOCC) (pp. 17-18). IEEE.

Khalajzadeh, H., Mansouri, M. and Teshnehlab, M., 2014. Face recognition using convolutional neural network and simple logistic classifier. In Soft Computing in Industrial Applications (pp. 197-207). Springer, Cham.

Latif, N. and Astuti, A., 2019. Pengolahan Citra Digital Untuk Menentukan Bobot Sapi Menggunakan Metode Canny Edge Detection. Jurnal Ilmiah Ilmu Komputer Fakultas Ilmu Komputer Universitas Al Asyariah Mandar, 5(1), pp.1-6.

Mutiu A. Adegboye, Abiodun M. Aibinu, Jonathan G. Kolo, Aliyu Ibrahim, Aliha A, Folorunso, Sun-Ho Lee "Incorporating Intelligence in Fish Feeding System for Dispensing Feed based on Fish Feeding Intensity" IEE Access, vol. 7, 2020.

Makalalag, R.A., Lumenta, A.S., Sompie, S.R. and Sugiarso, B.A., 2012. Perancangan sistem pemantau dan penentuan tempat parkir berdasarkan digital image processing. Jurnal Teknik Elektro dan Komputer, 1(3).

Novianto Padaunan. Sistem Estimasi Berat Kepiting Bakau Menggunakan Pengolahan Citra Digital / Novianto Padaunan .2018.

Novianda, N., Fitria, L., Ihsan, A. and Munawir, M., 2019. Sistem Cerdas Pemberian Pakan Otomatis Dalam Peningkatan Produktivitas Panen Udang. *JURUTERA-Jurnal Umum Teknik Terapan*, *6*(02), pp.19-22.

Perdhana, D.N.K.C., 2020. Studi Pola Operasi Irigasi Tambak Komoditas Udang Vannamei di Kota Probolinggo (Doctoral dissertation, Universitas Brawijaya).

Reis, J., Weldon, A., Ito, P., Stites, W., Rhodes, M. and Davis, D.A., 2021. Automated feeding systems for shrimp: Effects of feeding schedules and passive feedback feeding systems. Aquaculture, 541, p.736800.

Syafaat, Muhammad Nu Abdul Mansyur, Syarifuddin Tonnek, and Muhammad Chaidir Undu. "Persentase Sisa Pakan Protein Tinggi dan Rendah Ddi Anco (Feeding Tray) Pada Budidaya Udang Vaname (Litopenaeus Vannamei) Intensif Dengan Teknik Pergiliran Pakan." Prosiding Forum Inovasi Teknologi Akuakultur. Balai Penelitian dan Pengembangan Budidaya Air Payau. 2016.

Tahe, S. and Suwoyo, H.S., 2011. Pertumbuhan dan sintasan udang vaname (Litopenaeus vannamei) dengan kombinasi pakan berbeda dalam wadah terkontrol. Jurnal Riset Akuakultur, 6(1), pp.31-40.

Montgomery, D.C., Peck, E.A., and Vining, G.G., 2012. Introduction to Linear Regression Analysis. 5th ed. Hoboken, NJ: John Wiley & Sons, Inc.

Weisberg, S., 2014. Applied Linear Regression. 4th ed. Hoboken, NJ: John Wiley & Sons, Inc.

Taufiq. Deteksi Rasa Kantuk pada Pengendara Kendaraan Bermotor Berbasis Pengolahan Citra Digital. Jurnal Teknologi Terapan and Sains 4.0, 2021, 2.1.

Uddin, M.N., Rashid, M.M. and Mostafa, M.G., 2016. Development of automatic fish feeder. Global Journal of Research In Engineering.

Wicaksono, M., Wibawa, I.P.D. and Jati, A.N., 2020. Implementasi Kontrol Posisi Pada Sistem Pemberi Pakan Udang Otomatis Dengan Metode Perencanaan Jalur. eProceedings of Engineering, 7(1).

Wyban, J. and Sweeney, J.N., 1991. Intensive shrimp production technology: the Oceanic Institute shrimp manual. The Institute.

DAFTAR LAMPIRAN

DATA PELATIHAN DAN PENGUJIAN

Data Perbandingan Berat Manual dan Sistem

No.	Jumlah Piksel	Berat Aktual (gram)	Linear (gram)	Polinomial (gram)
1	6678	8.450	7.214	7.486
2	6693	8.722	7.238	7.506
3	6834	8.262	7.460	7.698
4	7057	9.136	7.811	8.001
5	7171	8.453	7.990	8.157
6	7205	9.061	8.043	8.203
7	7421	8.903	8.383	8.499
8	7558	9.164	8.599	8.687
9	7565	9.123	8.610	8.696
10	7909	9.222	9.151	9.171
11	8290	9.065	9.750	9.699
12	8340	9.204	9.829	9.768
13	8443	8.883	9.793	9.913
14	8481	9.114	9.842	9.965
15	8484	9.136	9.845	9.980
16	8596	10.583	10.232	10.125
17	8696	8.912	10.144	10.278
18	8745	9.070	10.219	10.337
19	8750	8.855	10.225	10.340
20	8945	9.112	10.494	10.611
21	9012	12.250	10.592	10.719
22	9085	10.939	10.693	10.817
23	9198	12.061	10.855	10.970

24	9308	10.777	11.015	11.136
25	9318	11.239	11.028	11.148
26	9426	12.025	11.185	11.295
27	9433	12.133	11.186	11.303
28	9550	12.032	11.361	11.471
29	9729	12.188	11.613	11.711
30	9821	11.976	11.752	11.854
31	9823	12.083	11.757	11.863
32	9832	12.076	11.766	11.879
33	9865	12.089	11.818	11.926
34	9908	11.193	11.870	11.980
35	9948	12.050	11.935	12.043
36	9951	12.155	11.939	12.040
37	9964	11.977	12.384	12.058
38	10076	12.000	12.124	12.224
39	10285	12.220	12.452	12.526
40	10606	12.174	12.961	12.989
41	10609	12.067	12.961	12.980
42	10701	12.127	13.100	13.127
43	10974	12.377	13.517	13.518
44	11334	12.104	14.081	14.048
46	11641	16.776	14.552	14.489
47	12605	16.596	16.188	15.912
48	12853	17.453	16.405	16.284
49	12931	16.870	16.526	16.318
50	12942	17.117	16.547	16.417
51	13049	16.900	16.713	16.572
52	13147	17.193	16.864	16.727
53	13248	18.140	17.030	16.870
54	13290	16.938	17.100	16.937

55	13309	17.482	17.136	16.963
56	13783	17.367	17.935	17.673
57	13796	16.840	17.967	17.697
58	13927	17.534	18.145	17.899
59	14670	18.514	19.157	19.036
60	15541	20.000	20.510	20.370

Data Perbandingan Perhitungan Sistem dan Manual

No.	Jumlah (Manual)	Jumlah (Sistem)
1	12	12
2	13	13
3	18	18
4	28	28
5	30	30
6	39	36
7	38	38
8	37	35
9	36	35
10	35	35
11	34	33
12	33	33
13	32	32
14	31	31
15	30	30
16	29	29
17	28	28
18	27	27
19	26	26
20	25	25

21	24	24
22	23	23
23	22	22
24	21	21
25	20	20
26	52	48
27	49	45
28	44	42
29	41	40
30	39	38
31	37	36
32	35	35
33	33	33
34	31	30
35	30	30
36	28	28
37	27	27
38	25	25
39	23	23
40	20	20
41	17	17
42	15	15
43	13	13
44	10	10
45	32	31
46	30	30
47	29	29
48	26	26
49	24	24
50	23	23

51	21	21
52	19	19
53	18	18
54	15	15
55	14	14
56	11	11
57	10	10
58	9	9
59	7	7
60	5	5

Dataset Berat Berbanding Jumlah Piksel

nilai_piksel	berat
1175	1
2363	2
3451	3
4239	4
5067	5
5743	6
6446	7
7160	8
7872	9
8597	10
9302	11
9998	12
10633	13
11285	14
11934	15
12601	16
13233	17

13822	18
14578	19
15209	20
15857	21
16469	22
17068	23
17575	24
18225	25
18904	26
19601	27
20292	28
20942	29
21532	30
22199	31
22821	32
23401	33
24098	34
24637	35
25201	36
25710	37
26209	38
26712	39
27298	40
27721	41
28173	42
28621	43
29202	44
29793	45
30421	46

KODINGAN SISTEM

Estimasi Berat

```
1   import tkinter as tk
2   from tkinter import messagebox
3   # from tkinter import filedialog  # Hapus import filedialog
4   from PIL import Image, ImageTk
5   import numpy as np
6   import pandas as pd
7   # from scipy.interpolate import interp1d  # Hapus import interp1d 8
9   # Baca data dari file Excel
10  file_path = 'beratxpixel.xlsx' # Ganti dengan path file Excel Anda 11   df =
    pd.read_excel(file_path)
12
13  # Pisahkan variabel independen dan dependen
14  X = df['nilai_piksel']
15  y = df['berat']
16
17  # Buat fungsi interpolasi polinomial
18  polynomial_degree = 2 # Ganti dengan derajat polinomial yang diinginkan
19  coefficients = np.polyfit(X, y, polynomial_degree)
20  polynomial_interpolation = np.poly1d(coefficients)
21
22  # Fungsi untuk menghitung prediksi
23  # Fungsi untuk menghitung prediksi 24   def
    calculate_prediction(): try:
                    new_pixel_value =
                    float(pixel_entry.get())
                    # Menggunakan
                    persamaan linear
                    khusus
                    linear_prediction = 0.00157315 * new_pixel_value - 3.291094957
                    polynomial_prediction = polynomial_interpolation(new_pixel_value)

                    linear_result_label.config(text=f"{linear_prediction:.3f} gram") # Format untuk
                    tiga angka di belakang koma
                    polynomial_result_label.config(text=f"{polynomial_prediction:.3f} gram") #
                    Format untuk tiga angka di
            belakang koma except
            ValueError:
                    messagebox.showerror("Error", "Masukkan nilai piksel yang valid
(numerik)") 35   # Membuat GUI
36  root = tk.Tk()
37  root.title("Estimasi Berat Udang")
38
39  # Label untuk menampilkan gambar (Opsional, bisa dihapus jika tidak diperlukan)
40  image_label = tk.Label(root)
41  image_label.grid(row=1, column=0, columnspan=2, padx=5, pady=5) 42
```

```
# Label dan Entry untuk input nilai piksel
pixel_label = tk.Label(root, text="Jumlah Piksel:")
pixel_label.grid(row=2, column=0, padx=5, pady=5)

pixel_entry = tk.Entry(root)
pixel_entry.grid(row=2, column=1, padx=5, pady=5)

# Tombol untuk menghitung prediksi
calculate_button = tk.Button(root, text="Hitung Prediksi", command=calculate_prediction)
calculate_button.grid(row=3, column=0, columnspan=2, padx=5, pady=5)
# Tabel untuk menampilkan hasil prediksi
results_frame = tk.Frame(root)
results_frame.grid(row=4, column=0, columnspan=2, padx=5, pady=5)
linear_label = tk.Label(results_frame, text="Metode Linear")
linear_label.grid(row=0, column=0, padx=5, pady=5)

polynomial_label = tk.Label(results_frame, text="Metode
Polynomial")
polynomial_label.grid(row=0, column=1, padx=5,
pady=5)
linear_result_label = tk.Label(results_frame, text="-", font=("Helvetica", 14))
linear_result_label.grid(row=1, column=0, padx=5, pady=5)
```

lt_label = tk.Label(results_frame, text="-", font=("Helvetica", 14)) 68 polynomial_result_label.grid(row=1, column=1,

Deteksi Pakan

```
1  import cv2
2  import numpy as np
3  from datetime import datetime
4
5      def color_based_segmentation(image, lower_color, upper_color):
6      # Konversi gambar ke ruang warna HSV
7      hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV) 8
9       # Buat mask berdasarkan range warna yang diinginkan
10     mask = cv2.inRange(hsv_image, lower_color, upper_color)
11
12     # Aplikasikan mask ke gambar asli untuk mendapatkan hasil segmentasi
13     segmented_image = cv2.bitwise_and(image, image, mask=mask) 14
15         return segmented_image
16
17
18     def binary_image(image, threshold_value):
19     # Konversi gambar ke grayscale
20     gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) 21
22     # Binerisasi gambar dengan metode Otsu
23     _, binary_image = cv2.threshold(gray_image, threshold_value, 255, cv2.THRESH_BINARY)
24
25         return binary_image
26
27     if __name__ == "__main__":
28     # Load gambar
29     image = cv2.imread("qaz.jpg")
30
31     # Perkecil resolusi gambar menjadi 50%
32     width = int(image.shape[1] * 0.5)
33     height = int(image.shape[0] * 0.5)
34     resized_image = cv2.resize(image, (width, height), interpolation=cv2.INTER_AREA) 35
36     # Definisikan range warna yang ingin disegmentasi (dalam format HSV)
37     lower_color = np.array([4, 12, 20]) # Ganti dengan nilai minimum H, S, dan V yang sesuai
```

```
= np.array([50, 100, 200]) # Ganti dengan nilai maksimum H, S, dan V yang sesuai

gmentasi berbasis warna segmented_image = color_based_segmentation(resized_image, lower_color, upper_color)

oses binerisasi gambar dan hitung jumlah piksel hasil binerisasi threshold_value = 100 # Ganti dengan nilai
threshold yang sesuai binary_result = binary_image(segmented_image, threshold_value)

    # Hitung jumlah piksel hasil binerisasi yang bernilai 255 (putih) num_white_pixels = np.sum(binary_result == 255)

    # Gabungkan gambar asli, hasil segmentasi, dan hasil binerisasi secara horizontal combined_image = np.hstack([resized_image,
    segmented_image, cv2.cvtColor( binary_result, cv2.COLOR_GRAY2BGR)])

    # Tambahkan teks jumlah piksel pada gambar
    cv2.putText(combined_image, f"Num White Pixels: {num_white_pixels}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255,
            0), 2)

    # Dapatkan waktu saat ini untuk nama gambar current_time =
    datetime.now().strftime("%Y%m%d%H%M%S")

    # Nama file gambar yang akan disimpan file_name =
    f"combined_image_{current_time}.jpg"

    # Tampilkan gambar gabungan dalam satu jendela
    cv2.imshow("Original, Segmented, Binary (50% Resized)", combined_image)
    65
      66  # Tunggu input dari pengguna (tekan 's' untuk menyimpan)
      67  key = cv2.waitKey(0)
    68
    # Jika tombol 's' ditekan, simpan gambar 70          if key ==
ord('s'):
            71  # Simpan gambar dengan ukuran tampilan jendela yang sesuai
            72  window_size = cv2.getWindowImageRect("Original, Segmented, Binary (50% Resized)")
            73  window_width, window_height = window_size[2], window_size[3]
            74  cv2.imwrite(file_name, combined_image[0:window_height, 0:window_width]) 75
76              cv2.destroyAllWindows()
```

Matrik Evaluasi

```
1  import pandas as pd
2  from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score 3  import numpy as np
4
5  # Fungsi untuk menghitung MAE, MSE, RMSE, dan R2 6  def evaluate_metrics(actual,
predicted):
        mae = mean_absolute_error(actual, predicted) mse = mean_squared_error(actual,
        predicted) rmse = np.sqrt(mse) r2 = r2_score(actual, predicted) return mae, mse,
        rmse, r2

                    13  # Load data 14  file_path = 'X2.xlsx' # Ganti dengan path file Anda

15 data = pd.read_excel(file_path)
16

                    17  # Ekstrak nilai aktual dan prediksi

18 actual = data['Berat Aktual (gram)']
```

```
tions = data['Linear (gram)']
predictions = data['Polinomial (gram)']

tions = data['Splin (gram)'] 22  23  # Evaluasi metrik untuk setiap
                                     metode prediksi

24  linear_metrics = evaluate_metrics(actual, linear_predictions)
25  polynomial_metrics = evaluate_metrics(actual, polynomial_predictions)
26  # spline_metrics = evaluate_metrics(actual, spline_predictions)
27

                                28  # Gabungkan hasil dalam DataFrame untuk perbandingan

                                    29  metrics_df = pd.DataFrame([linear_metrics, polynomial_metrics],
                                    30  columns=['MAE', 'MSE', 'RMSE', 'R2'],
                                    31  index=['Linear', 'Polynomial'])
32
33  print(metrics_df)
```

etectron2 for Custom Instance Segmentation

Detectron2 repository by Facebook. This notebook shows training on **your own custom instance segmentation**

ist

We recommend that you follow along in this notebook while reading the blog post on how to train Detectron2 for custom instance segmentation, concurrently.

Steps Covered in this Tutorial

In this tutorial, we will walk through the steps required to train Detectron2 on your custom instance segmentation objects. We use a public American Sign Language instance segmentation dataset, which is open source and free to use. You can also use this notebook on your own data.

To train our segmenter we take the following steps:

- Install Detectron2 dependencies
- Download custom instance segmentation data from Roboflow
- Visualize Detectron2 training data
- Write our Detectron2 Training configuration
- Run Detectron2 training
- Evaluate Dectron2 performance
- Run Detectron2 inference on test images
- Export saved Detectron2 weights for future inference

About

Roboflow enables teams to deploy custom computer vision models quickly and accurately. Convert data from to annotation format, assess dataset health, preprocess, augment, and more. It's free for your first 1000 source images.

Looking for a vision model available via API without hassle? Try Roboflow Train.

roboflow

```
[ ] !python -m pip install roboflow
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.8/58.8 kB 3.7 MB/s eta 0:00:00
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.4.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from roboflow) (3.7.1)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.25.2)
Collecting opencv-python-headless==4.8.0.74 (from roboflow)
  Downloading opencv_python_headless-4.8.0.74-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (49.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.1/49.1 MB 16.5 MB/s eta 0:00:00
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from roboflow) (9.4.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.8.2)
Collecting python-dotenv (from roboflow)
  Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.31.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.16.0)
Collecting supervision (from roboflow)
  Downloading supervision-0.18.0-py3-none-any.whl (86 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.7/86.7 kB 13.0 MB/s eta 0:00:00
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.0.7)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.66.2)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (6.0.1)
Collecting requests-toolbelt (from roboflow)
  Downloading requests_toolbelt-1.0.0-py2.py3-none-any.whl (54 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.5/54.5 kB 7.9 MB/s eta 0:00:00
Collecting python-magic (from roboflow)
  Downloading python_magic-0.4.27-py2.py3-none-any.whl (13 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (1.2.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (4.48.1)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (23.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (3.1.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->roboflow) (3.3.2)
Requirement already satisfied: defusedxml<0.8.0,>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from supervision->roboflow) (0.7.1)
Requirement already satisfied: scipy<2.0.0,>=1.10.0 in /usr/local/lib/python3.10/dist-packages (from supervision->roboflow) (1.11.4)
Installing collected packages: python-magic, python-dotenv, opencv-python-headless, idna, cycler, chardet, certifi, supervision, requests-toolbelt, roboflow
  Attempting uninstall: opencv-python-headless
    Found existing installation: opencv-python-headless 4.9.0.80
    Uninstalling opencv-python-headless-4.9.0.80:
      Successfully uninstalled opencv-python-headless-4.9.0.80
  Attempting uninstall: idna
    Found existing installation: idna 3.6
    Uninstalling idna-3.6:
      Successfully uninstalled idna-3.6
  Attempting uninstall: cycler
    Found existing installation: cycler 0.12.1
    Uninstalling cycler-0.12.1:
      Successfully uninstalled cycler-0.12.1
  Attempting uninstall: chardet
    Found existing installation: chardet 5.2.0
    Uninstalling chardet-5.2.0:
      Successfully uninstalled chardet-5.2.0
  Attempting uninstall: certifi
    Found existing installation: certifi 2024.2.2
    Uninstalling certifi-2024.2.2:
      Successfully uninstalled certifi-2024.2.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
lida 0.0.10 requires fastapi, which is not installed.
lida 0.0.10 requires kaleido, which is not installed.
lida 0.0.10 requires python-multipart, which is not installed.
lida 0.0.10 requires uvicorn, which is not installed.
Successfully installed certifi-2023.7.22 chardet-4.0.0 cycler-0.10.0 idna-2.10 opencv-python-headless-4.8.0.74 python-dotenv-1.0.1 python-magic-0.4.27 requests-toolbelt-1.0.0 roboflow-1.1.19 supervision-0.18.0
```

```
[ ] !python -m pip install 'git+https://github.com/facebookresearch/detectron2.git'
```

```
Collecting git+https://github.com/facebookresearch/detectron2.git
  Cloning https://github.com/facebookresearch/detectron2.git to /tmp/pip-req-build-5ipol5dy
  Running command git clone --filter=blob:none --quiet https://github.com/facebookresearch/detectron2.git /tmp/pip-req-build-5ipol5dy
  Resolved https://github.com/facebookresearch/detectron2.git to commit 1ff5d81cff4417af070970644135c9f28d7481d5c
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Pillow>=7.1 in /usr/local/lib/python3.10/dist-packages (from detectron2==0.6) (9.4.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from detectron2==0.6) (3.7.1)
Requirement already satisfied: pycocotools>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from detectron2==0.6) (2.0.7)
Requirement already satisfied: termcolor>=1.1 in /usr/local/lib/python3.10/dist-packages (from detectron2==0.6) (2.4.0)
Collecting yacs>=0.1.8 (from detectron2==0.6)
  Downloading yacs-0.1.8-py3-none-any.whl (14 kB)
Requirement already satisfied: tabulate in /usr/local/lib/python3.10/dist-packages (from detectron2==0.6) (0.9.0)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from detectron2==0.6) (2.2.1)
Requirement already satisfied: tqdm>4.29.0 in /usr/local/lib/python3.10/dist-packages (from detectron2==0.6) (4.66.2)
Requirement already satisfied: tensorboard in /usr/local/lib/python3.10/dist-packages (from detectron2==0.6) (2.15.2)
Collecting fvcore<0.1.6,>=0.1.5 (from detectron2==0.6)
  Downloading fvcore-0.1.5.post20221221.tar.gz (50 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.2/50.2 kB 1.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting iopath<0.1.10,>=0.1.7 (from detectron2==0.6)
  Downloading iopath-0.1.9-py3-none-any.whl (27 kB)
Collecting omegaconf<2.4,>=2.1 (from detectron2==0.6)
```

```
  Preparing metadata (setup.py) ... done
Collecting portalocker (from iopath<0.1.10,>=0.1.7->detectron2==0.6)
  Downloading portalocker-2.8.2-py3-none-any.whl (17 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->detectron2==0.6) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->detectron2==0.6) (0.10.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->detectron2==0.6) (4.48.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->detectron2==0.6) (1.4.5)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->detectron2==0.6) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->detectron2==0.6) (2.8.2)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from black->detectron2==0.6) (8.1.7)
Collecting mypy-extensions>=0.4.3 (from black->detectron2==0.6)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Collecting pathspec>=0.9.0 (from black->detectron2==0.6)
  Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)
Requirement already satisfied: platformdirs>=2 in /usr/local/lib/python3.10/dist-packages (from black->detectron2==0.6) (4.2.0)
Requirement already satisfied: tomli>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from black->detectron2==0.6) (2.0.1)
Requirement already satisfied: typing-extensions>=4.0.1 in /usr/local/lib/python3.10/dist-packages (from black->detectron2==0.6) (4.9.0)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.10/dist-packages (from tensorboard->detectron2==0.6) (1.4.0)
Requirement already satisfied: grpcio>=1.48.2 in /usr/local/lib/python3.10/dist-packages (from tensorboard->detectron2==0.6) (1.60.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard->detectron2==0.6) (2.27.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard->detectron2==0.6) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard->detectron2==0.6) (3.5.2)
```

Install Roboflow and Detectron2 Pip Packages, Import Deps

```
import logging
import torch
from collections import OrderedDict
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()

# import some common libraries
import numpy as np
import os, json, cv2, random

# import some common detectron2 utilities
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog
from detectron2.data.datasets import register_coco_instances
from detectron2.engine import DefaultTrainer
from detectron2.utils.visualizer import ColorMode
from detectron2.solver import build_lr_scheduler, build_optimizer
from detectron2.checkpoint import DetectionCheckpointer, PeriodicCheckpointer
from detectron2.utils.events import EventStorage
from detectron2.modeling import build_model
import detectron2.utils.comm as comm
from detectron2.engine import default_argument_parser, default_setup, default_writers, launch
from detectron2.data import (
    MetadataCatalog,
    build_detection_test_loader,
    build_detection_train_loader,
)
from detectron2.evaluation import (
    CityscapesInstanceEvaluator,
    CityscapesSemSegEvaluator,
    COCOEvaluator,
    COCOPanopticEvaluator,
    DatasetEvaluators,
    LVISEvaluator,
    PascalVOCDetectionEvaluator,
    SemSegEvaluator,
    inference_on_dataset,
    print_csv_format,
)

from roboflow import Roboflow

from matplotlib import pyplot as plt
from PIL import Image
```

```
[ ] VERSION = 8

    rf = Roboflow(api_key="vjnd8B5LR2vbqFguBA22")
    project = rf.workspace("mubarak").project("test-gsia6")
    dataset = project.version(8).download("coco")

    loading Roboflow workspace...
    loading Roboflow project...
    Downloading Dataset Version Zip in test-8 to coco:: 100%|██████████| 54349/54349 [00:00<00:00, 68715.08it/s]

    Extracting Dataset Version Zip to test-8 in coco:: 100%|██████████| 1126/1126 [00:00<00:00, 3782.83it/s]
```

Download dataset via Roboflow pip package and show a sample annotation

```
[ ] register_coco_instances("asl_poly_train", {}, f"./test-{VERSION}/train/_annotations.coco.json", f"./test-{VERSION}/train/")
    register_coco_instances("asl_poly_valid", {}, f"./test-{VERSION}/valid/_annotations.coco.json", f"./test-{VERSION}/valid/")
    register_coco_instances("asl_poly_test", {}, f"./test-{VERSION}/test/_annotations.coco.json", f"./test-{VERSION}/test/")

    ---------------------------------------------------------------------------
    AssertionError                            Traceback (most recent call last)
    <ipython-input-14-07d8c07568af> in <cell line: 1>()
    ----> 1 register_coco_instances("asl_poly_train", {}, f"./test-{VERSION}/train/_annotations.coco.json", f"./test-{VERSION}/train/")
          2 register_coco_instances("asl_poly_valid", {}, f"./test-{VERSION}/valid/_annotations.coco.json", f"./test-{VERSION}/valid/")
          3 register_coco_instances("asl_poly_test", {}, f"./test-{VERSION}/test/_annotations.coco.json", f"./test-{VERSION}/test/")

    ------------------------------------------------------------ 3 frames ------------------------------------------------------------
    /usr/local/lib/python3.10/dist-packages/detectron2/data/catalog.py in register(self, name, func)
         35         """
         36         assert callable(func), "You must register a function with `DatasetCatalog.register`!"
    ---> 37         assert name not in self, "Dataset '{}' is already registered!".format(name)
         38         self[name] = func
         39

    AssertionError: Dataset 'asl_poly_train' is already registered!
```

```
[ ] dataset_train = DatasetCatalog.get("asl_poly_train")
    fig, ax = plt.subplots()
    dataset_dict = random.choice(dataset_train)
    im = Image.open(dataset_dict['file_name'])
    ax.imshow(im)
    for ann in dataset_dict['annotations']:
        for poly in ann['segmentation']:
            x = poly[0::2]
```

```
linewidth=1, color='blue')

---------------------------------------------------------------
                                Traceback (most recent call last)
<ipython-input-8a58a347> in <cell line: 1>()
----> 1 dataset_dicts = DatasetCatalog.get("asl_poly_train")
      2 fig = plt.subplots()
      3 for d in random.choice(dataset_train):
      4     img = cv2.imread(dataset_dict['file_name'])

                                  3 frames
/usr/local/lib/python3.10/dist-packages/pycocotools/coco.py in __init__(self, annotation_file)
     79             print('loading annotations into memory...')
     80             tic = time.time()
---> 81             with open(annotation_file, 'r') as f:
     82                 dataset = json.load(f)
     83             assert type(dataset)==dict, 'annotation file format {} not supported'.format(type(dataset))

FileNotFoundError: [Errno 2] No such file or directory: './test-1/train/_annotations.coco.json'
```

Configure Detectron2 for fine tuning from COCO, Define training loop helper functions (from detectron2 repo), Run custom training loop

```
[ ] cfg = get_cfg()
    cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
    cfg.DATASETS.TRAIN = ("asl_poly_train",)#Train dataset registered in a previous cell
    cfg.DATASETS.TEST = ("asl_poly_test",)#Test dataset registered in a previous cell
    cfg.DATALOADER.NUM_WORKERS = 2
    cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")  # let training initialize from model zoo
    cfg.SOLVER.IMS_PER_BATCH = 2
    cfg.SOLVER.BASE_LR = 0.00025
    cfg.SOLVER.MAX_ITER = 1000 #We found that with a patience of 500, training will early stop before 10,000 iterations
    cfg.SOLVER.STEPS = []
    cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512
    cfg.MODEL.ROI_HEADS.NUM_CLASSES = 27 # 26 letters plus one super class
    cfg.TEST.EVAL_PERIOD = 0 # Increase this number if you want to monitor validation performance during training

    PATIENCE = 500 #Early stopping will occur after N iterations of no improvement in total_loss

    os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
```

```
[ ] def get_evaluator(cfg, dataset_name, output_folder=None):
        """
        Create evaluator(s) for a given dataset.
        This uses the special metadata "evaluator_type" associated with each builtin dataset.
        For your own dataset, you can simply create an evaluator manually in your
        script and do not have to worry about the hacky if-else logic here.
        """
        if output_folder is None:
            output_folder = os.path.join(cfg.OUTPUT_DIR, "inference")
        evaluator_list = []
        evaluator_type = MetadataCatalog.get(dataset_name).evaluator_type
        if evaluator_type in ["sem_seg", "coco_panoptic_seg"]:
            evaluator_list.append(
                SemSegEvaluator(
                    dataset_name,
                    distributed=True,
                    output_dir=output_folder,
                )
            )
        if evaluator_type in ["coco", "coco_panoptic_seg"]:
            evaluator_list.append(COCOEvaluator(dataset_name, output_dir=output_folder))
        if evaluator_type == "coco_panoptic_seg":
            evaluator_list.append(COCOPanopticEvaluator(dataset_name, output_folder))
        if evaluator_type == "cityscapes_instance":
            return CityscapesInstanceEvaluator(dataset_name)
        if evaluator_type == "cityscapes_sem_seg":
            return CityscapesSemSegEvaluator(dataset_name)
        if evaluator_type == "pascal_voc":
            return PascalVOCDetectionEvaluator(dataset_name)
        if evaluator_type == "lvis":
            return LVISEvaluator(dataset_name, cfg, True, output_folder)
        if len(evaluator_list) == 0:
            raise NotImplementedError(
                "no Evaluator for the dataset {} with the type {}".format(dataset_name, evaluator_type)
            )
        if len(evaluator_list) == 1:
            return evaluator_list[0]
        return DatasetEvaluators(evaluator_list)

    def do_test(cfg, model):
        results = OrderedDict()
        for dataset_name in cfg.DATASETS.TEST:
            data_loader = build_detection_test_loader(cfg, dataset_name)
            evaluator = get_evaluator(
                cfg, dataset_name, os.path.join(cfg.OUTPUT_DIR, "inference", dataset_name)
            )
            results_i = inference_on_dataset(model, data_loader, evaluator)
            results[dataset_name] = results_i
            if comm.is_main_process():
                logger.info("Evaluation results for {} in csv format:".format(dataset_name))
                print_csv_format(results_i)
        if len(results) == 1:
            results = list(results.values())[0]
        return results


    logger = logging.getLogger("detectron2")
    resume=false
    model = build_model(cfg)
    optimizer = build_optimizer(cfg, model)
    scheduler = build_lr_scheduler(cfg, optimizer)

    BEST_LOSS = np.inf

    checkpointer = DetectionCheckpointer(
        model, cfg.OUTPUT_DIR, optimizer=optimizer, scheduler=scheduler
    )
    start_iter = (
        checkpointer.resume_or_load(cfg.MODEL.WEIGHTS, resume=resume).get("iteration", -1) + 1
    )
    prev_iter = start_iter
    max_iter = cfg.SOLVER.MAX_ITER

    periodic_checkpointer = PeriodicCheckpointer(
        checkpointer, cfg.SOLVER.CHECKPOINT_PERIOD, max_iter=max_iter
    )

    writers = default_writers(cfg.OUTPUT_DIR, max_iter) if comm.is_main_process() else []

    # compared to "train_net.py", we do not support accurate timing and
    # precise BN here, because they are not trivial to implement in a small training loop
    data_loader = build_detection_train_loader(cfg)
    logger.info("Starting training from iteration {}".format(start_iter))
    patience_counter = 0
    with EventStorage(start_iter) as storage:
        for data, iteration in zip(data_loader, range(start_iter, max_iter)):
            storage.iter = iteration
```

```
        model(data)
        loss_dict.values())
        isfinite(losses).all(), loss_dict

        uced = {k: v.item() for k, v in comm.reduce_dict(loss_dict).items()}
        l = sum(loss for loss in loss_dict_reduced.values())
        in_process():
        st_scalars(total_loss=losses_reduced, **loss_dict_reduced)

        _grad()
        ()
    optimizer.step()
    storage.put_scalar("lr", optimizer.param_groups[0]["lr"], smoothing_hint=False)
    scheduler.step()

    if (
        cfg.TEST.EVAL_PERIOD > 0
        and (iteration + 1) % cfg.TEST.EVAL_PERIOD == 0
        and iteration != max_iter - 1
    ):
        do_test(cfg, model)
        # Compared to "train_net.py", the test results are not dumped to EventStorage
        comm.synchronize()

    if iteration - start_iter > 5 and (
        (iteration + 1) % 20 == 0 or iteration == max_iter - 1
    ):
        for writer in writers:
            writer.write()
    periodic_checkpointer.step(iteration)

    if iteration > prev_iter:
        prev_iter = iteration
        if losses_reduced < BEST_LOSS:
            BEST_LOSS = losses_reduced
            patience_counter = 0
        else:
            patience_counter += 1
            if patience_counter % 100 == 0:
                print(f"Loss has not improved for {patience_counter} iterations")
            if patience_counter >= PATIENCE:
                print(f"EARLY STOPPING")
                break

do_test(cfg, model)
```

```
[09/11 08:15:20 d2.checkpoint.detection_checkpoint]: [DetectionCheckpointer] Loading from https://dl.fbaipublicfiles.com/detectron2/COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x/137849600/model_final_f10217.pkl ...
model_final_f10217.pkl: 178MB [00:01, 93.4MB/s]
WARNING [fvcore.common.checkpoint]: Skip loading parameter 'roi_heads.box_predictor.cls_score.weight' to the model due to incompatible shapes: (81, 1024) in the checkpoint but (28, 1024) in the model! You might want to double check if this is
WARNING [fvcore.common.checkpoint]: Skip loading parameter 'roi_heads.box_predictor.cls_score.bias' to the model due to incompatible shapes: (81,) in the checkpoint but (28,) in the model! You might want to double check if this is expected.
WARNING [fvcore.common.checkpoint]: Skip loading parameter 'roi_heads.box_predictor.bbox_pred.weight' to the model due to incompatible shapes: (320, 1024) in the checkpoint but (108, 1024) in the model! You might want to double check if this
WARNING [fvcore.common.checkpoint]: Skip loading parameter 'roi_heads.box_predictor.bbox_pred.bias' to the model due to incompatible shapes: (320,) in the checkpoint but (108,) in the model! You might want to double check if this is expected.
WARNING [fvcore.common.checkpoint]: Skip loading parameter 'roi_heads.mask_head.predictor.weight' to the model due to incompatible shapes: (80, 256, 1, 1) in the checkpoint but (27, 256, 1, 1) in the model! You might want to double check if
WARNING [fvcore.common.checkpoint]: Skip loading parameter 'roi_heads.mask_head.predictor.bias' to the model due to incompatible shapes: (80,) in the checkpoint but (27,) in the model! You might want to double check if this is expected.
WARNING [fvcore.common.checkpoint]: Some model parameters or buffers are not found in the checkpoint:
roi_heads.box_predictor.bbox_pred.{bias, weight}
roi_heads.box_predictor.cls_score.{bias, weight}
roi_heads.mask_head.predictor.{bias, weight}
WARNING [09/11 08:15:23 d2.data.datasets.coco]:
Category ids in annotations are not in [1, #categories]! We'll apply a mapping for you.

[09/11 08:15:23 d2.data.datasets.coco]: Loaded 617 images in COCO format from ../asl-0/train/_annotations.coco.json
[09/11 08:15:23 d2.data.build]: Removed 0 images with no usable annotations. 617 images left.
[09/11 08:15:23 d2.data.build]: Distribution of instances among all 2 categories:
|  category  | #instances   |  category  | #instances   |
|------------|--------------|------------|--------------|
|   shrimp   | 0            |   shrimp   | 3612         |
|            |              |            |              |
|   total    | 3612         |            |              |
[09/11 08:15:23 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used in training: [ResizeShortestEdge(short_edge_length=(640, 672, 704, 736, 768, 800), max_size=1333, sample_style='choice'), RandomFlip()]
[09/11 08:15:23 d2.data.build]: Using training sampler TrainingSampler
[09/11 08:15:23 d2.data.common]: Serializing the dataset using: <class 'detectron2.data.common.TorchSerializedList'>
[09/11 08:15:23 d2.data.common]: Serializing 617 elements to byte tensors and concatenating them all ...
[09/11 08:15:23 d2.data.common]: Serialized dataset takes 5.11 MiB
[09/11 08:15:24 detectron2]: Starting training from iteration 0
torch.meshgrid: in an upcoming release, it will be required to pass the indexing argument. (Triggered internally at ../aten/src/ATen/native/TensorShape.cpp:3483.)
[09/11 08:15:30 d2.utils.events]:  iter: 19  total_loss: 2.498  loss_cls: 1.814  loss_box_reg: 0.427  loss_mask: 0.1893  loss_rpn_cls: 0.01298  loss_rpn_loc: 0.01784     lr: 1.9953e-05  max_mem: 2102M
[09/11 08:15:38 d2.utils.events]:  eta: 0:08:06  iter: 39  total_loss: 2.291  loss_cls: 1.491  loss_box_reg: 0.4638  loss_mask: 0.1798  loss_rpn_cls: 0.0123  loss_rpn_loc: 0.02285     lr: 3.9902e-05  max_mem: 2102M
[09/11 08:15:48 d2.utils.events]:  eta: 0:08:10  iter: 59  total_loss: 2.011  loss_cls: 1.207  loss_box_reg: 0.4852  loss_mask: 0.1673  loss_rpn_cls: 0.01777  loss_rpn_loc: 0.04189     lr: 5.9851e-05  max_mem: 2102M
[09/11 08:16:06 d2.utils.events]:  eta: 0:08:05  iter: 79  total_loss: 3.435  loss_cls: 2.19  loss_box_reg: 0.5186  loss_mask: 0.1825  loss_rpn_cls: 0.01868  loss_rpn_loc: 0.04644     lr: 7.9801e-05  max_mem: 2102M
[09/11 08:16:15 d2.utils.events]:  eta: 0:08:32  iter: 99  total_loss: 2.727  loss_cls: 1.59  loss_box_reg: 0.5151  loss_mask: 0.1907  loss_rpn_cls: 0.01671  loss_rpn_loc: 0.04777     lr: 9.9751e-05  max_mem: 2102M
[09/11 08:16:23 d2.utils.events]:  eta: 0:08:30  iter: 119  total_loss: 2.079  loss_cls: 0.7528  loss_box_reg: 0.4848  loss_mask: 0.1685  loss_rpn_cls: 0.012  loss_rpn_loc: 0.05272     lr: 0.00011970  max_mem: 2102M
[09/11 08:16:31 d2.utils.events]:  eta: 0:08:56  iter: 139  total_loss: 1.813  loss_cls: 0.6702  loss_box_reg: 0.4064  loss_mask: 0.1688  loss_rpn_cls: 0.01487  loss_rpn_loc: 0.04609     lr: 0.00013965  max_mem: 2102M
[09/11 08:16:40 d2.utils.events]:  eta: 0:08:16  iter: 159  total_loss: 1.648  loss_cls: 0.5668  loss_box_reg: 0.5246  loss_mask: 0.1734  loss_rpn_cls: 0.01039  loss_rpn_loc: 0.0389     lr: 0.00015960  max_mem: 2102M
[09/11 08:16:49 d2.utils.events]:  eta: 0:08:20  iter: 179  total_loss: 1.51  loss_cls: 0.4695  loss_box_reg: 0.5025  loss_mask: 0.1453  loss_rpn_cls: 0.01458  loss_rpn_loc: 0.03555     lr: 0.00017955  max_mem: 2102M
[09/11 08:16:59 d2.utils.events]:  eta: 0:08:14  iter: 199  total_loss: 1.392  loss_cls: 0.3908  loss_box_reg: 0.5154  loss_mask: 0.1436  loss_rpn_cls: 0.01311  loss_rpn_loc: 0.03503     lr: 0.00019950  max_mem: 2102M
[09/11 08:17:08 d2.utils.events]:  eta: 0:08:02  iter: 219  total_loss: 1.237  loss_cls: 0.3415  loss_box_reg: 0.4814  loss_mask: 0.1705  loss_rpn_cls: 0.01298  loss_rpn_loc: 0.04215     lr: 0.00021945  max_mem: 2102M
[09/11 08:17:18 d2.utils.events]:  eta: 0:08:02  iter: 239  total_loss: 1.434  loss_cls: 0.4064  loss_box_reg: 0.5029  loss_mask: 0.1685  loss_rpn_cls: 0.01172  loss_rpn_loc: 0.02995     lr: 0.00023940  max_mem: 2102M
[09/11 08:17:27 d2.utils.events]:  eta: 0:07:53  iter: 259  total_loss: 1.008  loss_cls: 0.2578  loss_box_reg: 0.4779  loss_mask: 0.2098  loss_rpn_cls: 0.01186  loss_rpn_loc: 0.03881     lr: 0.00025935  max_mem: 2102M
[09/11 08:17:37 d2.utils.events]:  eta: 0:07:59  iter: 279  total_loss: 1.016  loss_cls: 0.2423  loss_box_reg: 0.4723  loss_mask: 0.2589  loss_rpn_cls: 0.01542  loss_rpn_loc: 0.04301     lr: 0.00027930  max_mem: 2102M
[09/11 08:17:46 d2.utils.events]:  eta: 0:07:25  iter: 298  total_loss: 0.8401  loss_cls: 0.1958  loss_box_reg: 0.4189  loss_mask: 0.1897  loss_rpn_cls: 0.009293  loss_rpn_loc: 0.02051     lr: 0.00029925  max_mem: 2102M
[09/11 08:17:56 d2.utils.events]:  eta: 0:07:17  iter: 319  total_loss: 0.8678  loss_cls: 0.207  loss_box_reg: 0.4289  loss_mask: 0.214  loss_rpn_cls: 0.007788  loss_rpn_loc: 0.04068     lr: 0.00031920  max_mem: 2102M
[09/11 08:18:05 d2.utils.events]:  eta: 0:04:57  iter: 339  total_loss: 0.8672  loss_cls: 0.1568  loss_box_reg: 0.3373  loss_mask: 0.1589  loss_rpn_cls: 0.005219  loss_rpn_loc: 0.02078     lr: 0.00033915  max_mem: 2102M
[09/11 08:18:13 d2.utils.events]:  eta: 0:04:36  iter: 359  total_loss: 0.6309  loss_cls: 0.14  loss_box_reg: 0.4009  loss_mask: 0.128  loss_rpn_cls: 0.004471  loss_rpn_loc: 0.01448     lr: 0.00035910  max_mem: 2102M
[09/11 08:18:23 d2.utils.events]:  eta: 0:04:55  iter: 379  total_loss: 0.7245  loss_cls: 0.16  loss_box_reg: 0.3395  loss_mask: 0.1701  loss_rpn_cls: 0.005556  loss_rpn_loc: 0.03788     lr: 0.00037905  max_mem: 2102M
[09/11 08:18:32 d2.utils.events]:  eta: 0:04:37  iter: 399  total_loss: 0.7519  loss_cls: 0.1877  loss_box_reg: 0.3218  loss_mask: 0.1909  loss_rpn_cls: 0.005397  loss_rpn_loc: 0.04431     lr: 9.99e-05  max_mem: 2102M
[09/11 08:18:41 d2.utils.events]:  eta: 0:04:30  iter: 419  total_loss: 0.6962  loss_cls: 0.1724  loss_box_reg: 0.2767  loss_mask: 0.1633  loss_rpn_cls: 0.006677  loss_rpn_loc: 0.03324     lr: 0.0001049  max_mem: 2102M
[09/11 08:18:50 d2.utils.events]:  eta: 0:04:02  iter: 439  total_loss: 0.5694  loss_cls: 0.1417  loss_box_reg: 0.2146  loss_mask: 0.1373  loss_rpn_cls: 0.003366  loss_rpn_loc: 0.02667     lr: 0.00010989  max_mem: 2102M
[09/11 08:18:59 d2.utils.events]:  eta: 0:03:52  iter: 459  total_loss: 0.5648  loss_cls: 0.1427  loss_box_reg: 0.2328  loss_mask: 0.14  loss_rpn_cls: 0.001839  loss_rpn_loc: 0.01558     lr: 0.00011489  max_mem: 2102M
[09/11 08:19:07 d2.utils.events]:  eta: 0:03:46  iter: 479  total_loss: 0.6252  loss_cls: 0.1514  loss_box_reg: 0.2379  loss_mask: 0.1646  loss_rpn_cls: 0.006049  loss_rpn_loc: 0.0398     lr: 0.00011988  max_mem: 2102M
[09/11 08:19:16 d2.utils.events]:  eta: 0:03:48  iter: 499  total_loss: 0.5681  loss_cls: 0.1467  loss_box_reg: 0.2277  loss_mask: 0.1353  loss_rpn_cls: 0.004772  loss_rpn_loc: 0.03002     lr: 0.00012488  max_mem: 2102M
[09/11 08:19:25 d2.utils.events]:  eta: 0:03:36  iter: 519  total_loss: 0.4806  loss_cls: 0.1014  loss_box_reg: 0.1684  loss_mask: 0.1098  loss_rpn_cls: 0.001282  loss_rpn_loc: 0.009462     lr: 0.00012987  max_mem: 2102M
[09/11 08:19:34 d2.utils.events]:  eta: 0:03:23  iter: 539  total_loss: 0.443  loss_cls: 0.118  loss_box_reg: 0.1922  loss_mask: 0.1227  loss_rpn_cls: 0.001865  loss_rpn_loc: 0.02346     lr: 0.00013487  max_mem: 2102M
```

[ ]

Test model and show example output

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

```
[ ] import os
    from detectron2 import model_zoo
    from detectron2.engine import DefaultPredictor
    from detectron2.config import get_cfg
    from detectron2.utils.visualizer import Visualizer
    from PIL import Image
    import numpy as np
    import matplotlib.pyplot as plt

    import cv2

    # from google.colab.patches import cv2_imshow

    # os.listdir('/content/drive/Othercomputers/Realmebook/python/projectPython/')
```

```
[ ] cfg = get_cfg()
    cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
    cfg.DATASETS.TRAIN = ("asl_poly_train",)#Train dataset registered in a previous cell
    cfg.DATASETS.TEST = ("asl_poly_test",)#Test dataset registered in a previous cell
    cfg.DATALOADER.NUM_WORKERS = 2
    cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")  # Let training initialize from model zoo
    cfg.SOLVER.IMS_PER_BATCH = 2
```

```
.00025
 100  #We found that with a patience of 500, training will early stop before 10,000 iterations

ATCH_SIZE_PER_IMAGE = 512
M_CLASSES = 27 # 26 letters plus one super class
= 0 # Increase this number if you want to monitor validation performance during training

 stopping will occur after N iterations OF no improvement in total_loss

T_DIR, exist_ok=True)
```

```
[ ] model_path = '/content/drive/Othercomputers/Realmebook/python/projectPython/'
```

```
[ ] # cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")  # path to the model we just trained
    cfg.MODEL.WEIGHTS = os.path.join(model_path, "model_final.pth")  # salam yang ubah
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.2   # set a custom testing threshold
    predictor = DefaultPredictor(cfg)

    dataset_dicts = DatasetCatalog.get("asl_poly_test")
    for d in random.sample(dataset_dicts, 9):
        im = cv2.imread(d["file_name"])
        outputs = predictor(im)
        v = Visualizer(im[:, :, ::-1],
                       scale=0.5,
                       instance_mode=ColorMode.IMAGE_BW
        )
        out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
        im = Image.fromarray(out.get_image()[:, :, ::-1])
        im
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-9-66f8429dbd26> in <cell line: 6>()
      4 predictor = DefaultPredictor(cfg)
      5
----> 6 dataset_dicts = DatasetCatalog.get("asl_poly_test")
      7 for d in random.sample(dataset_dicts, 9):
      8     im = cv2.imread(d["file_name"])

NameError: name 'DatasetCatalog' is not defined
```

```
[ ] # from google.colab import drive
    # drive.mount('/content/drive')

    folder_testing = '/content/drive/Othercomputers/Realmebook/python/projectPython/Matrial&Result/testing/'

    os.listdir(folder_testing)
```

```
['3', '1', '4', 'Ori', '10.jpg', '20.jpg', '30.jpg', '0.jpg', 'Pakan', '3']
```

```
[ ] # Ganti PATH_TO_IMAGE dengan path lengkap ke gambar yang ingin Anda uji
    # Misalnya: "C:/path/to/directory/image.jpg" (untuk Windows) atau "/path/to/directory/image.jpg" (untuk Linux/Mac)
    PATH_TO_IMAGE = "/content/drive/Othercomputers/Realmebook/python/projectPython/Matrial&Result/testing/3/419.2.jpg"

    # cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")  # path to the model we just trained
    cfg.MODEL.WEIGHTS = os.path.join(model_path, "model_final.pth")
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.8   # set a custom testing threshold
    predictor = DefaultPredictor(cfg)

    # Buat dictionary untuk representasi data gambar yang akan diuji
    image_data = {
        "file_name": PATH_TO_IMAGE,
    }

    # Memuat gambar dari path yang ditentukan
    im = cv2.imread(image_data["file_name"])
    outputs = predictor(im)

    num_detected_objects = len(outputs["instances"])
    # Visualisasi hasil prediksi
    v = Visualizer(im[:, :, ::-1],
                   scale=0.5

    )
    out = v.draw_instance_predictions(outputs["instances"].to("cpu"))

    # img = cv2.imread(PATH_TO_IMAGE, cv2.IMREAD_UNCHANGED)
    # cv2_imshow(img)

    # Konversi gambar kembali ke format Image dan tampilkan
    im = Image.fromarray(out.get_image()[:, :, ::-1])
    # im.show()
    display(im)
    print("Jumlah objek yang terdeteksi:", num_detected_objects)
```

```
Jumlah objek yang terdeteksi: 35
```

```
[ ] mask = outputs['instances'].get('pred_masks').to('cpu').numpy()
    len(mask)
```

```
35
```

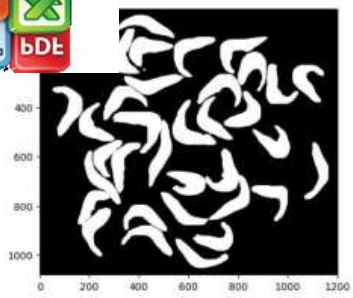
```
[ ] tinggi = len(mask[0])
    lebar = len(mask[0][0])
    bg_mask = np.full((tinggi, lebar), False)
    # bg_mask
```

```
[ ] for index in range(len(mask)):
      data = mask[index]
      for i in range(tinggi):
        for j in range(lebar):
```

```
= True

map='gray')

<matplotlib.image.AxesImage at 0x7c271f0c7510>
```




```
[ ] total = 0
    for i in range(len(mask)):
      # plt.imshow(mask[i])
      unique, counts = np.unique(mask[i], return_counts=True)
      # np.unique(mask[i])
      pixel_true = np.asarray((unique, counts)).T[1][1]
      total += pixel_true
      # break

    print(f'total pixel udang = {total}')
    print(f'total pixel gambar = {tinggi * lebar}')

    total pixel udang = 348745
    total pixel gambar = 1207080

[ ] total /  len(mask)

    9964.142857142857

[ ] from google.colab.patches import cv2_imshow

[ ] # img = cv2.imread(PATH_TO_IMAGE, cv2.IMREAD_UNCHANGED)
    # cv2_imshow(img)
```

Download trained model for future inference

```
[ ] from google.colab import files
    files.download(os.path.join(cfg.OUTPUT_DIR, "model_final.pth"))

[ ] import os
    from detectron2.evaluation import COCOEvaluator, inference_on_dataset
    from detectron2.data import build_detection_test_loader

    # Ganti PATH_TO_MODEL_WEIGHTS dengan path lengkap ke berat model (model_final.pth)
    PATH_TO_MODEL_WEIGHTS = "PATH_TO_MODEL_WEIGHTS.pth"

    cfg.MODEL.WEIGHTS = PATH_TO_MODEL_WEIGHTS
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.2

    # Buat evaluator COCO untuk menghitung mAP
    evaluator = COCOEvaluator("asl_poly_test", cfg, False, output_dir="./output/")

    # Buat data loader untuk dataset tes
    data_loader = build_detection_test_loader(cfg, "asl_poly_test")

    # Evaluasi model menggunakan COCOEvaluator
    inference_on_dataset(predictor.model, data_loader, evaluator)

    # Cetak hasil evaluasi
    print(evaluator.evaluate())

    WARNING [07/21 19:13:13 d2.evaluation.coco_evaluation]: COCO Evaluator instantiated using config, this is deprecated behavior. Please pass in explicit arguments instead.
    WARNING [07/21 19:13:13 d2.data.datasets.coco]:
    Category ids in annotations are not in [1, #categories]! We'll apply a mapping for you.

    [07/21 19:13:13 d2.data.datasets.coco]: Loaded 1 images in COCO format from ./test-7/test/_annotations.coco.json
    [07/21 19:13:13 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used in inference: [ResizeShortestEdge(short_edge_length=(800, 800), max_size=1333, sample_style='choice')]
    [07/21 19:13:13 d2.data.common]: Serializing the dataset using: <class 'detectron2.data.common._TorchSerializedList'>
    [07/21 19:13:13 d2.data.common]: Serializing 1 elements to byte tensors and concatenating them all ...
    [07/21 19:13:13 d2.data.common]: Serialized dataset takes 0.00 MiB
    [07/21 19:13:13 d2.evaluation.evaluator]: Start inference on 1 batches
    [07/21 19:13:13 d2.evaluation.evaluator]: Inference done 1/1. Dataloading: 0.0000 s/iter. Inference: 0.1366 s/iter. Eval: 0.0035 s/iter. Total: 0.1400 s/iter. ETA=0:00:00
    [07/21 19:13:13 d2.evaluation.evaluator]: Total inference time: 0:00:00.196947 (0.196947 s / iter per device, on 1 devices)
    [07/21 19:13:13 d2.evaluation.evaluator]: Total inference pure compute time: 0:00:00 (0.136562 s / iter per device, on 1 devices)
    [07/21 19:13:13 d2.evaluation.coco_evaluation]: Preparing results for COCO format ...
    [07/21 19:13:13 d2.evaluation.coco_evaluation]: Saving results to ./output/coco_instances_results.json
    [07/21 19:13:13 d2.evaluation.coco_evaluation]: Evaluating predictions with unofficial COCO API...
    Loading and preparing results...
    DONE (t=0.00s)
    creating index...
    index created!
    [07/21 19:13:13 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
    [07/21 19:13:13 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.01 seconds.
    [07/21 19:13:13 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
    [07/21 19:13:13 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.01 seconds.
     Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
     Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.000
     Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
     Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
     Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
     Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.000
     Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.000
     Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.000
     Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
     Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
     Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
     Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.000
    [07/21 19:13:13 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
    |  AP   |  AP50  |  AP75  |  APs  |  APm  |  APl  |
    |:------:|:------:|:------:|:-----:|:-----:|:-----:|
    | 0.000 | 0.000 | 0.000 |  nan  |  nan  | 0.000 |
    [07/21 19:13:13 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
    [07/21 19:13:13 d2.evaluation.coco_evaluation]: Per-category bbox AP:
    | category   | AP   | category   | AP    |
    |:-----------|:-----|:-----------|:------|
    | shrimp     | nan  | shrimp     | 0.000 |
    Loading and preparing results...
    DONE (t=0.00s)
    creating index...
    index created!
    [07/21 19:13:13 d2.evaluation.fast_eval_api]: Evaluate annotation type *segm*
    [07/21 19:13:13 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.00 seconds.
    [07/21 19:13:13 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
```

```
valuation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.01 seconds.
(AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
```

```
[ ]
```