

# Progress Report 2: Proyek UAS

*Neural Network untuk Deteksi Objek: Analisis Komparatif Faster R-CNN dan YOLO*

## Kelompok 9

Achmad Zaidan Lazuardy (2206059793)

Andrew Kristofer Jian (2206059673)

Darren Adam Dewantoro (2206816600)

Hanif Nur Ilham Sanjaya (2206059692)

Samuel Tanaka Sibarani (2206059710)

Universitas Indonesia

27 Mei 2025

# Pendahuluan

## Tujuan Laporan Proyek UAS Bagian Ini:

- Melaporkan kemajuan pada tugas deteksi objek.
- Mengimplementasikan dan mengevaluasi dua model *deep learning* populer untuk deteksi objek:
  - Faster R-CNN
  - YOLO (You Only Look Once)
- Melakukan analisis komparatif kinerja kedua model berdasarkan metrik standar.
- Dataset yang digunakan adalah Aquarium Data COTS, yang berisi gambar objek bawah air.

## Konteks Proyek Keseluruhan:

- Mengembangkan model *neural network* untuk klasifikasi, deteksi objek, dan prediksi data sekuensial.
- Dataset dari Kaggle, diuji pada *test set* yang sama untuk perbandingan antar kelompok.

# Metodologi Deteksi Objek

## Model yang Digunakan:

- **Faster R-CNN**
  - Arsitektur: ResNet50-FPN
  - Pretrained pada COCO
- **YOLOv8n**
  - Pretrained pada COCO

## Preprocessing & Training:

- Resize normalisasi input
- Augmentasi (rotasi, flip, ...)
- Optimizer: Adam / SGD
- Loss: sesuai standar model
- Batch size & epochs (misal 20 50)

## Metrik Evaluasi:

- mAP@0.5, mAP@0.5:0.95
- Precision, Recall
- Inference time per gambar
- FPS
- Ukuran model (MB) & jumlah parameter (M)

# Kurva Training dan Validasi Loss Faster R-CNN

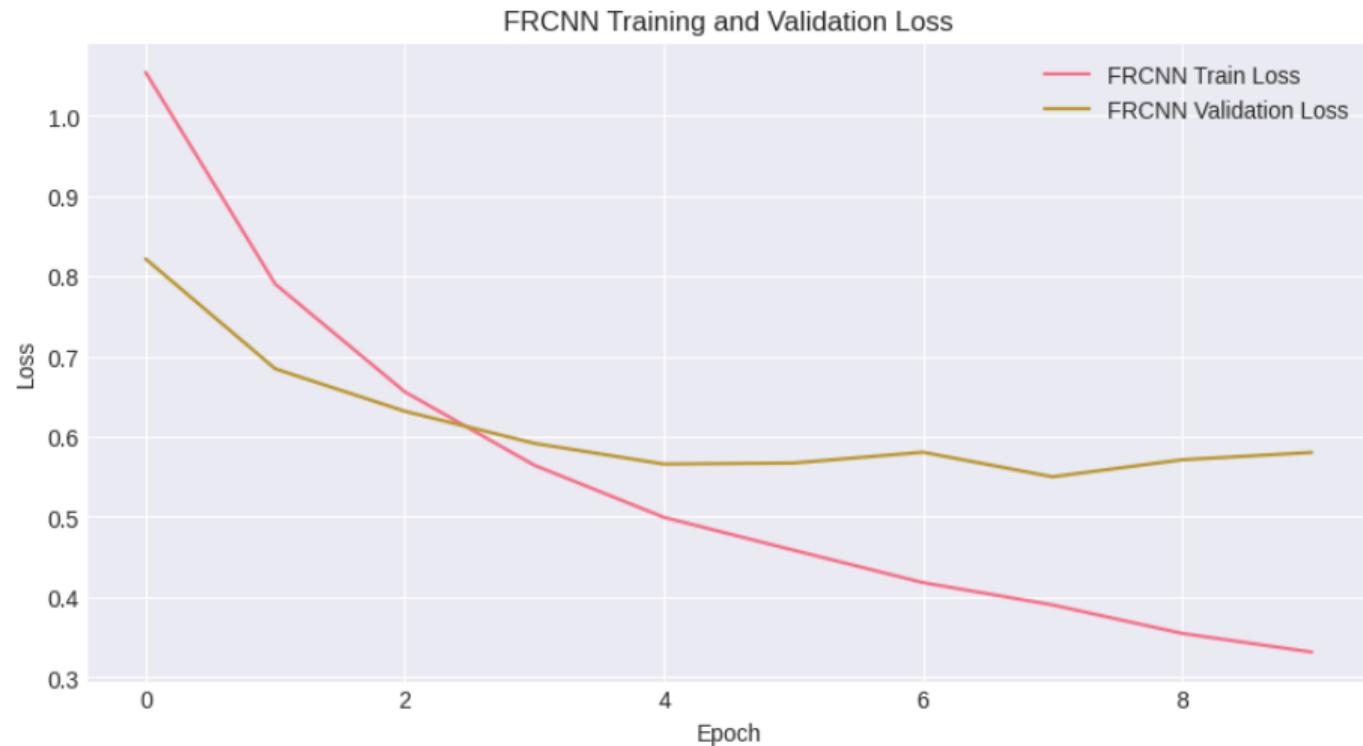
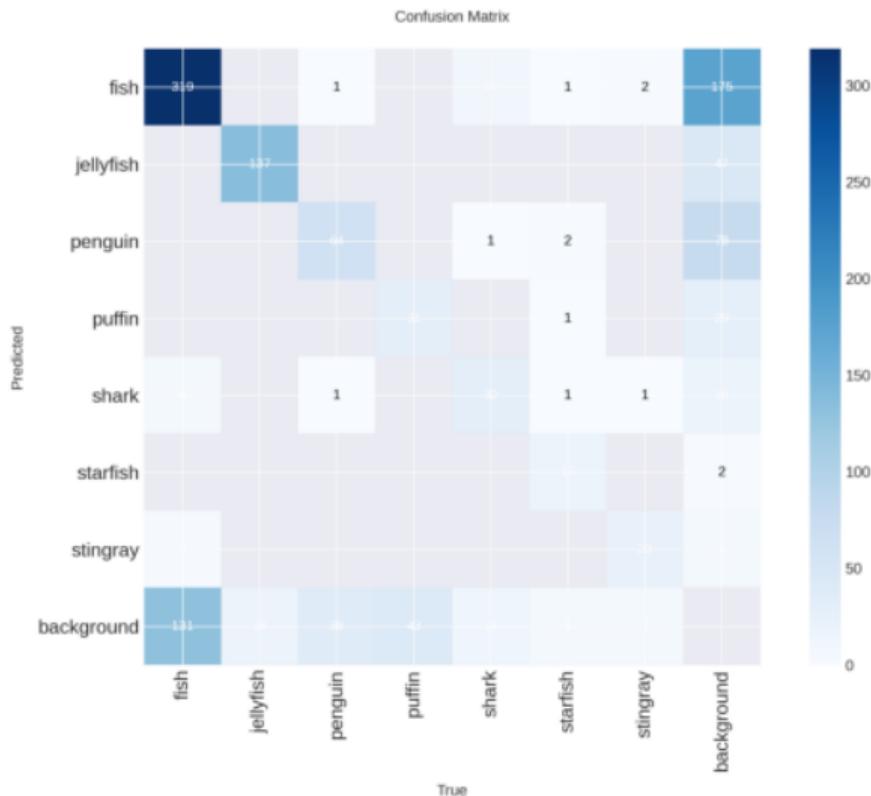


Figure: Kurva loss training dan validasi untuk model Faster R-CNN selama epoch.

# Confusion Matrix Model YOLO

YOLO Confusion Matrix (Validation)



# Perbandingan Metrik Kinerja Model

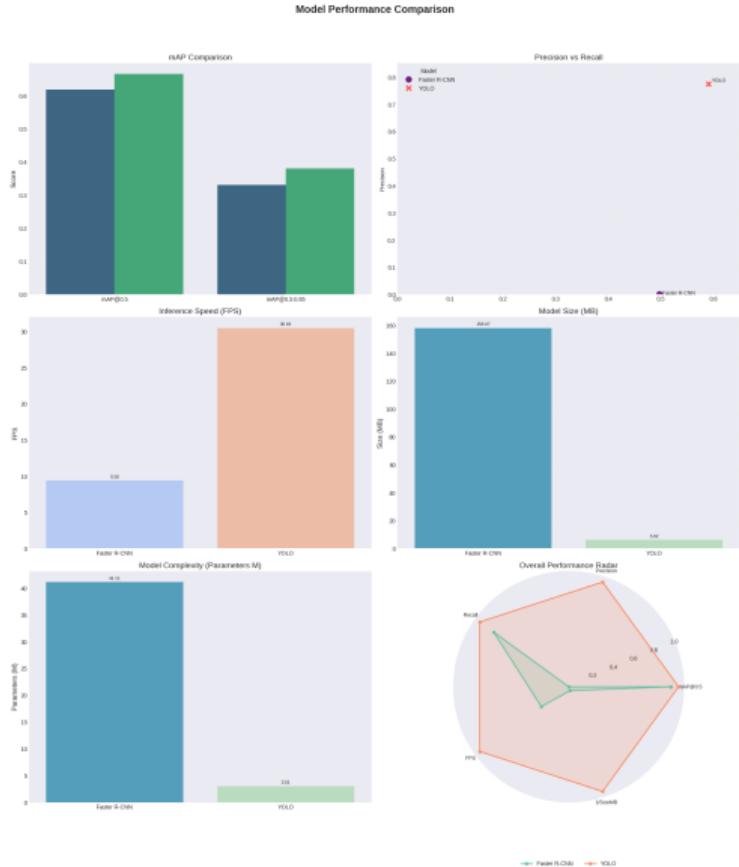
Table: Perbandingan Metrik Kinerja Model Deteksi Objek (Aquarium COTS Dataset)

Model	mAP @0.5	mAP @0.5:.95	Prec.	Recall	Infer. Time (s)	FPS	Size (MB)	Params (M)	Catatan
Faster R-CNN	0.6185	0.3310	0.0000	0.4974	0.1066	9.38	158.07	41.11	Precision metric might not be directly comparable to YOLO (usually not primary torchmetrics output for detection). Recall metric is MAR@100 from torchmetrics. Inference time updated based on typical FPS.
YOLO	0.6656	0.3799	0.7744	0.5907	0.0328	30.49	5.92	3.01	Performa lebih unggul di sebagian besar metrik.

## Catatan Tabel

Null.

# Visualisasi Komprehensif Perbandingan Model



# Ringkasan dan Analisis Hasil

## Faster R-CNN

- mAP@0.5: 0.6185
- mAP@0.5:0.95: 0.3310
- Precision: 0.0000 (catatan: mungkin tidak langsung sebanding)
- Recall: 0.4974 (MAR@100 dari torchmetrics)
- FPS: 9.38
- Model Size: 158.07 MB
- Parameters: 41.11 M
- **Kelebihan:** Arsitektur dua tahap yang menunjukkan peningkatan akurasi mAP yang signifikan pada update terakhir.
- **Kekurangan:** Masih lebih lambat, model lebih besar, dan parameter lebih banyak dibandingkan YOLO.

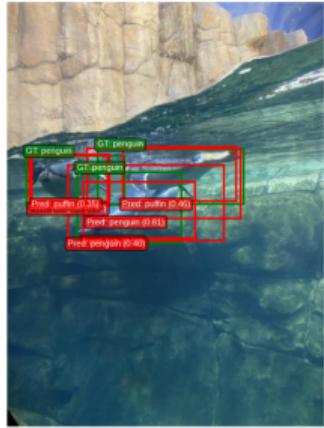
## YOLO

- mAP@0.5: 0.6656
- mAP@0.5:0.95: 0.3799
- Precision: 0.7744
- Recall: 0.5907
- FPS: 30.49
- Model Size: 5.92 MB
- Parameters: 3.01 M
- **Kelebihan:** Performa mAP@0.5 sedikit lebih unggul, kecepatan inferensi (FPS) superior, ukuran model dan jumlah parameter jauh lebih kecil. Sangat cocok untuk aplikasi real-time.
- **Kekurangan:** Akurasi pada objek sangat kecil terkadang bisa menjadi tantangan untuk model satu tahap.

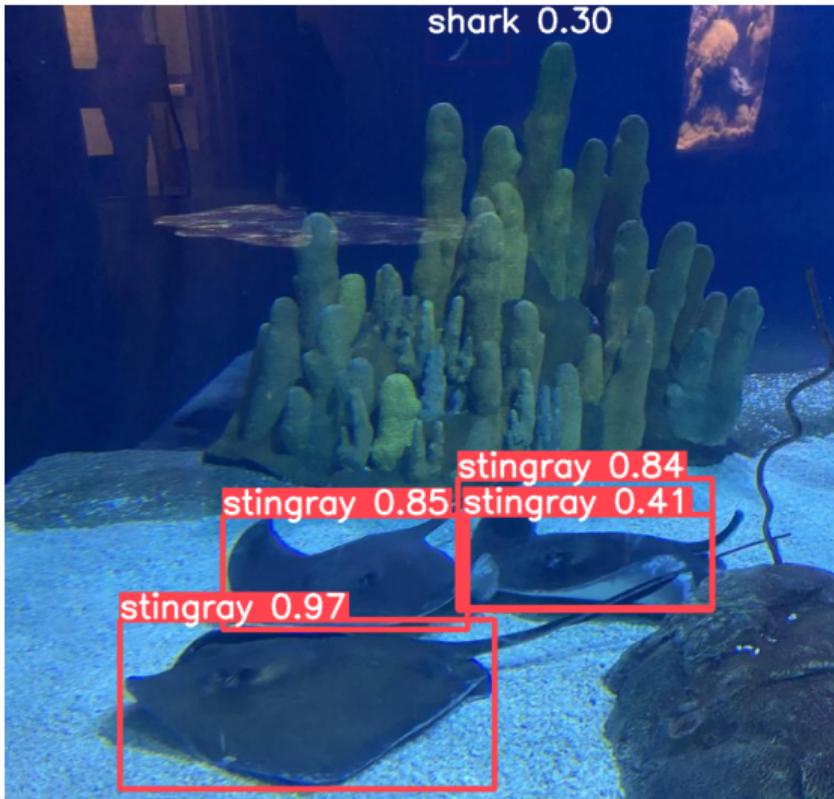
## Perbandingan Hasil Deteksi



FRCNN Sample 55



A photograph of a penguin swimming in an aquarium. The water is clear and reflects the surrounding environment. A green rectangular box highlights the penguin, with the text "GT penguin" above it. The background shows the tiled wall of the aquarium.



# Kesimpulan & Langkah Selanjutnya

## Kesimpulan Deteksi Objek

- Implementasi dan evaluasi model Faster R-CNN dan YOLO untuk deteksi objek pada dataset Aquarium COTS telah berhasil dilakukan.
- Model YOLO menunjukkan kinerja yang sedikit lebih unggul dalam akurasi mAP@0.5 (0.6656) dibandingkan Faster R-CNN (0.6185).
- YOLO tetap superior dalam kecepatan inferensi (30.49 FPS vs 9.38 FPS) dan efisiensi model (ukuran 5.92MB vs 158.07MB).
- Hasil ini menyoroti keunggulan arsitektur YOLO untuk tugas deteksi objek pada dataset ini, terutama untuk potensi aplikasi yang membutuhkan kecepatan dan efisiensi tinggi.

## Langkah Selanjutnya

- **Fine-tuning Lanjutan:** Melakukan fine-tuning lebih lanjut pada kedua model (misalnya, penyesuaian hyperparameter, scheduler laju pembelajaran) untuk potensi peningkatan akurasi.
- **Analisis Error Mendalam:** Menganalisis kesalahan prediksi kedua model (misalnya, false positives/negatives) untuk memahami keterbatasan dan area perbaikan.
- **Augmentasi Data Lanjutan:** Mengeksplorasi teknik augmentasi data yang lebih canggih yang spesifik untuk objek bawah air.
- **Perbandingan dengan Varian Model Lain:** Jika memungkinkan, membandingkan dengan varian model lain atau model deteksi objek state-of-the-art lainnya.

## **DESKRIPSI DATASET**

### **Butterfly Image Classification**

Identify the class to which each butterfly belongs to



Dataset ini memiliki 75 kelas berbeda dari jenis kupu-kupu. Terdapat lebih dari 1000 gambar yang telah diberi label, termasuk gambar untuk validasi. Setiap gambar hanya mewakili satu kategori kupu-kupu. Label untuk setiap gambar disimpan dalam file Training\_set.csv.

## CLASSIFICATION MODELS

### ★ ANN Multilayer Perceptron

Adalah jenis Artificial Neural Network (ANN) yang terdiri dari beberapa lapisan neuron, termasuk input, hidden, dan output layer. MLP bekerja dengan memproses data melalui koneksi penuh antar neuron dan mampu melakukan klasifikasi non-linear.

### ★ ResNet

adalah arsitektur deep learning yang dirancang untuk mengatasi degradasi akurasi pada jaringan sangat dalam. Keunggulan utamanya terletak pada penggunaan residual connection, yang memungkinkan jaringan belajar lebih efektif meskipun memiliki banyak lapisan.

## OBJECT DETECTION MODELS

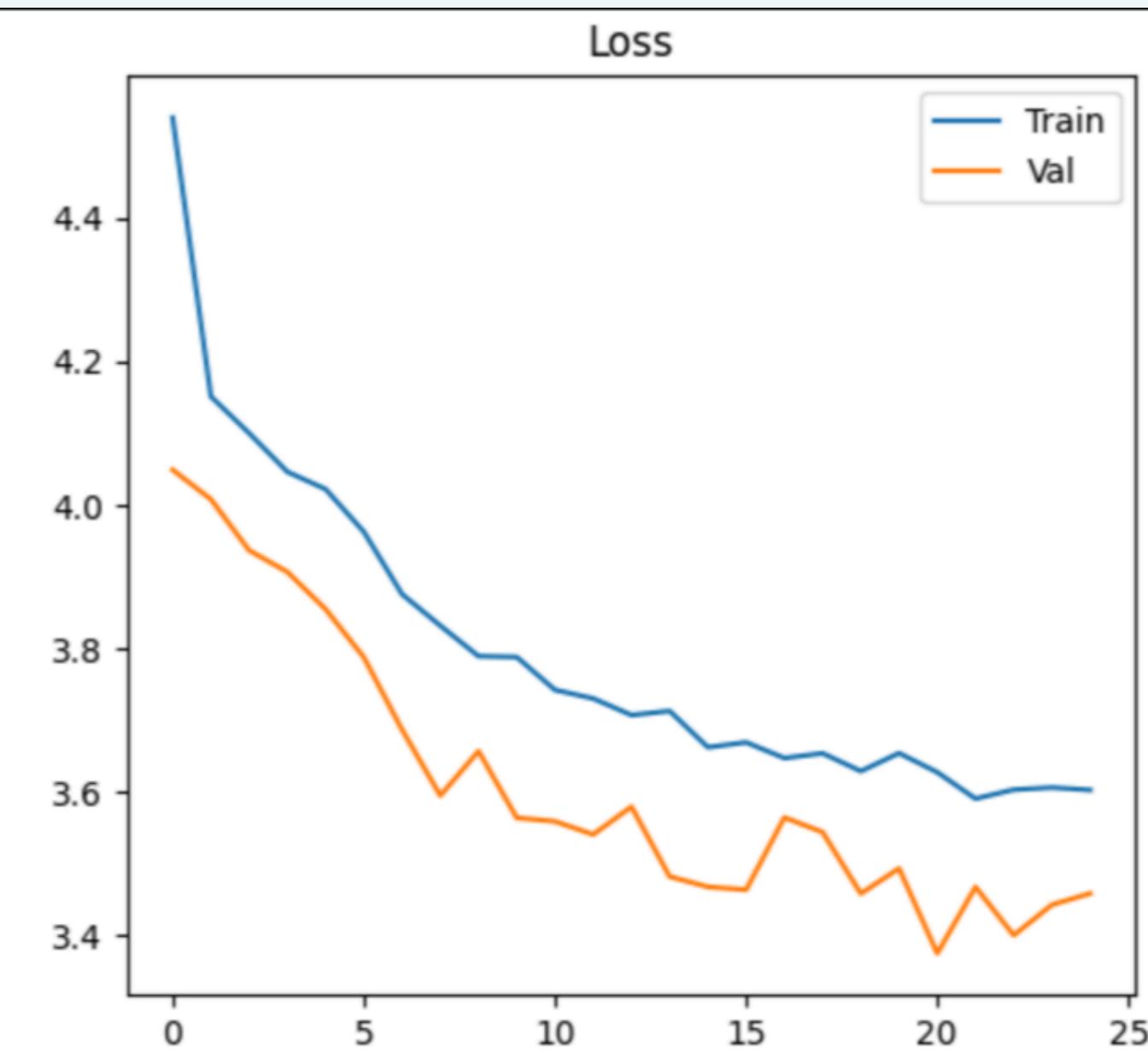
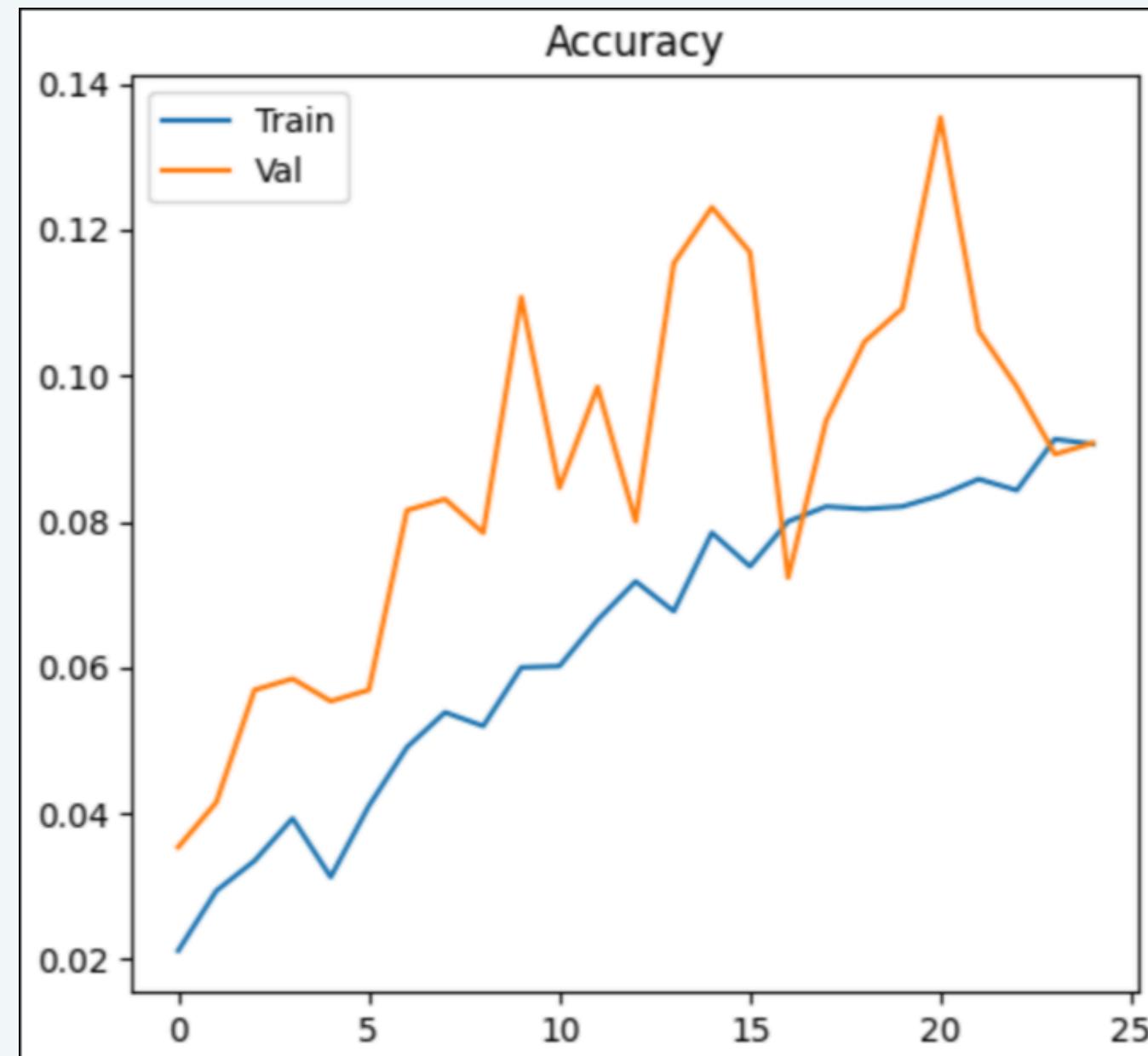
### **Faster R-CNN**

Model pendekripsi objek yang mampu mengidentifikasi objek dalam sebuah gambar dan membentuk kotak-kotak pembatas di sekelilingnya, sembari mengklasifikasikan objek-objek tersebut.

### **YOLO**

YOLO (You Only Look Once) merupakan algoritma deteksi objek berbasis deep learning yang mampu mengenali objek dalam sebuah gambar hanya melalui satu kali proses pemindaian. Kelebihan utama dari YOLO adalah kecepatannya yang tinggi dalam melakukan deteksi objek.

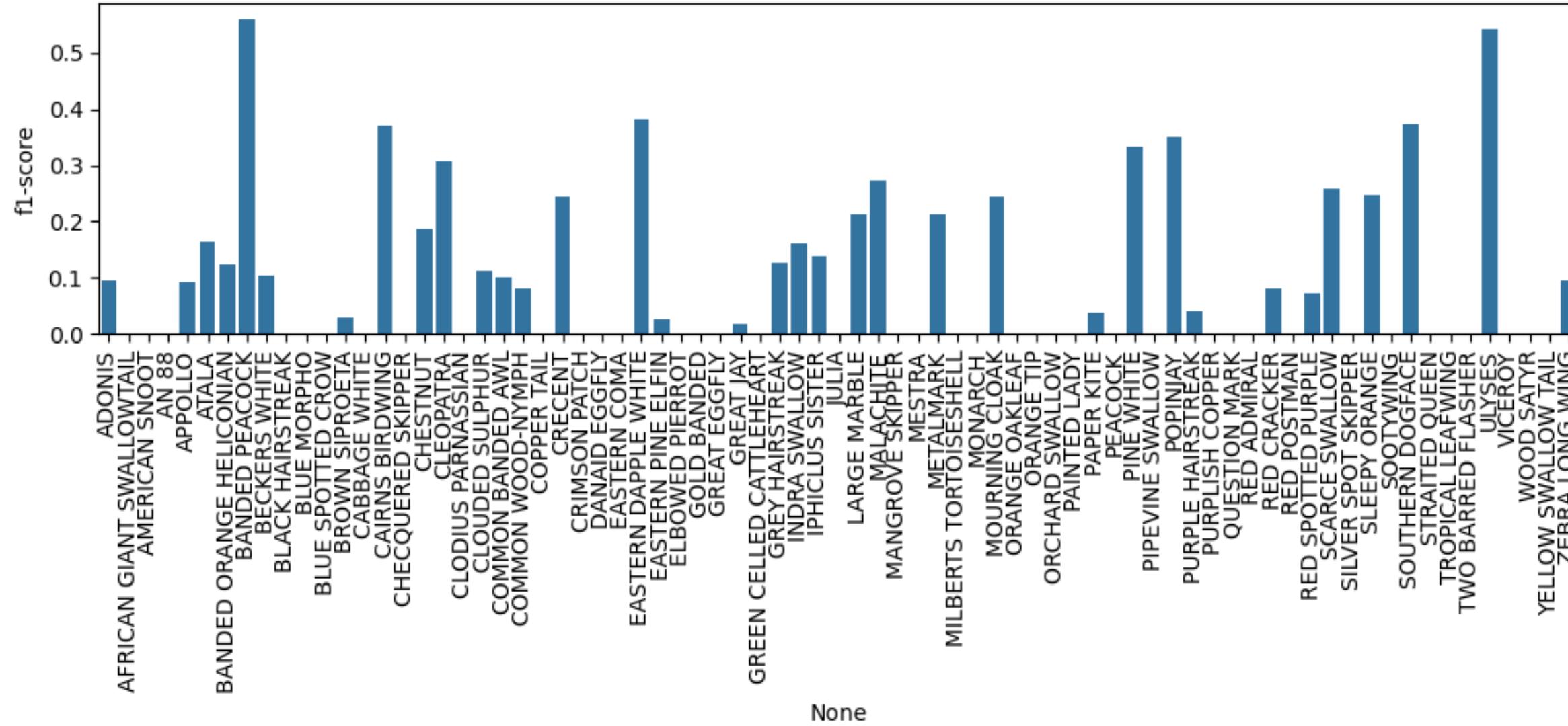
## Hasil ANN Multilayer Perception



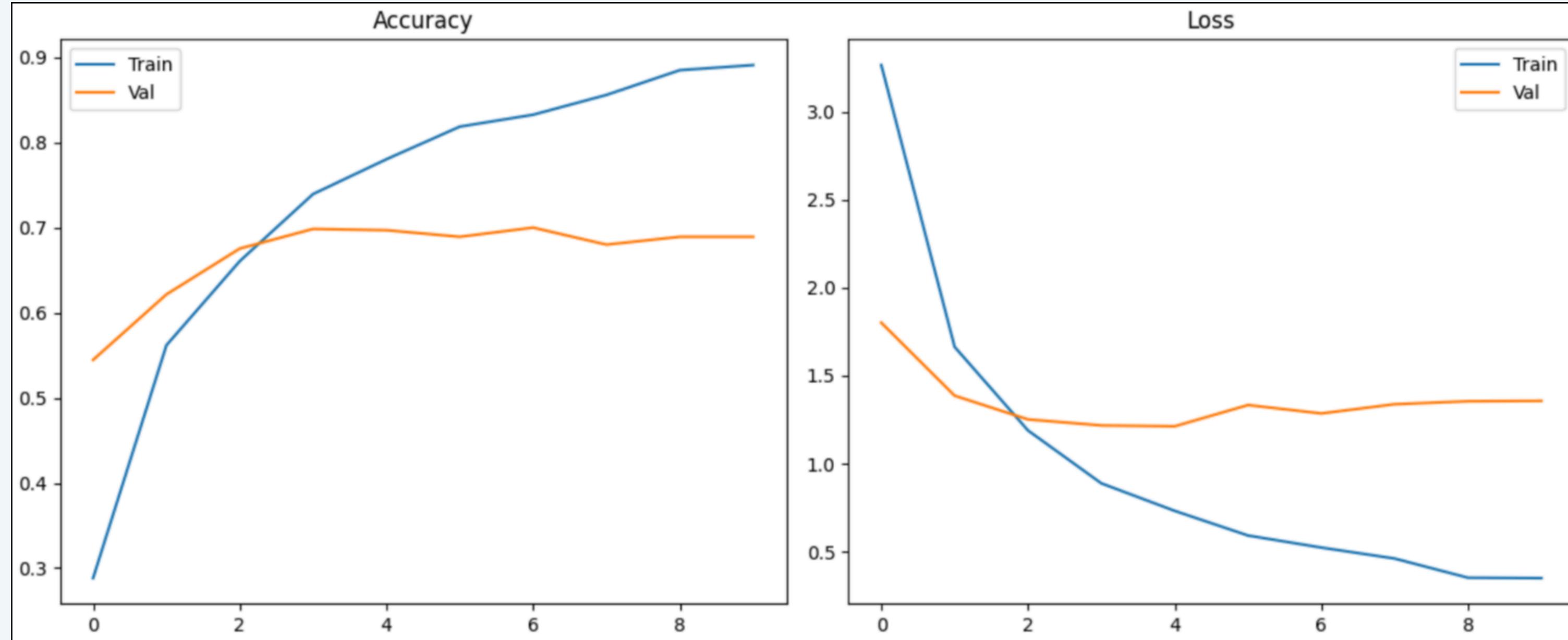
# Hasil ANN Multilayer Perception

	precision	recall	f1-score	support
accuracy	0.152485	0.152485	0.152485	0.152485
macro avg	0.090128	0.145603	0.090574	6499.000000
weighted avg	0.092740	0.152485	0.093136	6499.000000

F1 Score per Class

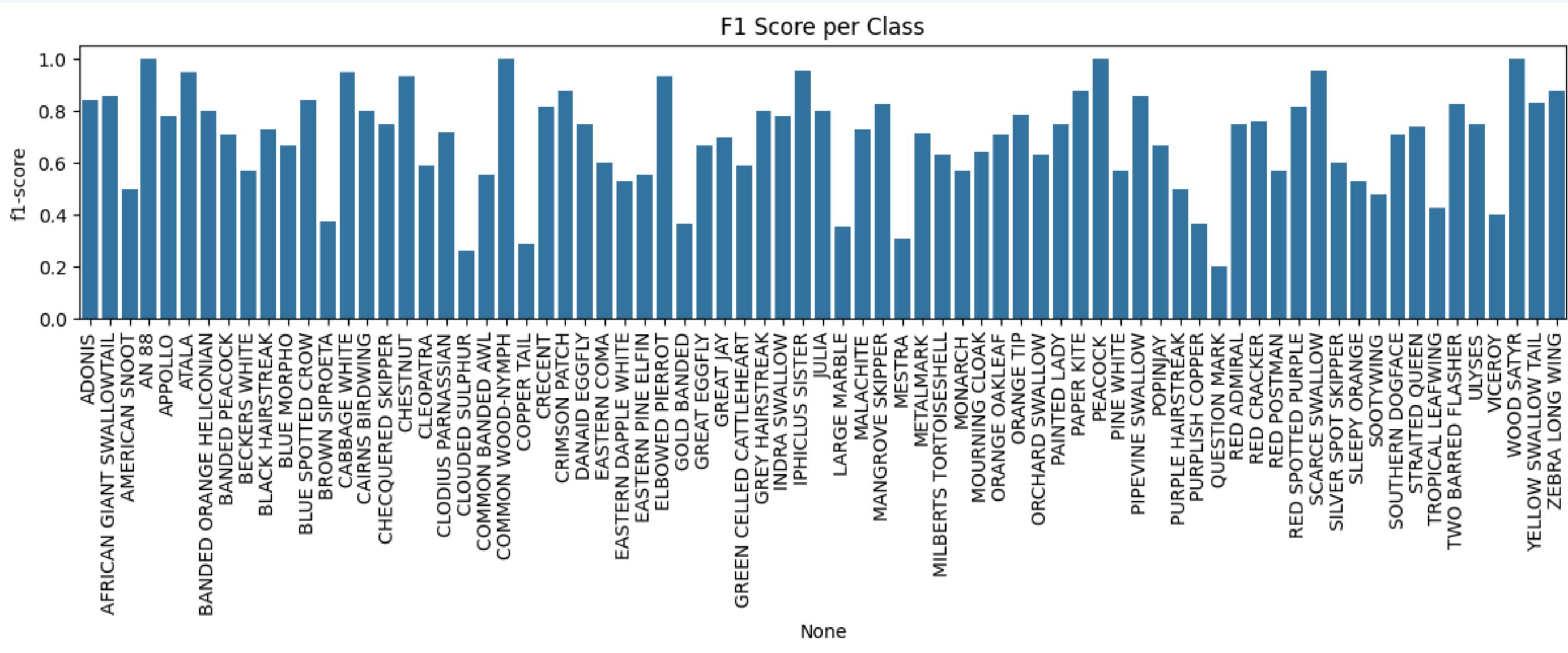


## Hasil ResNet

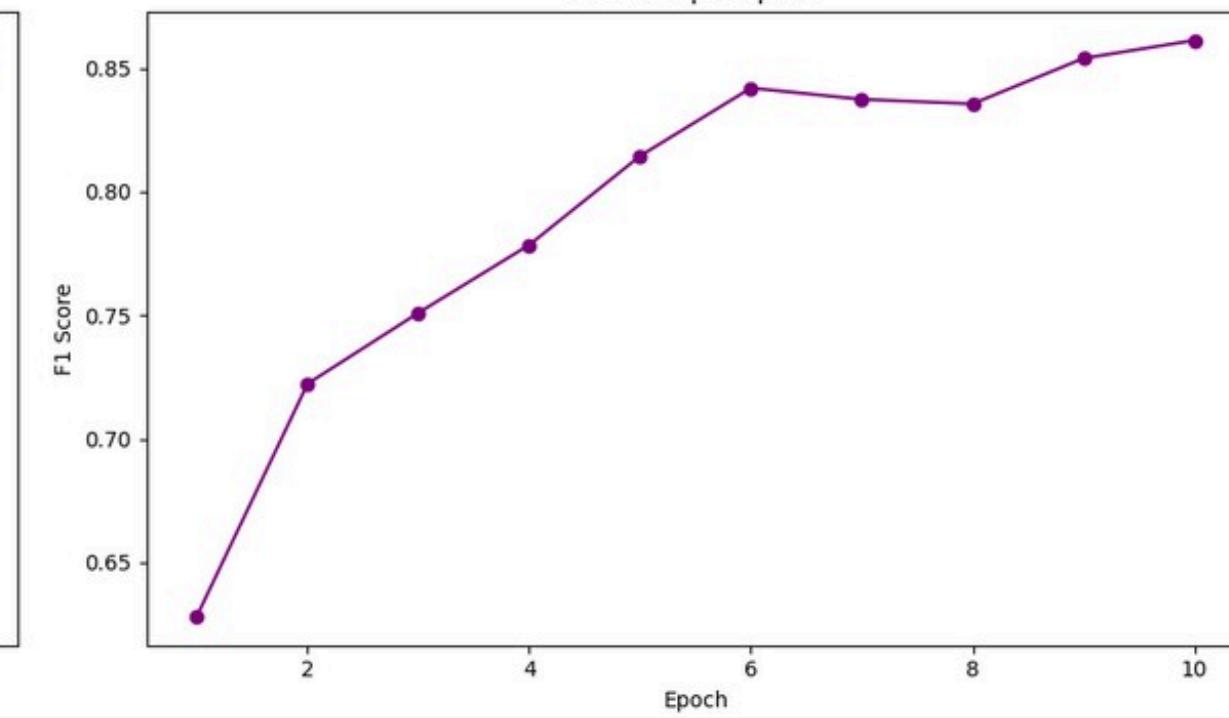
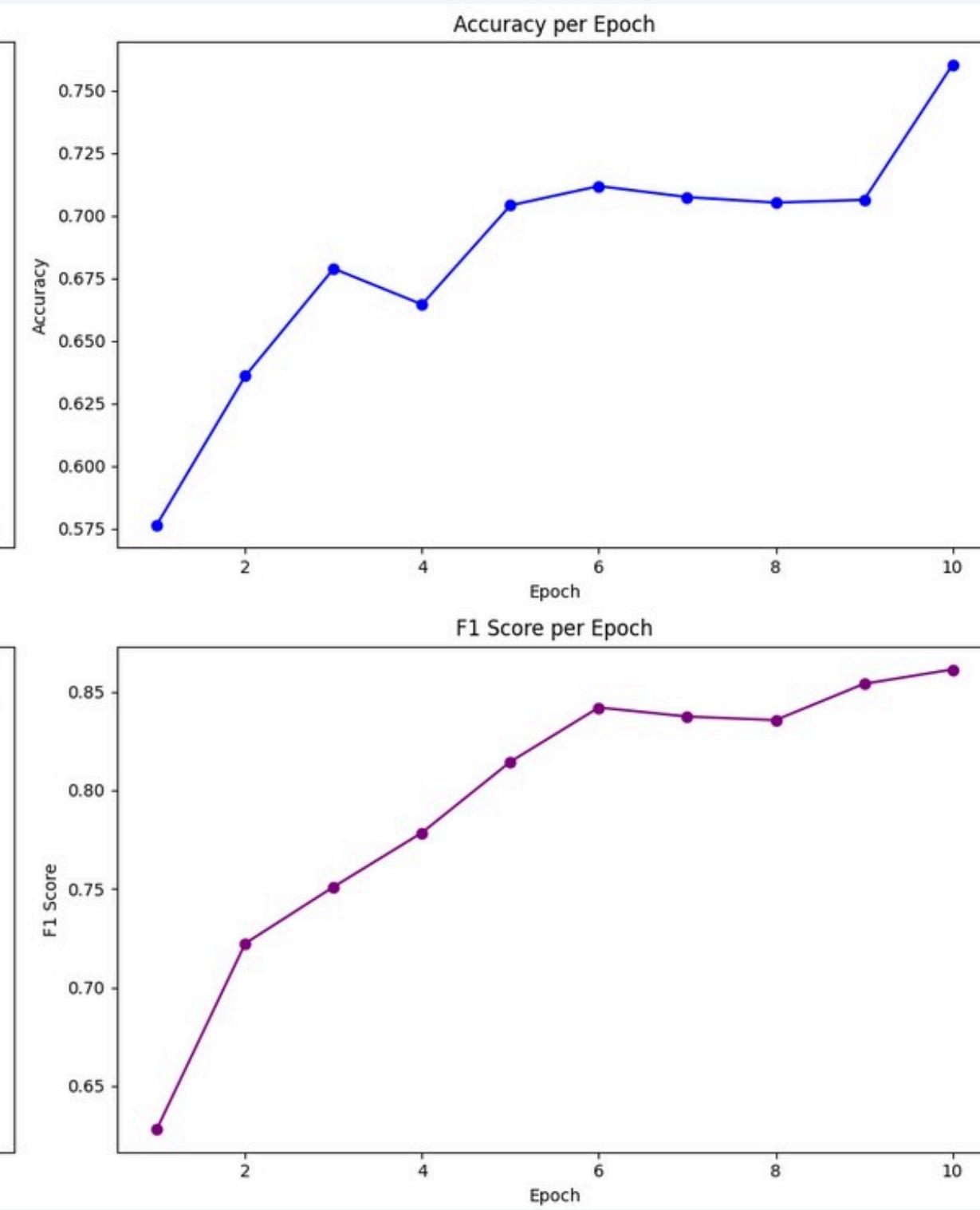
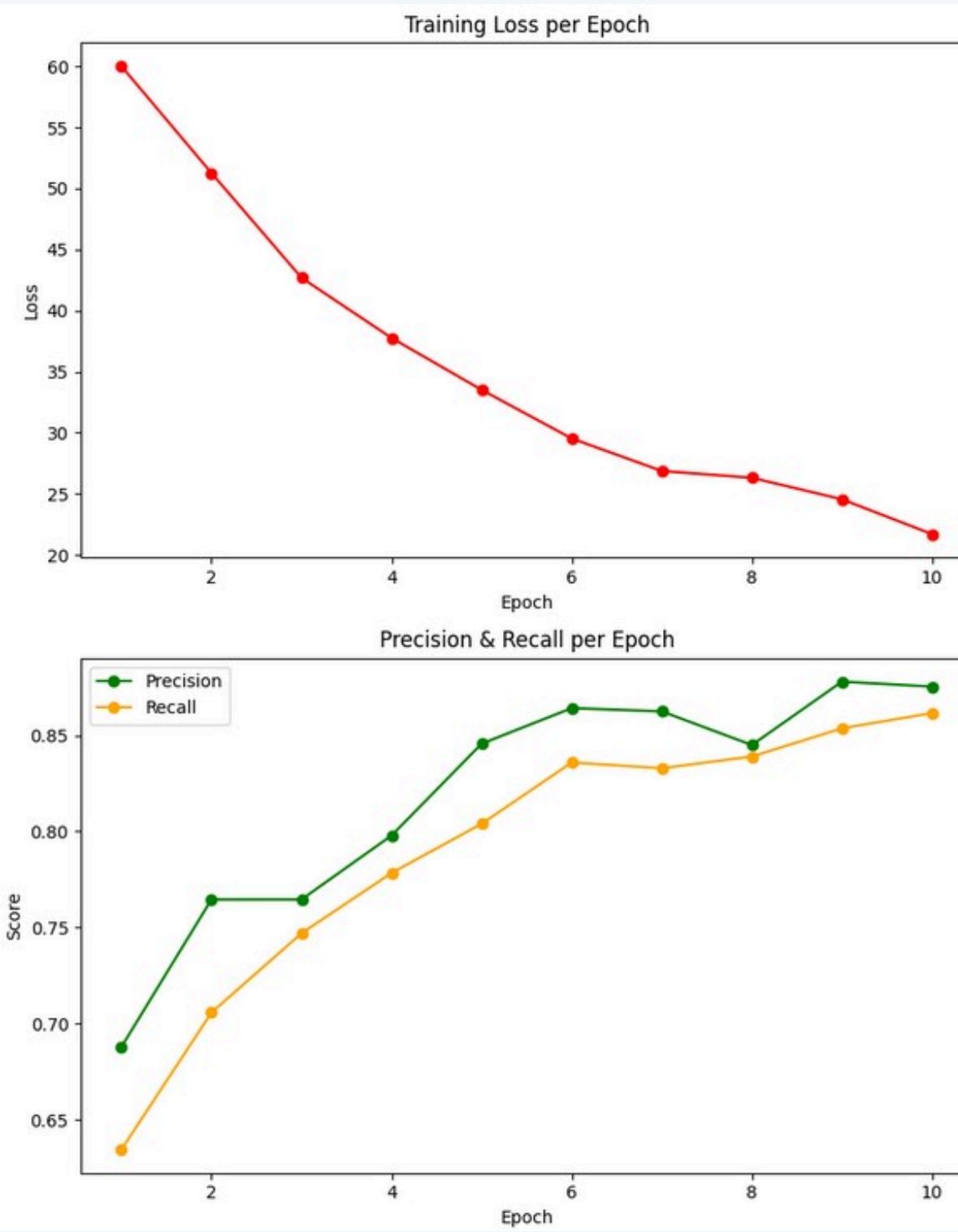


# Hasil ResNet

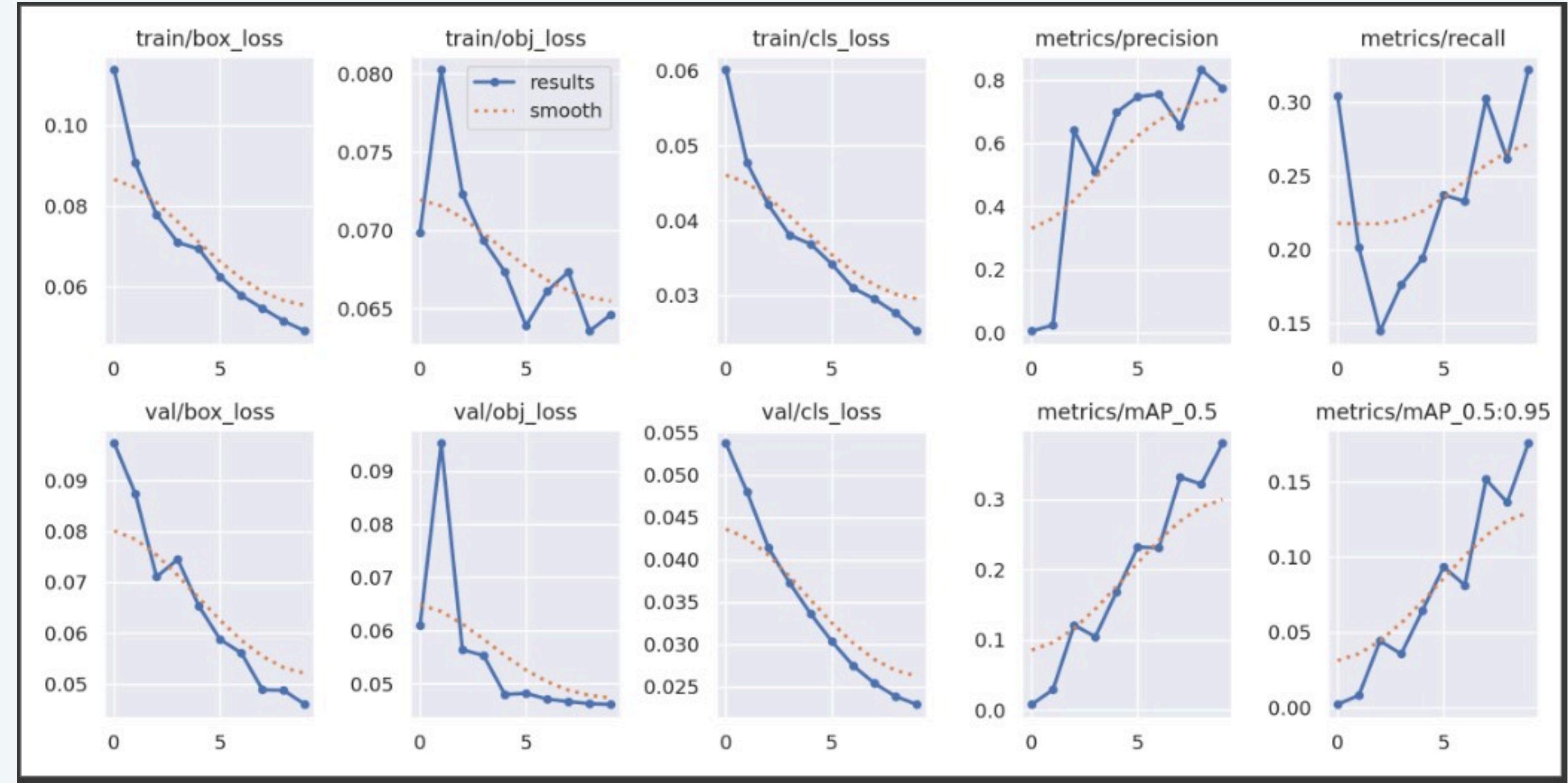
	precision	recall	f1-score	support
accuracy	0.696923	0.696923	0.696923	0.696923
macro avg	0.715828	0.696096	0.691222	650.000000
weighted avg	0.712322	0.696923	0.690253	650.000000



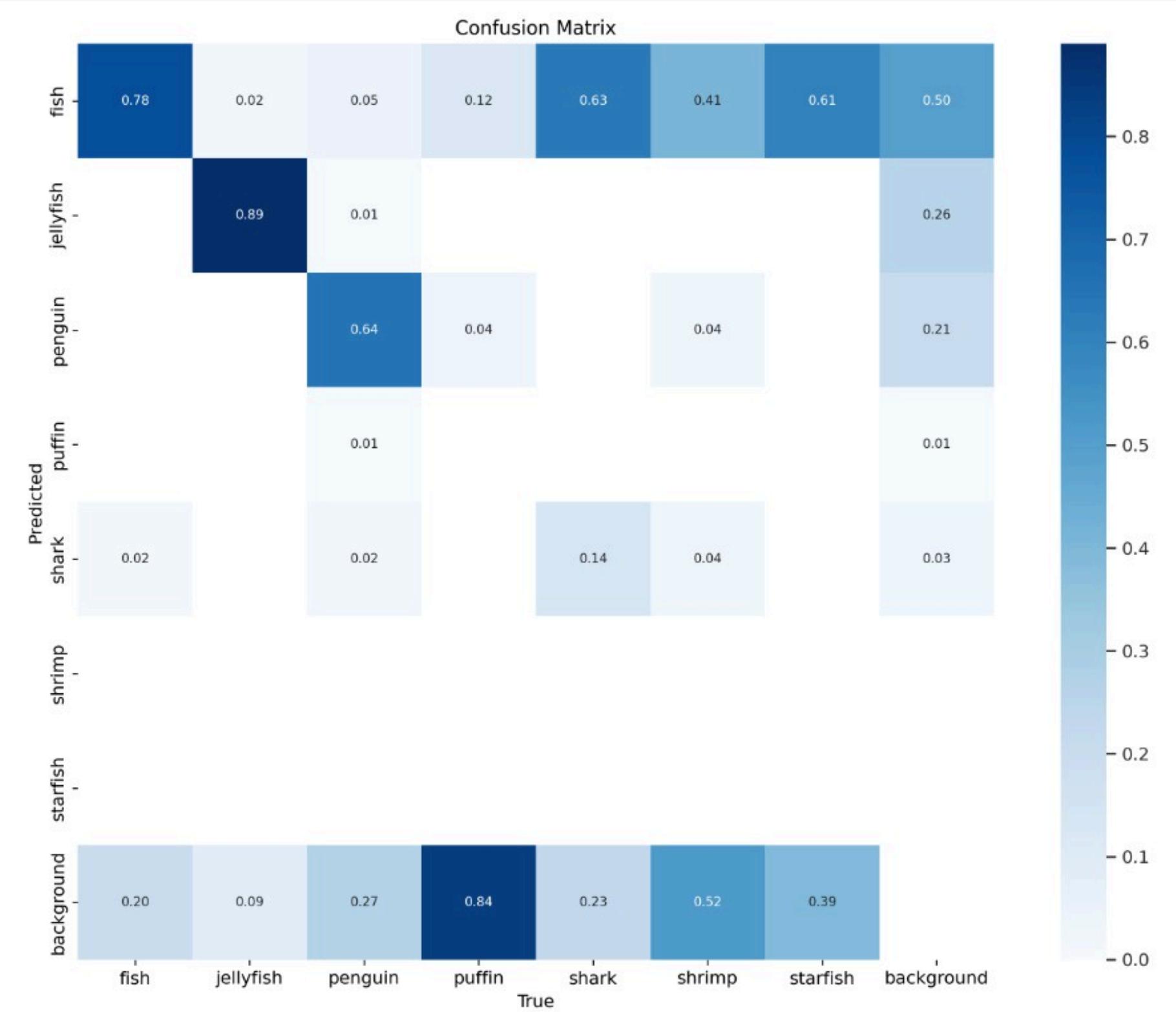
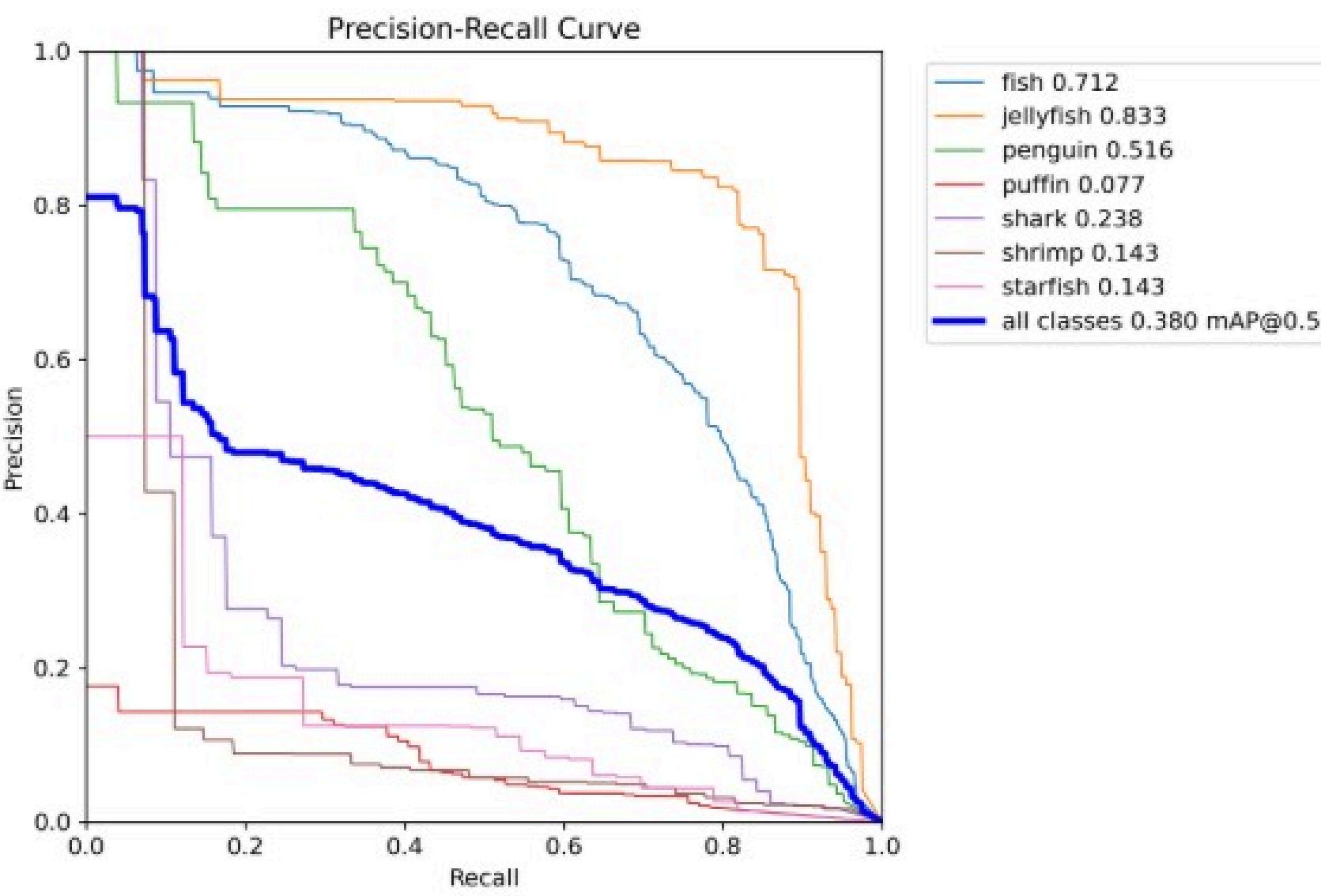
## Hasil Faster R-CNN



# Hasil YOLO



# Matriks YOLO

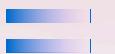




# Progress Report Week - 2

Sihombing Giovano Geraldo (2206059566)  
Yoel Dwi Miryano (2206059534)  
Naufal Rusyda Santosa (2206813353)  
Muhammad Nadhif Fasichul Ilmi (2206813416)

**Kelompok 13**



# Dataset Classification

## Butterfly Image Classification



### Deskripsi

Dataset berupa gambar-gambar .png dari berbagai spesies kupu-kupu. Setiap gambar merepresentasikan satu kelas spesifik dari jenis kupu-kupu.

### Tujuan

Membangun sistem klasifikasi yang dapat mengidentifikasi spesies kupu-kupu berdasarkan warna, bentuk sayap, dan pola unik pada gambar.



## Data Pre-Processing

### 1. Transformasi Gambar (transforms.Compose)

```
transform = transforms.Compose([
    transforms.Resize((64, 64)), # Resize semua gambar
    transforms.ToTensor(),      # Convert ke tensor
    transforms.Normalize(mean=[0.5]*3, std=[0.5]*3) # Normalisasi RGB
])
```

### 2. Label Encoding (LabelEncoder)

```
label_encoder = LabelEncoder()
train_df['label'] = label_encoder.fit_transform(train_df['label'])
```

### 3. Train-Validation Split (train\_test\_split)

```
train_data, val_data = train_test_split(train_df, test_size=0.2, stratify=train_df['label'], random_state=42)
```

# ANN Multilayer Perceptron

## Arsitektur MLP

1. Dimensi Input

2. Hidden Layer

3. Aktivasi ReLU

4. Dropout

5. Output Layer

```
input_size = 64 * 64 * 3

class MLP(nn.Module):
    def __init__(self, input_size, hidden1=512, hidden2=256, num_classes=75):
        super(MLP, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(input_size, hidden1),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(hidden1, hidden2),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(hidden2, num_classes)
        )

    def forward(self, x):
        return self.model(x)
```



## Loss Function dan Optimizer

**1. CrossEntropyLoss**

**2. Adam**

**3. Epoch number**

```
criterion = nn.CrossEntropyLoss()  
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)  
num_epochs = 30
```

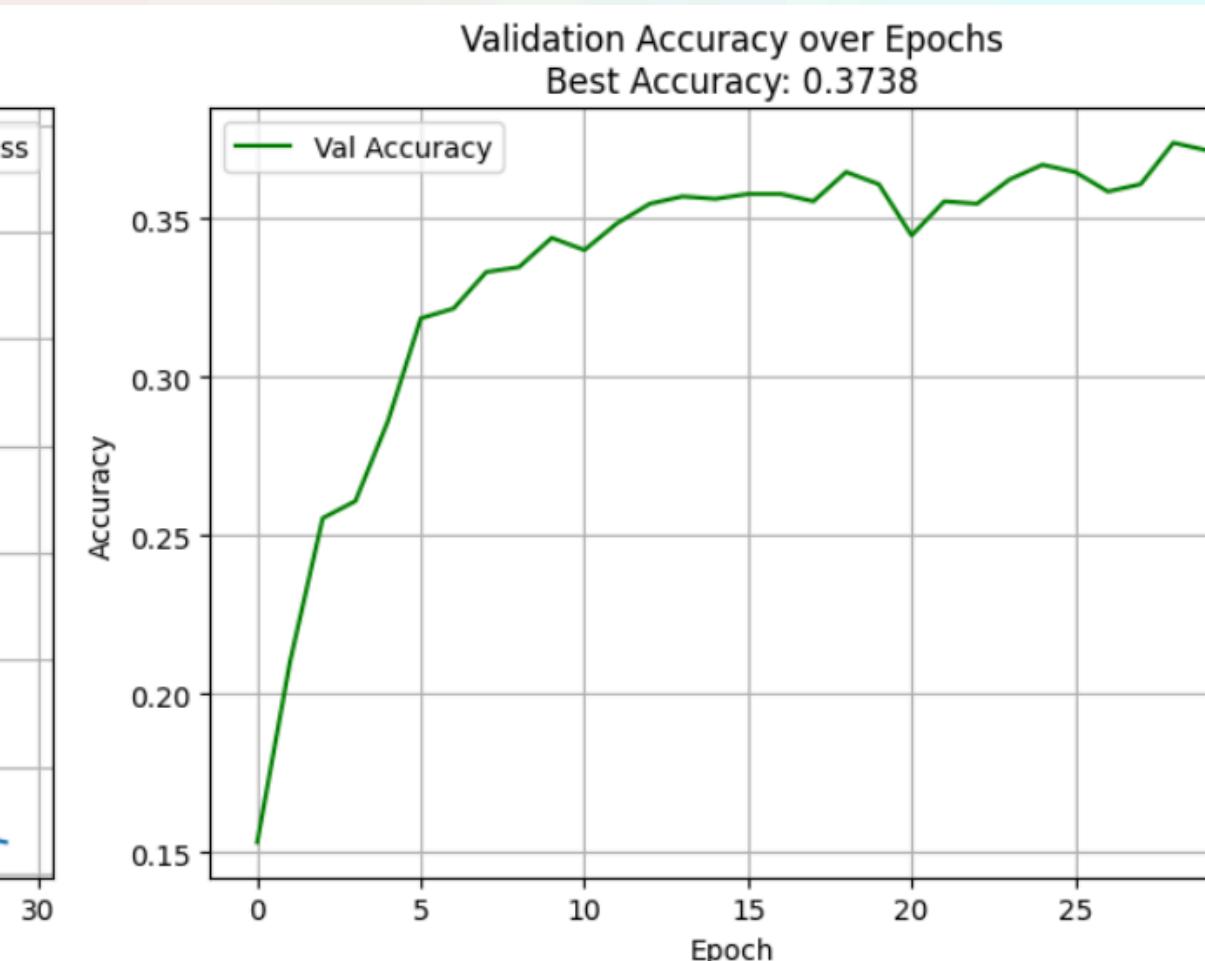
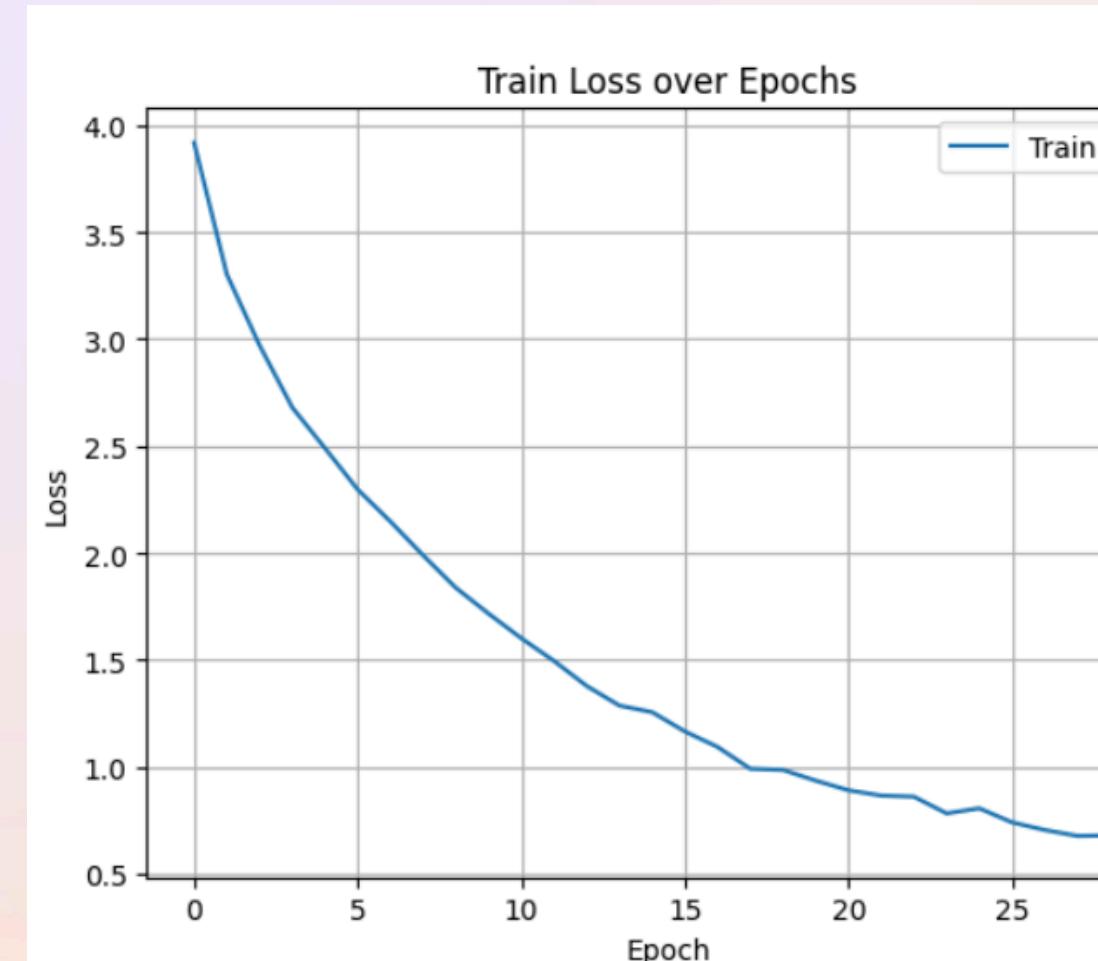
# ANN Multilayer Perceptron



# Result

Final Evaluation on Validation Set:  
Best Validation Accuracy: 0.3738

accuracy	0.37	1300
macro avg	0.37	1300
weighted avg	0.37	1300



# ANN Multilayer Perceptron



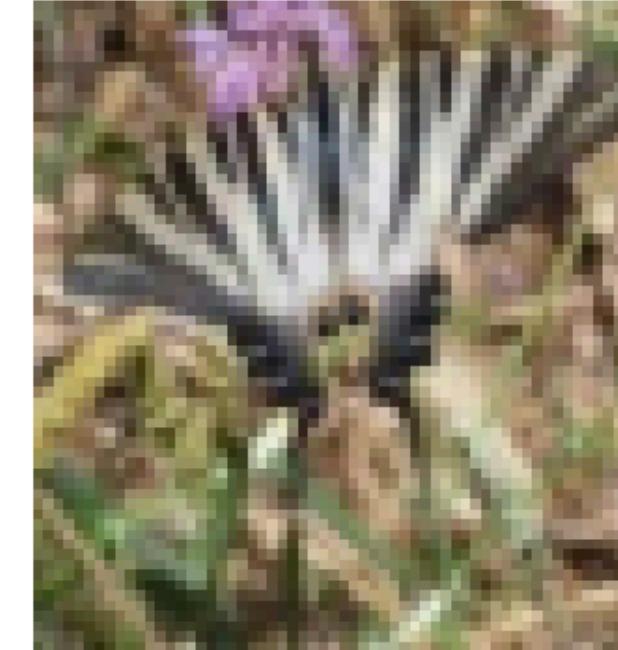
## Sample Test

Sample Predictions on Test Set

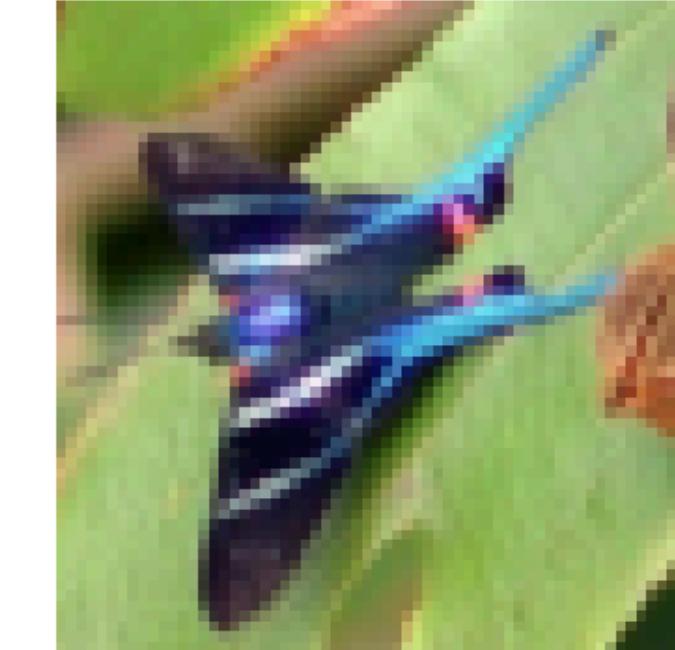
Pred: PEACOCK



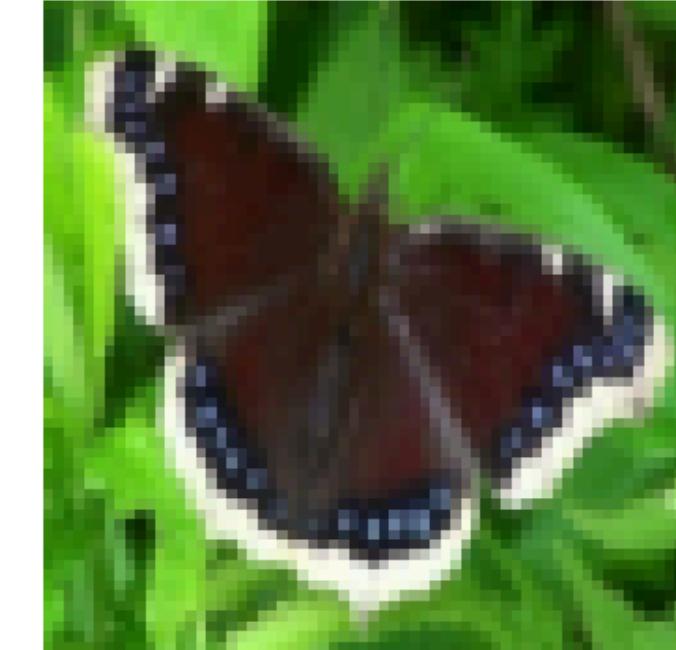
Pred: EASTERN DAPPLE WHITE



Pred: METALMARK



Pred: SOOTYWING



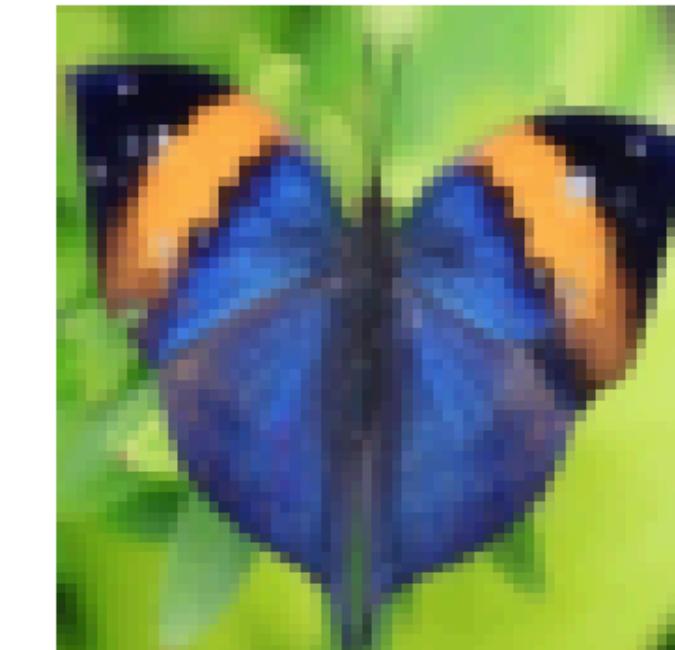
Pred: BROWN SIPROETA



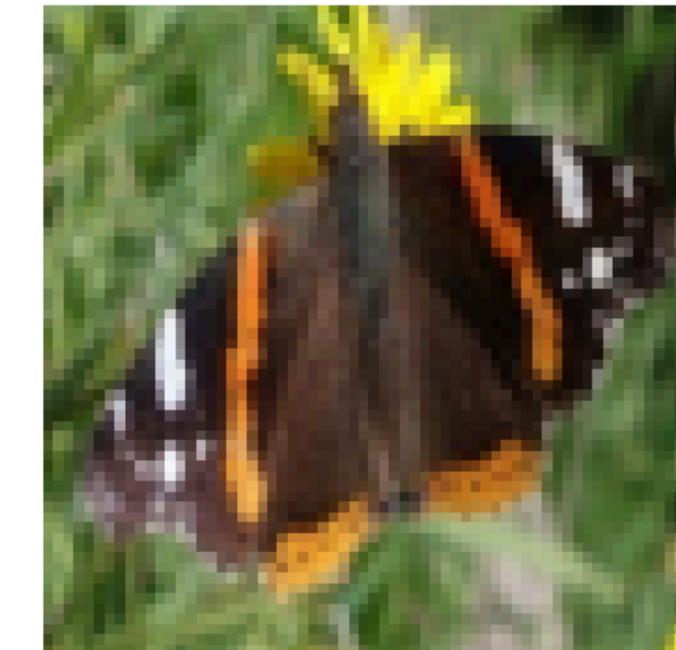
Pred: RED CRACKER



Pred: ORANGE OAKLEAF



Pred: BLUE SPOTTED CROW





## AlexNet

```
Training data loaded: (6499, 64, 64, 3)  
Unique labels: 75
```

Terdapat 6499 gambar pelatihan yang telah dimuat ke dalam model, masing-masing sudah diubah ukurannya menjadi  $64 \times 64$  piksel dengan 3 channel warna (RGB).

```
Final testing data: (1300, 64, 64, 3)
```

Untuk evaluasi, 1300 gambar digunakan sebagai data uji. Data ini dipisahkan dari data pelatihan dan dipakai untuk mengukur performa akhir model (misalnya akurasi dan top-k accuracy).

```
Image size: (64, 64)  
Batch size: 64  
Epochs trained: 30
```

```
# Normalize pixel values to [0, 1]  
img = img.astype(np.float32) / 255.0
```

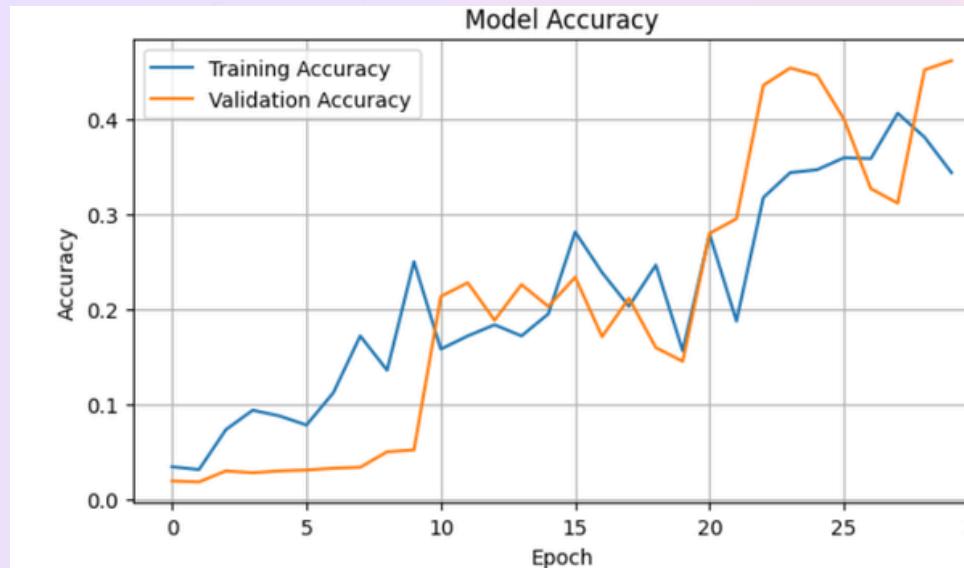
```
def create_alexnet(input_shape, num_classes):  
    model = models.Sequential([  
        layers.Conv2D(64, (7, 7), strides=2, activation='relu', padding='same'),  
        layers.BatchNormalization(),  
        layers.MaxPooling2D((2, 2), strides=2),  
        ...  
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),  
        layers.BatchNormalization(),  
        layers.MaxPooling2D((2, 2), strides=2),  
        layers.Flatten(),  
        layers.Dense(1024, activation='relu'),  
        layers.Dropout(0.5),  
        layers.Dense(512, activation='relu'),  
        layers.Dropout(0.5),  
        layers.Dense(num_classes, activation='softmax')  
    ])
```

Arsitektur AlexNet yang digunakan dalam proyek ini merupakan versi modifikasi untuk gambar berukuran kecil ( $64 \times 64$  piksel), terdiri dari lima lapis konvolusi (Conv2D) dengan kombinasi batch normalization dan max pooling untuk ekstraksi fitur dan reduksi dimensi. Setelah fitur diekstrak, data diratakan (flatten) dan diproses melalui dua lapis fully connected (Dense) masing-masing berukuran 1024 dan 512 neuron dengan dropout 50% untuk mencegah overfitting. Terakhir, output layer menggunakan softmax dengan 75 neuron untuk mengklasifikasikan gambar ke dalam 75 kelas kupu-kupu. Optimasi dilakukan menggunakan Adam dengan loss function categorical crossentropy.

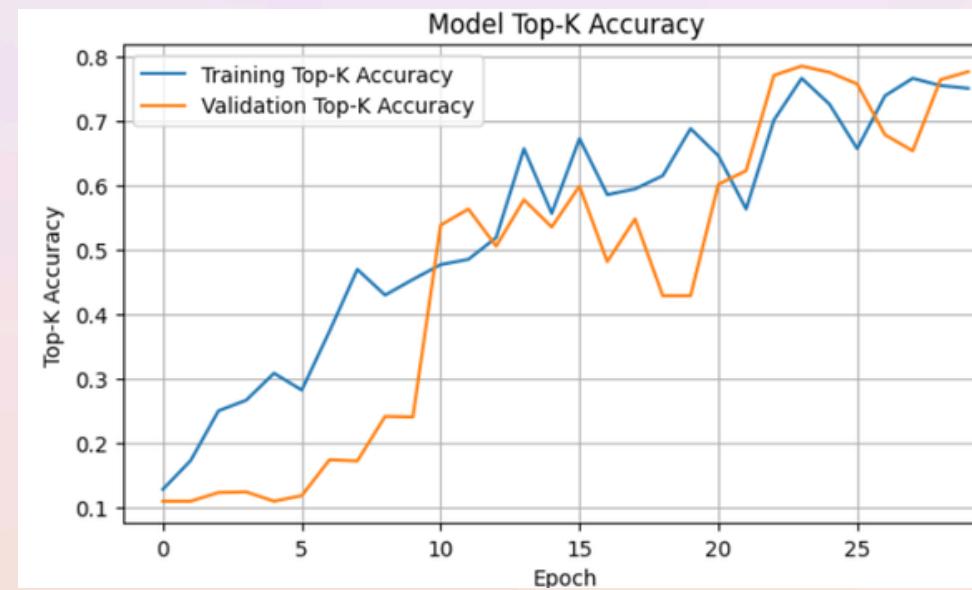
```
Total params: 4,059,467 (15.49 MB)  
Trainable params: 4,057,803 (15.48 MB)  
Non-trainable params: 1,664 (6.50 KB)
```

# Arsitektur CNN

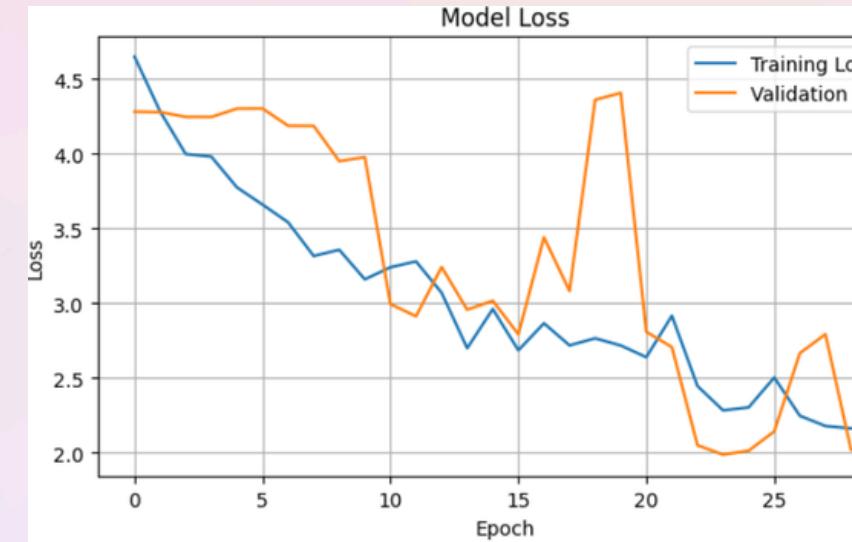
## AlexNet



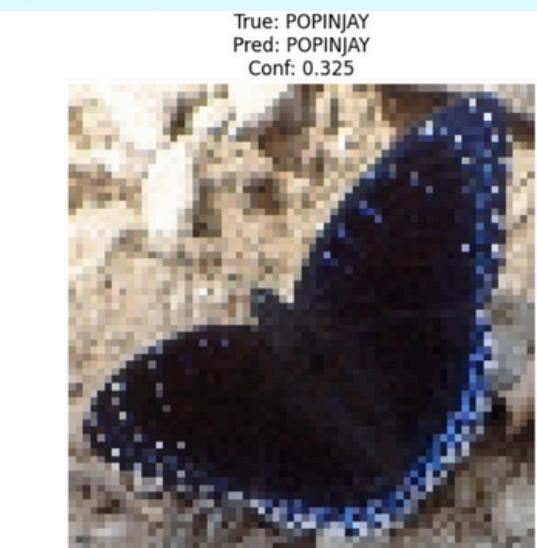
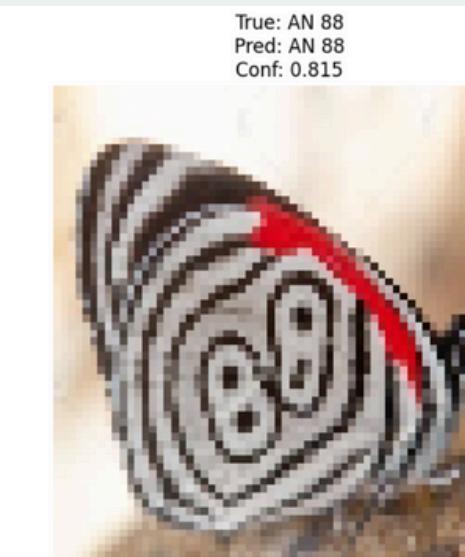
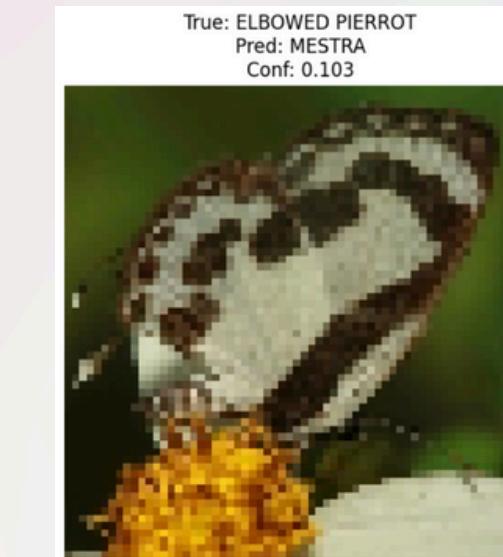
Grafik akurasi menunjukkan perkembangan performa model selama 30 epoch, di mana terjadi peningkatan bertahap terutama setelah epoch ke-10. Menariknya, akurasi validasi pada beberapa titik justru lebih tinggi dibanding akurasi training, yang mengindikasikan bahwa model tidak mengalami overfitting secara signifikan. Di akhir pelatihan, akurasi validasi mencapai sekitar 46%, mencerminkan bahwa model telah belajar mengenali pola dalam data dengan cukup baik.



Grafik top-k accuracy menunjukkan seberapa sering label yang benar masuk dalam lima prediksi teratas model. Sepanjang proses pelatihan, nilai top-k accuracy meningkat secara signifikan hingga mencapai sekitar 77% pada akhir epoch. Hal ini menunjukkan bahwa meskipun prediksi teratas model belum selalu tepat, model sudah cukup baik dalam mengenali label yang benar sebagai salah satu dari lima pilihan utama, yang menandakan pemahaman fitur yang semakin membaik terhadap data.



Grafik loss memperlihatkan penurunan nilai kerugian (loss) pada data training dan validasi seiring bertambahnya epoch, yang menunjukkan bahwa model semakin baik dalam mempelajari pola dari data. Meskipun loss training tampak menurun secara stabil, loss pada data validasi menunjukkan fluktuasi yang cukup besar, mengindikasikan adanya ketidakstabilan dalam proses generalisasi. Hal ini bisa disebabkan oleh kompleksitas dataset atau kebutuhan akan regularisasi tambahan. Secara keseluruhan, model menunjukkan kemajuan dalam pembelajaran meskipun validasi loss belum sepenuhnya stabil.



# Arsitektur CNN



## AlexNet

```
DETAILED PERFORMANCE ANALYSIS
=====
Overall Accuracy: 0.4423 (44.23%)

Per-class Performance:
-----
ADONIS | Acc: 0.667 | Samples: 18 | Avg Conf: 0.647
AFRICAN GIANT SWALLOWTAIL | Acc: 0.467 | Samples: 15 | Avg Conf: 0.475
AMERICAN SNOOT | Acc: 0.067 | Samples: 15 | Avg Conf: 0.174
AN 88 | Acc: 0.588 | Samples: 17 | Avg Conf: 0.658
APOLLO | Acc: 0.167 | Samples: 18 | Avg Conf: 0.306
ATALA | Acc: 0.750 | Samples: 20 | Avg Conf: 0.537
BANDED ORANGE HELICONIAN | Acc: 0.650 | Samples: 20 | Avg Conf: 0.441
BANDED PEACOCK | Acc: 0.647 | Samples: 17 | Avg Conf: 0.632
BECKERS WHITE | Acc: 0.125 | Samples: 16 | Avg Conf: 0.487
BLACK HAIRSTREAK | Acc: 0.353 | Samples: 17 | Avg Conf: 0.205
BLUE MORPHO | Acc: 0.133 | Samples: 15 | Avg Conf: 0.368
BLUE SPOTTED CROW | Acc: 0.353 | Samples: 17 | Avg Conf: 0.329
BROWN SIPROETA | Acc: 0.350 | Samples: 20 | Avg Conf: 0.279
CABBAGE WHITE | Acc: 1.000 | Samples: 18 | Avg Conf: 0.771
CAIRNS BIRDWING | Acc: 0.294 | Samples: 17 | Avg Conf: 0.392
CHEQUERED SKIPPER | Acc: 0.579 | Samples: 19 | Avg Conf: 0.339
CHESTNUT | Acc: 0.235 | Samples: 17 | Avg Conf: 0.321
CLEOPATRA | Acc: 0.263 | Samples: 19 | Avg Conf: 0.493
CLODIUS PARNASSIAN | Acc: 0.294 | Samples: 17 | Avg Conf: 0.407
CLOUDED SULPHUR | Acc: 0.167 | Samples: 18 | Avg Conf: 0.440
COMMON BANDED AWL | Acc: 0.765 | Samples: 17 | Avg Conf: 0.298
COMMON WOOD-NYMPH | Acc: 0.111 | Samples: 18 | Avg Conf: 0.208
```

```
Classification Report:
      precision    recall   f1-score   support
   ADONIS       0.80      0.67      0.73      18
AFRICAN GIANT SWALLOWTAIL     0.44      0.47      0.45      15
   AMERICAN SNOOT      0.20      0.07      0.10      15
      AN 88       1.00      0.59      0.74      17
      APOLLO       0.30      0.17      0.21      18
      ATALA       0.65      0.75      0.70      20
BANDED ORANGE HELICONIAN     0.50      0.65      0.57      20
   BANDED PEACOCK      0.44      0.65      0.52      17
   BECKERS WHITE      0.22      0.12      0.16      16
   BLACK HAIRSTREAK      0.75      0.35      0.48      17
   BLUE MORPHO       1.00      0.13      0.24      15
   BLUE SPOTTED CROW      0.18      0.35      0.24      17
   BROWN SIPROETA      0.58      0.35      0.44      20
   CABBAGE WHITE      0.22      1.00      0.36      18
   CAIRNS BIRDWING      0.71      0.29      0.42      17
   CHEQUERED SKIPPER      0.52      0.58      0.55      19
   CHESTNUT       0.57      0.24      0.33      17
   CLEOPATRA       0.36      0.26      0.30      19
CLODIUS PARNASSIAN      0.31      0.29      0.30      17
   CLOUDED SULPHUR      0.21      0.17      0.19      18
   COMMON BANDED AWL      0.19      0.76      0.30      17
   COMMON WOOD-NYMPH      0.20      0.11      0.14      18
   COPPER TAIL       0.67      0.11      0.18      19
      CRECENT       0.58      0.35      0.44      20
```

### Confidence Statistics:

Correct predictions - Mean: 0.546, Std: 0.271  
Wrong predictions - Mean: 0.316, Std: 0.188

Statistik confidence menunjukkan bahwa rata-rata tingkat keyakinan model pada prediksi yang benar adalah 0.546, sedangkan pada prediksi yang salah hanya 0.316. Ini mengindikasikan bahwa model cenderung lebih yakin saat membuat prediksi yang benar dibandingkan ketika salah, yang merupakan tanda positif dalam keandalan prediksi model.

### ALEXNET MODEL SUMMARY

Dataset: Butterfly Image Classification

Architecture: AlexNet CNN

Training samples: 4159

Validation samples: 1040

Testing samples: 1300

Number of classes: 75

Image size: (64, 64)

Batch size: 64

Epochs trained: 30

Final test accuracy: 0.4423 (44.23%)

Final test top-k accuracy: 0.7692 (76.92%)

Total parameters: 4,059,467

# Dataset Object Detection

## Aquarium Object Detection



### Deskripsi

Dataset berisi gambar berbagai objek bawah laut seperti ikan, bintang laut, dan terumbu karang dengan anotasi bounding box.

### Tujuan

Membangun sistem deteksi objek untuk mengenali dan memberikan bounding box pada objek laut dalam gambar.

# Faster R-CNN

adalah salah satu arsitektur deep learning yang digunakan untuk tugas object detection, yaitu mengenali dan memberi kotak pembatas pada objek-objek tertentu di dalam sebuah gambar. Model ini merupakan pengembangan dari pendahulunya, R-CNN dan Fast R-CNN, dengan keunggulan utama terletak pada kecepatannya dalam mendeteksi objek.

Model berhasil mencapai mAP@0.5 sebesar 0.6176, menunjukkan performa deteksi objek yang cukup baik. Kelas dengan hasil terbaik adalah starfish dan jellyfish dengan precision > 0.89 dan IoU > 0.72. Sebaliknya, puffin dan penguin menunjukkan performa lebih rendah, kemungkinan karena kemiripan visual atau data terbatas. Secara keseluruhan, model menunjukkan generalisasi yang baik dan versi terbaik telah disimpan.

This dataset consists of 638 images

Class-wise Performance:

fish: Avg IoU = 0.640, Precision = 0.800  
shark: Avg IoU = 0.724, Precision = 0.891  
puffin: Avg IoU = 0.518, Precision = 0.661  
penguin: Avg IoU = 0.551, Precision = 0.681  
starfish: Avg IoU = 0.787, Precision = 0.952  
stingray: Avg IoU = 0.735, Precision = 0.871  
jellyfish: Avg IoU = 0.722, Precision = 0.899

Overall mAP@0.5: 0.6176

Best model saved! mAP: 0.6176

Training completed! Final mAP: 0.6176

## TRAINING SUMMARY

Dataset: Underwater Animals (7 classes)

Classes: fish, jellyfish, penguin, puffin, shark, starfish, stingray

Training samples: 448

Validation samples: 127

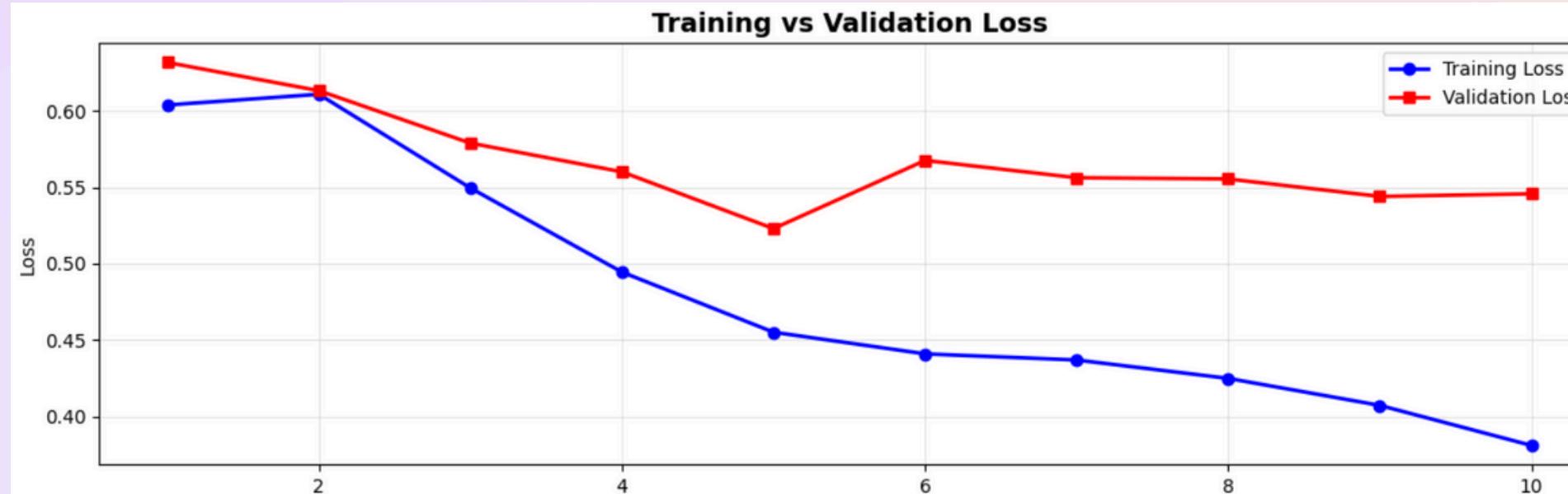
Final Training Loss: 0.3810

Final Validation Loss: 0.5457

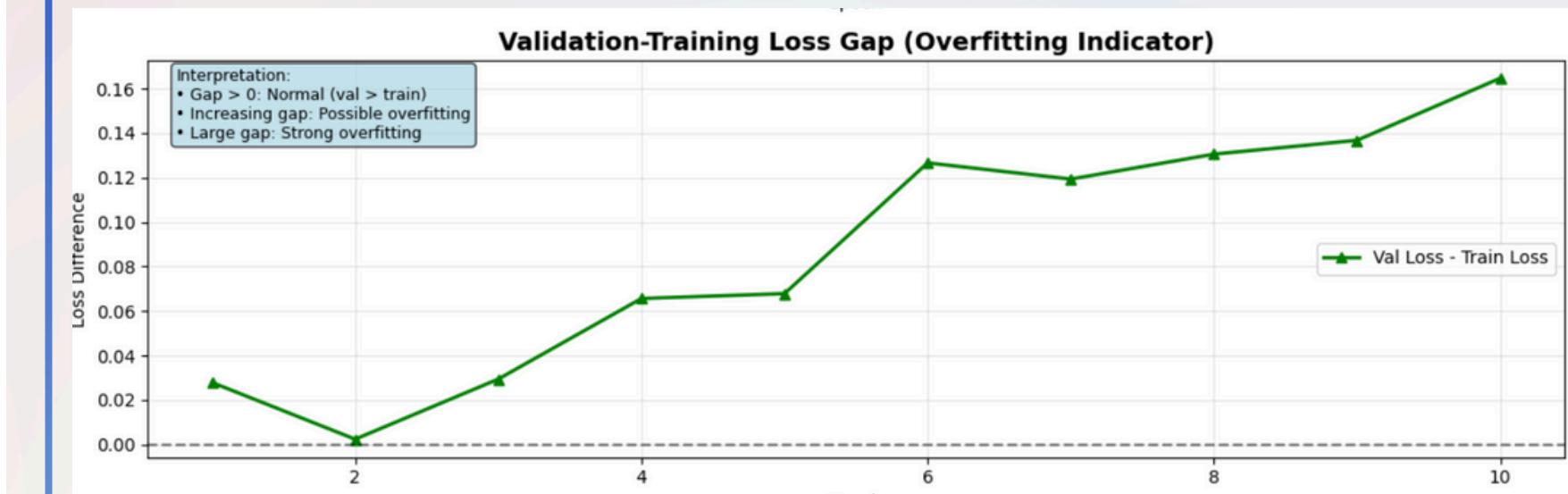
Final mAP: 0.6176

Best mAP: 0.6176

# Faster R-CNN



Grafik menunjukkan perbandingan training loss dan validation loss selama 10 epoch. Terlihat bahwa training loss terus menurun secara konsisten, menandakan bahwa **model mampu belajar dari data latih**. Namun, validation loss mengalami stagnasi dan fluktuasi kecil setelah epoch ke-5, meskipun secara umum menurun sedikit. Ini mengindikasikan bahwa **model mulai mengalami gejala overfitting ringan**, yaitu model terlalu menyesuaikan diri pada data latih sehingga tidak meningkat secara signifikan pada data validasi.



Grafik menunjukkan tren overfitting yang semakin kuat sejak epoch ke-3, ditandai dengan peningkatan gap antara validation loss dan training loss. Di akhir pelatihan, gap melebihi 0.16, mengindikasikan model terlalu menyesuaikan diri pada data latih dan kehilangan kemampuan generalisasi.

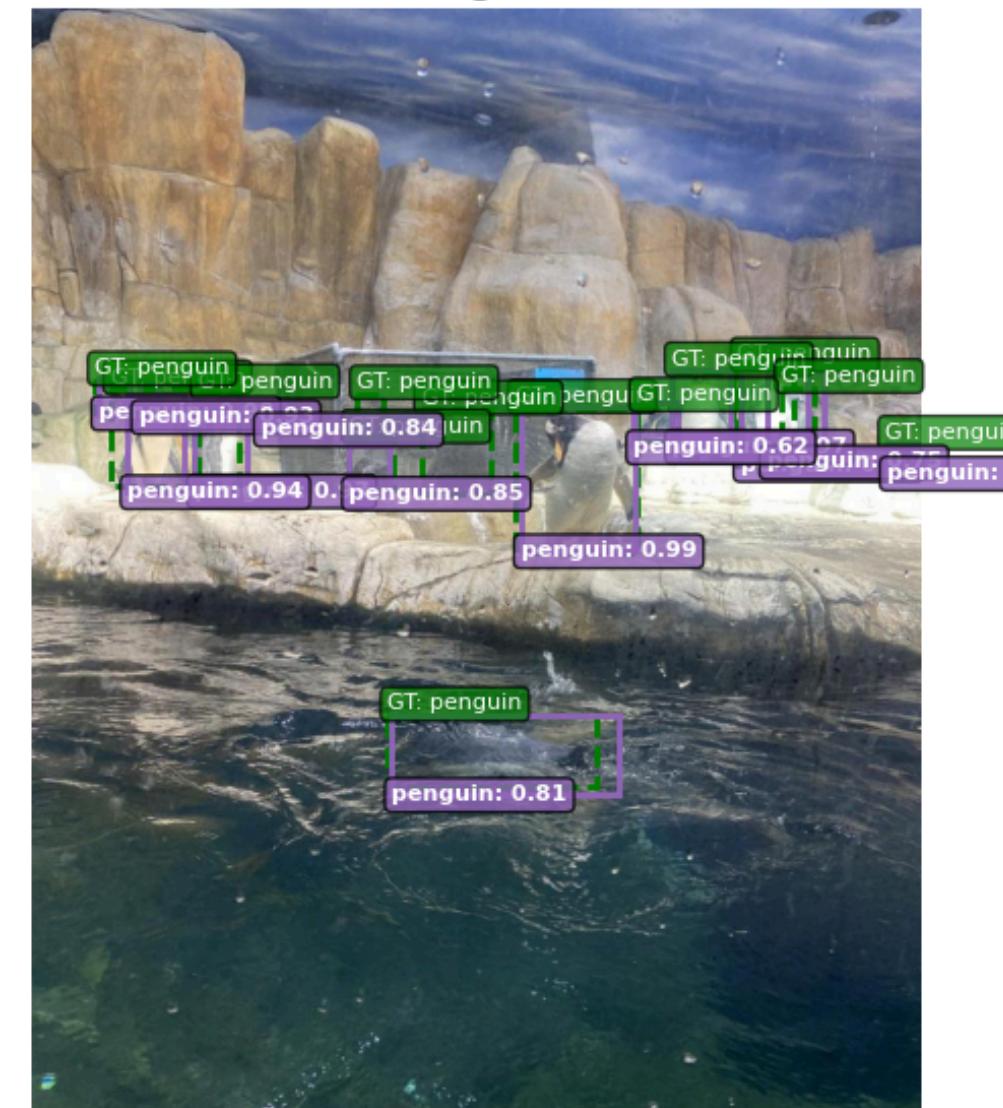
# Faster R-CNN

## Faster R-CNN Predictions on Underwater Animal (Green=Ground Truth, Colored=Predictions) Image 7

Image 88



Image 90



# Faster R-CNN

Image 51

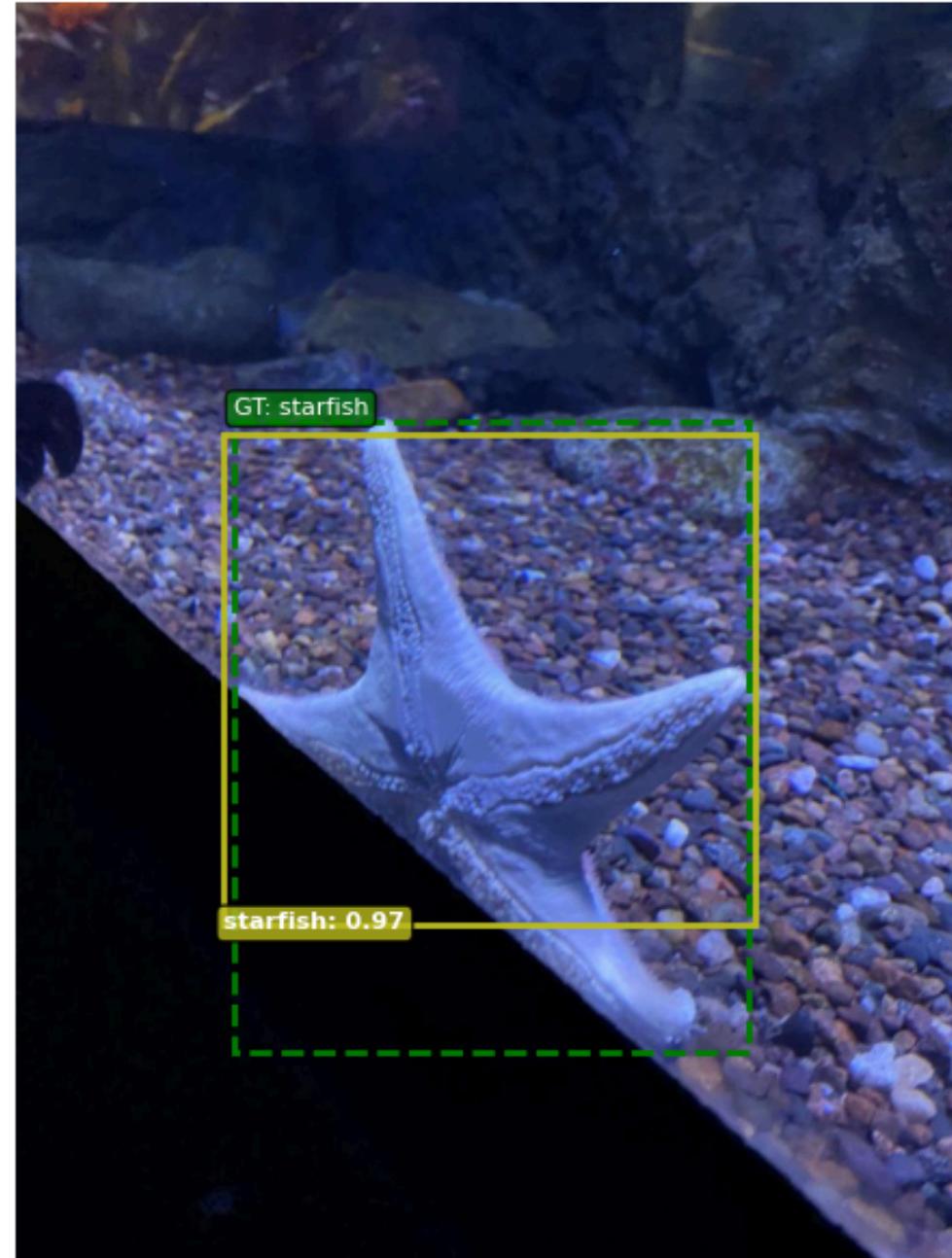


Image 105

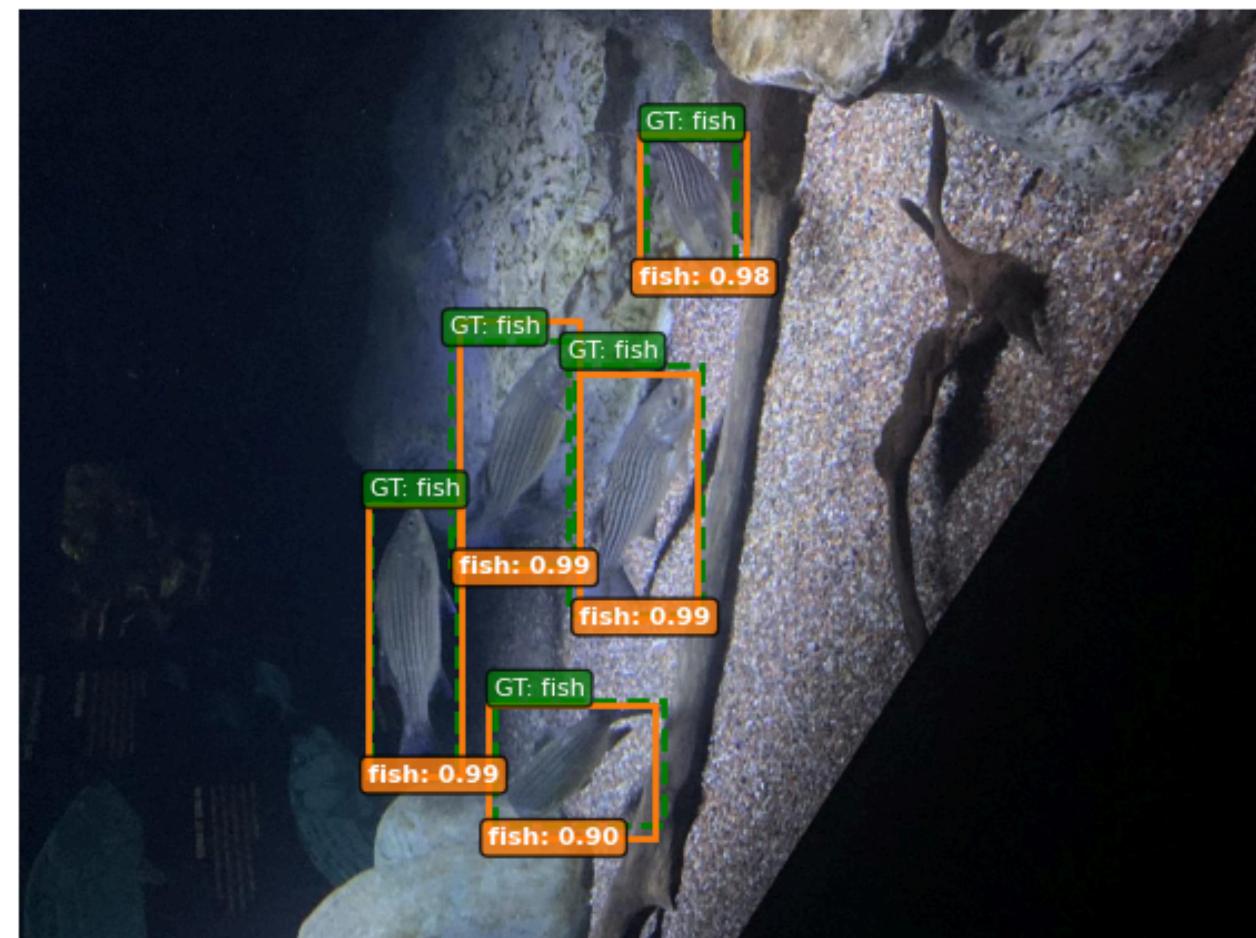
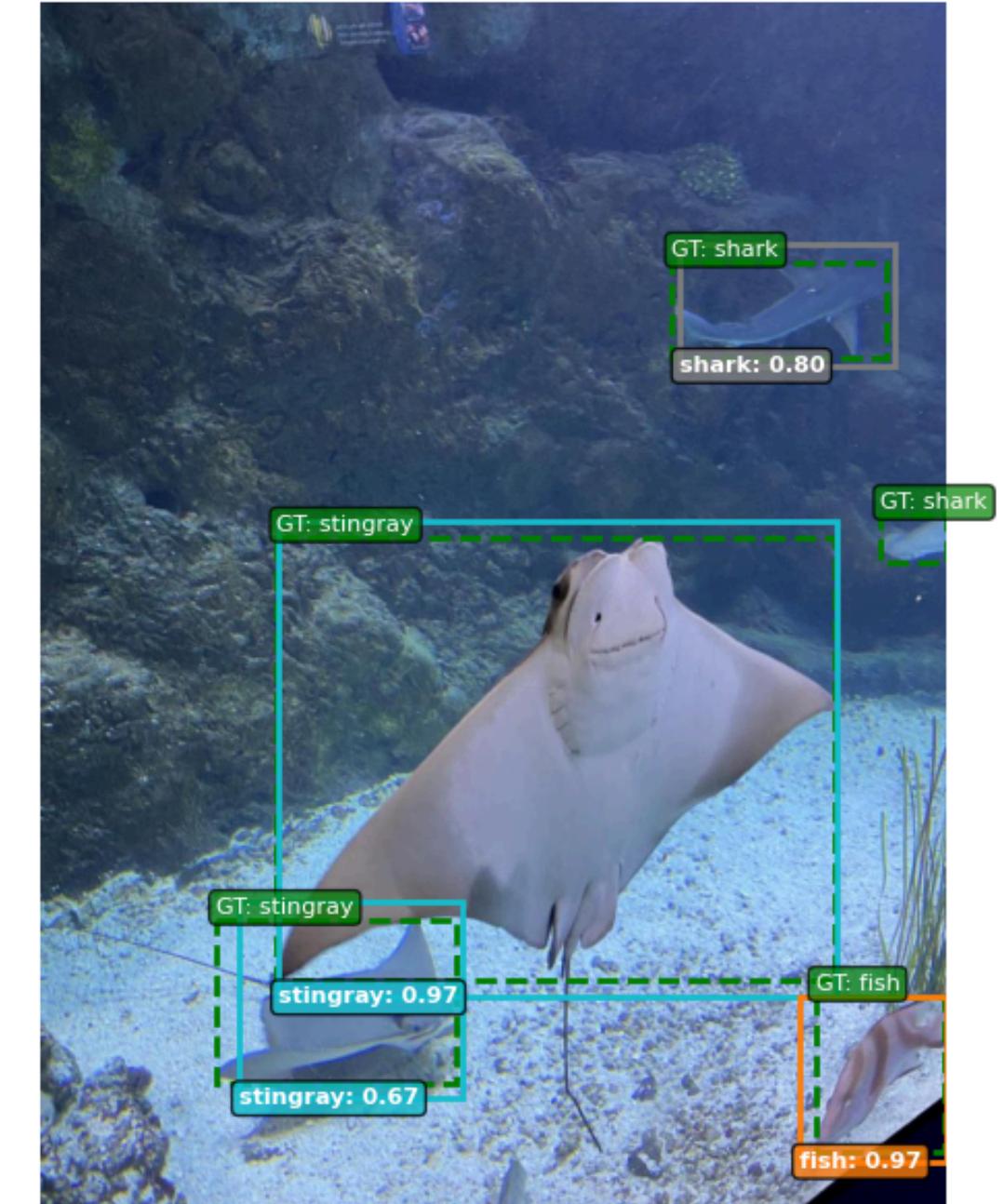


Image 68



Thank You.