OBJECT DETECTION USING OPEN IMAGES DATASET

ASSIGNMENT-1

CS6482 - DEEP REINFORCEMENT LEARNING

J.J. COLLINS

AAYUSH THAKAR (24041785)

DEEPANKUR LOHIYA (24008818)

TABLE OF CONTENT

1. INTRODUCTION

CNNs represent a deep learning technique which dominates the image recognition and classification and object detection sector. The CNN architecture contains convolutional layers that find spatial patterns before using pooling layers for dimension reduction and fully connected layers for producing a result. Due to their competence in discovering hierarchical patterns CNNs become highly efficient for computer vision applications which include autonomous driving and medical imaging and surveillance systems.

The assignment investigated the features of YOLOv8 and Inception module as applied to object detection scenarios. Our execution failed when we tried to combine YOLOv5 with YOLOv8 so we decided to directly assess YOLOv8 and Inception. The experiments performed detection accuracy assessment through multiple tests that included hyperparameter adjustments and augmentation methods and performance testing protocols. The evaluation analysis used precision and recall measures along with F1-score to find the advantages and weaknesses of both approaches.

2. HARDWARE REQUIREMENTS

Our CNN-based object detection models needed an efficiently powerful system which also offered ample memory to execute training and testing processes effectively. The model needed a multi-core processor between Intel i7 and AMD Ryzen 7 models to perform efficiently during data processing and execution of models. YOLOv8 and Inception both demand complex matrix operations so the use of a dedicated GPU with at least GTX 1650 was essential for parallel training processing. The system required at least 16GB of Random Access Memory to operate effectively during training since it needed this capacity for loading big datasets while conducting image augmentation and preventing memory-related obstacles. These hardware specifications delivered model convergence and cut down training time and delivered better performance during applications with deep learning tasks involving high-resolution images.

3. OVERVIEW

Computer vision applications need object detection which determines both object classification alongside exact localization inside images. We worked on detecting five classes composed of Cat and Dog and Fish and Lion and Tiger by utilizing Open Images dataset. The project analyzed both YOLOv8 as an object detection model able to perform real-time operations along with the Inception

module which displays excellence in extracting multi-scale features from images. The detection method in YOLOv8 operates without anchors and executes inference quickly and the Inception module functions through its multiple convolutional layers to identify precise features. We used precision, recall and F1-score to evaluate the model effectiveness of YOLOv8 compared to Inception for detecting four categories of animals within the data set. The experimental results demonstrate key capabilities and drawbacks that enabled the improvement of detection techniques for practical implementation.

3.a YOLO OVERVIEW

YOLO (You Only Look Once) represents a fast real-time object detection system which transformed the field when it treated detection as a regression problem to generate both bounding boxes and class probabilities during a single forward pass. Ultralytics unveiled YOLOv8 as the latest iteration of their detection system which advances all performance aspects including precision and processing speed and adaptability. The model performs multiple operations such as object detection together with instance segmentation and image classification for different practical uses. YOLOv8 provides a smooth and expandable user experience which operates across a wide spectrum of edge systems to high-end GPU setups. Real-time operation combined with high precision accuracy makes YOLOv8 work optimally in surveillance systems and autonomous driving systems and robotic applications.

The YOLO (You Only Look Once) system delivers deep learning model optimization specifically for deployment conditions of limited resources. YOLOv8 demonstrates remarkable detection performance but YOLO supports these models after training by applying quantization together with pruning and optimizations for specific hardware systems. Such performance optimizations allow models to execute using reduced resource usage and minimal latency which enables them to operate effectively on mobile phones and embedded systems. YOLOv8 and YOLO operate as a complementary pair where YOLOv8 gives superior detection results and YOLO makes the models ready for effective deployment through practical optimization methods. The joint offerings of YOLOv8 and YOLO provide organizations with an extensive solution that supports high-performance object detection system development and deployment capabilities.

3.b Inception Module overview

Inception represents a shift from traditional object detection methods through its GoogLeNet architecture, which integrates multiple feature extraction scales.

Unlike YOLO, which operates as a single-stage detector, Inception employs parallel 1x1, 3x3, and 5x5 convolutional layers to detect objects at different scales simultaneously. This approach allows it to recognize small and large objects effectively while optimizing computational efficiency using 1x1 convolutions for dimensionality reduction.

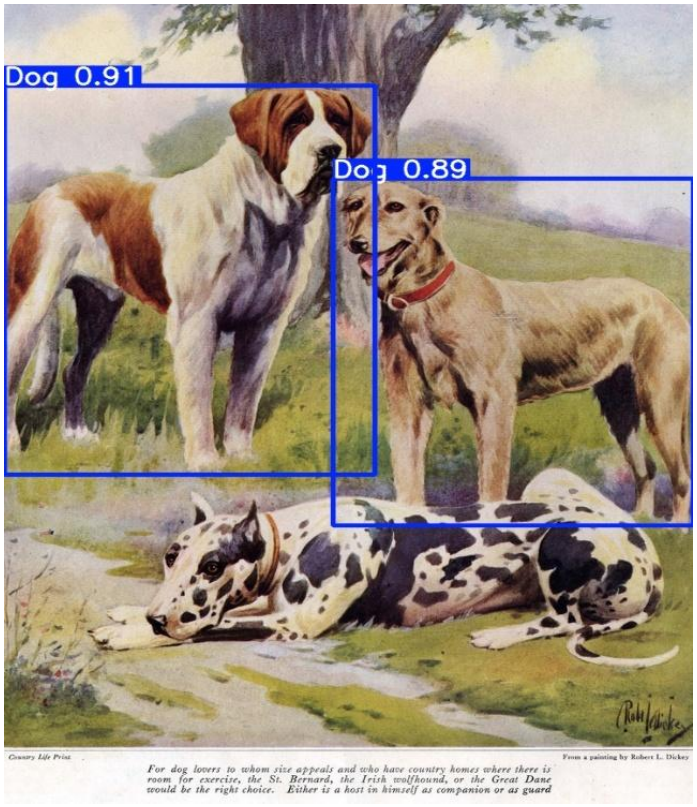
The main benefits of Inception include:

- Parallel convolution for capturing fine-to-coarse details while maintaining accuracy.
- Strong generalization on complex images with varied object sizes and orientations.

However, YOLOv8 outperformed the Inception model in our project due to its faster and more efficient single-stage detection approach. While Inception is more suitable for high-precision tasks like medical imaging and satellite analysis, YOLOv8 demonstrated superior real-time performance with higher accuracy in detecting Cat, Dog, Fish, Lion, and Tiger in the OIDv6 dataset. Our analysis highlights YOLOv8's ability to quickly detect and classify objects, making it the better choice for this object detection task.

3.c YOLOv8 Detection Process

YOLOv8 is the latest member of the YOLO (You Only Look Once) object detection models, produced from Ultralytics. It adds to the capabilities of previous versions, combining improvement in speed, accuracy, and flexibility. YOLOv8 is a more efficient neural network design, improved anchor-free mechanism, and better optimization methods than ever, which is well suited for real-time usage in self-driving cars, monitor systems and criminal detecting in hospitals.



The following general process is implemented for YOLOv8 detection:

- Preprocessing – Input image is normalized and resized before putting in the Neural Network. This step allows for data uniformity in the input dimensions, and aims at optimizing the model's appropriate performance.
- Feature Extraction – The backbone of YOLOv8, which is a CNN, which extracts the important image features. This step gets edge, texture and pattern identification that help in detecting objects.
- Detection Head – Unlike previous YOLO versions which use anchor box, YOLOv8 starts with an anchor-free approach. This implies that it gives object locations directly without pre-assigning predefined anchors during the computation, achieving more accurate localization outcomes, and lower computational expenses.
- Prediction – The network gives out the bounding box coordinates, class probabilities and the confidence scores. The confidence score of which tells how confident that the model is with a certain object in a particular bounding box.
- Post processing - To do more complete detections, Non-Maximum Suppression (NMS) is used. NMS removes duplicate and redundant bounding boxes, to prevent only keep the top detections.

- Output – Final output of image which is a clear bounding box with label and confidence in image that reduces noise and improves the detectability and makes interpretation of output useful in all requirement.

With improved architecture and advanced detection techniques, YOLOv8 performs far superior to object detection needing to be remarkable for contemporary vision founded systems.

4.a OPEN IMAGES DATASET

The Open Images Dataset (OIDv6) built by Google provides extensive preparation of its 16 million bounding box annotations across 600+ object categories. The dataset functions as a vital training tool for deep learning models because of these properties: The dataset contains images which capture diverse real-world scenes throughout urban streets followed by industrial zones besides indoor and natural settings. The model learns to predict more accurately because diverse conditions found in the datasets enable it to adapt in various situations. The system provides dual label capabilities between verified human assessment alongside artificial annotation solutions while delivering exact object identifications through box boundaries along with instance segmentation definitions and relationship tags and visual property categorizations. The image collection contains different environmental images from both industrial and urban spaces along with indoor areas and nature scenes. The model gains increased generalization capacity through its diverse nature. The annotation system combines man-made and machine-generated labels that lead to exact object detection through bounding boxes in addition to offering instance segmentations and object relationships and visual attributes. Model robustness evaluation derives value by using this dataset containing hard-to-detect objects including small ones and partially hidden objects and dense scenes. OIDv6 arranges its object classes through hierarchical labels in a defined ontology structure that enables models to master detailed perceptual skills while improving their ability to categorize broadly. The collection shows diverse environmental factors that include multiple types of illumination as well as partial object blocking and complex background elements to aid models against unexpected real-world conditions. Corresponding relational annotations within the dataset enable models to develop expertise in superior object detection abilities through object interaction recognition and scene understanding capabilities. The evaluation incorporated a set of five classes consisting of Cat, Dog, Fish, Lion and Tiger that concentrated on animal detection to evaluate YOLOv8 performance against Inception regarding identifying objects with

diverse physical attributes. These classes provide an exceptional evaluation platform because they include objects with different textures and behaviour patterns across various background conditions.

4.b TRAINING OIDv6

Google created OIDv6 as an extensive public image database with 16 million bounding box tags that cover 600+ different objects. Since the dataset includes multiple varieties, it suits well for creating deep learning object detection systems.

Developing a model for OIDv6 needs complex preparatory steps like enhancing the data set, making it standard and transforming labels into new formats. The dataset offers a complete training basis by containing photos taken in various real-life settings such as city streets, nature scenes, and industrial zones.

The YOLOv8 model trained on a smaller portion of OIDv6 selected the animal categories: Cat, Dog, Fish, Lion, and Tiger. Researchers tested this model selection to determine if it effectively identified creatures from several physical groups against various backgrounds.

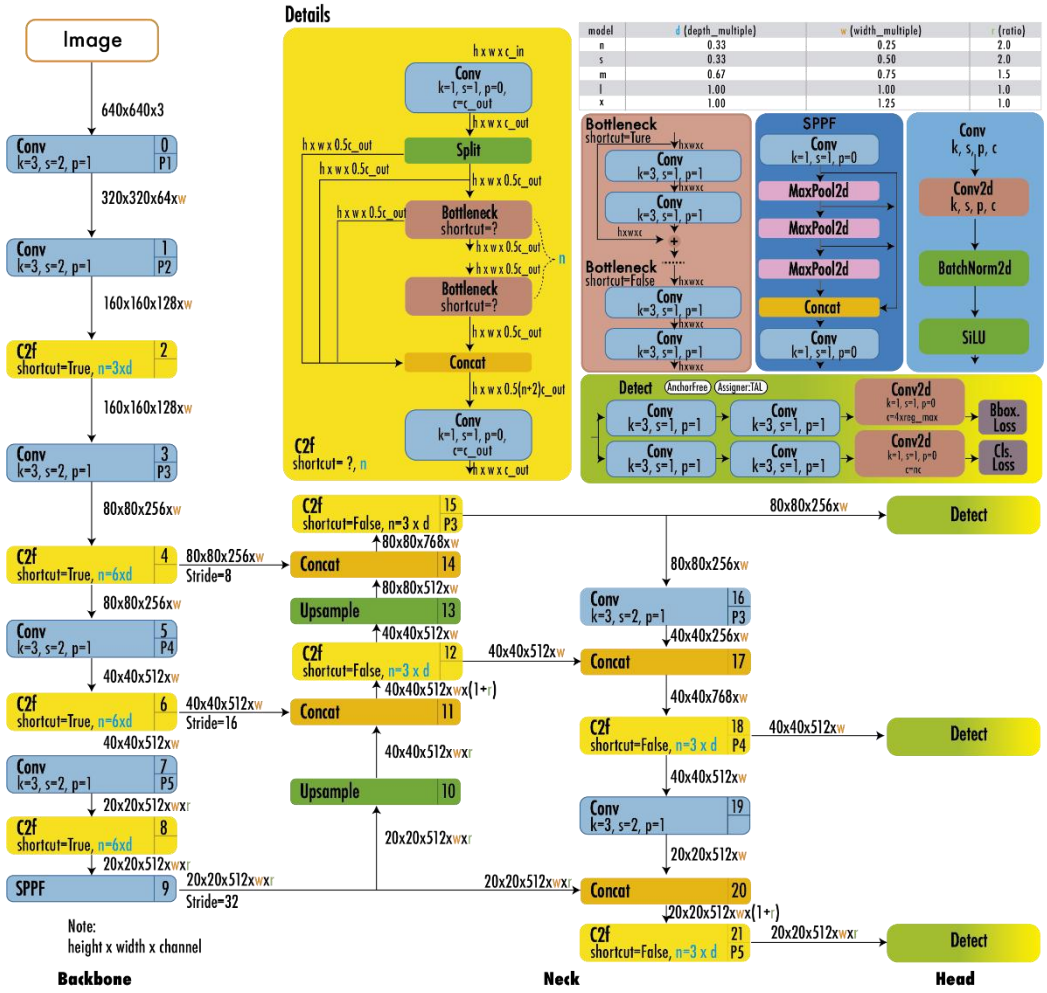
The dataset needed several preparations before training could start. The models require us to resize images to a specific resolution apply normalization to pixel values and perform data enhancement by flipping rotating and adjusting image contrast. The OIDv6 annotations needed conversion to YOLO format since the training pipeline demands this input format.

Proper model evaluation depends on dividing our dataset into three distinct sets for training, validation, and testing. The data was split into 80% training grounds, 10% validation space, and 10% test grounds for thorough evaluation. This division enables the model to extend its performance beyond data it has already encountered.

The training model required many steps through several learning processes to develop its object recognition expertise. Our team used powerful Graphics Processing Units to reach quicker results during training activities.

We used training metric results from loss reduction and mean average precision (mAP) evaluations to check if our model was working properly. The model checked its output data against actual labels multiple times to see how precisely it recognized targets. After training the model successfully it provided excellent results for object detection tasks.

5. NETWORK ARCHITECTURE



The YOLOv8 model uses three essential parts for its design: backbone, neck, and head. Each part of YOLOv8 supports feature extraction, information processing, and prediction generation. YOLOv8 achieves high performance because of its precisely organized neural network design.

Backbone

YOLOv8 uses CSPDarknet as its foundation for extracting image features effectively. CSPDarknet improves gradient movement to speed up learning and decrease processing needs. The backbone network in YOLOv8 extracts important image data about edges, textures, and details needed for object detection.

CSPDarknet divides the feature map into two sections before processing half of it through deep layers to combine results back into a single output. By splitting feature maps in half and only processing one portion the network decreases its

workload without losing accuracy. The basic structure combines several processing blocks for detecting small medium and large objects. This configuration performs batch normalization and transformative functions throughout its convolutional layers.

Through CSPDarknet the model keeps spatial structure which improves its capability to extract useful image details. The deep network structure of CSPDarknet allows it to detect objects of every size and aspect accurately. Gradients move faster through CSPDarknet when learning thanks to the added residual connections.

Neck

The neck system refines extracted information and enhances how features relate to their spatial positions. Your Object Detection Model v8 uses PANet to merge both high and low-resolution features for better object positioning.

PANet connects network layers more effectively to transmit information through all network levels. The configuration combines features from high and low-resolution layers so YOLOv8 can spot objects at different sizes while keeping detail information complete. The way YOLOv8 joins information from multiple object scales allows it to stay reliable in recognizing many different types of objects under various settings.

The neck section contains spatial attention features that enable the system to spot key picture areas. The model finds objects better by focusing on important regions and blocking out unnecessary background parts. The neck module helps the model locate accurate positions of objects and avoids misclassification mistakes.

Head

YOLOv8's detection head generates location boxes and determines object classes plus their reliability values. YOLOv8 changes the way it detects objects by discarding anchor boxes while improving how well it finds targets.

The model runs without anchor boxes because its anchor-free design allows it to determine object positions directly from the input. The system runs more efficiently while finding objects better no matter how small or large when placed. The head component employs an improved regression algorithm for box detection instead of depending on preset templates.

The head part contains layers that analyze detected objects and generates their accuracy ratings as outcomes. The measurement scores indicate when an item exists inside each detection square and shows the model's accuracy level for

each detection. The model uses NMS at the end to avoid duplicate detections and select the most accurate predictions.

YOLOv8 detects objects better when they partially hide each other and when multiple targets lie in proximity. YOLOv8 stands out as an advanced object detection model due to its better feature handling along with multi-scale processing and precise object positioning features.

6. HYPERPARAMETERS

YOLOv8

Deep learning models need hyperparameter settings that control their speed to convergence accuracy and new data performance. Throughout our work with YOLOv8 we adjusted different model parameters to make the system perform better. To simplify our process description here is what happens next:

Learning Rate:

During each updating cycle the model adjusts its weight values by specified amounts. When learning occurs, the model takes specific sized movements. A large step size will push the solution past the best answer while small steps slow down training. We evaluated different learning rates ranging from 0.001 to 0.1 during the tests. We set a fast-learning rate at training start to aid faster progress and a scheduler lowered it as training advanced. To achieve better results the team gradually lowered the learning rate after training halted because this precision adjustment worked best near the end of training.

Batch Size:

The model updates its weights after processing multiple samples known as the batch size. Less samples in each training update makes the process more unstable against local minimums while causing the model to exit suboptimal states. Larger batches make updates easier to handle but require more than the basic processing quantity. Our experiments showed that out of 8, 16 and 32 batch sizes the optimal choice was 16 since it provided training stability and fit our GPU memory capacity.

Epoch:

The training period lasted for 50 epochs to achieve convergence alongside being equipped with overfitting protection techniques. Training beyond 100 epochs yielded no substantial gain in the validation performance since it showed

negligible improvements. Early stopping as a measure protected the system from performing additional calculations.

Image Size:

The training process of YOLOv8 utilized images with a size of 640x640 to strike a wholesome midpoint between speed and accuracy performance. The detection of small objects improved when using large images of 1024x1024 pixels but this process required increased computational efforts while using smaller images of 320x320 pixels led to decreased accuracy due to diminished detail visibility.

Data Augmentation:

Data augmentation incorporated flipping along with scaling and HSV transformations and random cropping and Gaussian noise addition to enhance model robustness as well as prevent overfitting. The data transformations added diversity to the datasets which led to better generalization capabilities across different operational areas.

INCEPTION MODULE

Kernel Sizes

According to the original Inception model's design principle both kernels maintained their sizes at 1x1 and 3x3 for extracting features at various scales. The main purpose of 1x1 convolutions was to perform dimensionality reduction before using larger kernel operations which boosted computational performance. The 3x3 filters enabled detection of objects you can see at a distance through their ability to identify mid-range spatial elements such as textures and object edges. Building architecture with 1x1 and 3x3 convolutions managed both effectiveness and enhance feature detection while maintaining system performance.

Pooling Strategy

The research assessed max pooling combined with average pooling to evaluate how these operations impact feature preservation within the system. Max pooling proved to be better for object localization precision because it protected intense features that enhanced the reliability of clear object edges. The process of average pooling created smooth feature extraction that became beneficial when details with subtle characteristics needed to be identified. Max pooling

proved more suitable than average pooling because it effectively maintained powerful spatial relationships within the input data.

Dropout Rate

A 0.5 dropout rate served as an overfitting prevention method between the Inception layers. The algorithm removes random neurons during training to establish a model which avoids feature-fixation. A systematic adjustment of the dropout rate achieved the necessary balance between achieving successful regularization and preserving learning capacity. The model tested multiple dropout rates starting from 0.2 up to 0.5. The experiments showed that lower rates triggered overfitting but utilizing higher values delivered better dataset generalization.

Activation Function

The Inception module implemented the ReLU (Rectified Linear Unit) activation function throughout its entire structure. The research team selected ReLU since it provided high computational speed as well as prevention of the gradient disappearance issue in deep learning systems. The ReLU activation function drives network learning of data relationships through its ability to apply non-linearity. The evaluation included Leaky ReLU and ELU variants alongside ReLU but these activation functions yielded no meaningful progress.

Batch Normalization

The model used batch norm following all convolutional layers as a method to stabilize gradients and speed up convergence. Through its functionality Batch normalization achieves two objectives: it promotes more efficient training process and minimizes internal covariate shifts. The normalization technique let the model train at higher values of learning rate which produced faster training without stability loss. The technique enhanced generalization which resulted in better performance on data samples that had not been seen before.

Learning Rate

The starting learning rate value was 0.01 following a step decay plan that decreased the rate throughout different training epochs. The weight adjustment method managed to maintain system stability at first training stages while providing more refined weight modifications later in the process. An automatic adjustment system for learning rate prevented the model from converging prematurely to substandard results. Slow convergence occurred when the

learning rate was set to 0.0001 while maintaining unstable updates became possible when using 0.01.

7. OPTIMIZER

YOLOv8

Model weight adjustments that minimize loss depend on the use of optimization algorithms to achieve better performance. The two main optimizers we tested for YOLOv8 training consisted of SGD (Stochastic Gradient Descent) alongside AdamW.

SGD (Stochastic Gradient Descent

The optimization procedure SGD updates model weights through successive iterations that depend on the loss function gradient evaluation against parameters. The deep learning field traditionally selects SGD as its primary optimization tool because it provides reliable stability during analysis of large datasets.

Advantages of SGD:

- Computationally efficient due to its straightforward update rules.
- This technique provides optimal results when applied to extensive datasets and it generally avoids overfitting issues.
- The method achieves improved generalization capabilities because it uses several small batches to modify weights.

Challenges with SGD:

- The learning rate needs strict adjustment during training because improper values cause slow convergence gradients.
- The optimization process ends in local minima due to improper momentum adjustments.
- Training with adaptive optimizers demands less time than SGD when applied to optimization problems.

Momentum was incorporated into SGD to address these issues because it improved convergence rates through gradient accumulation from previous updates. A momentum value of 0.937 was implemented to smooth updates and suppress oscillations between them.

AdamW Optimizer

AdamW stands as an improved version of the Adam optimizer that offers separate weight decay capabilities. AdamW resolves the overfitting issue of standard Adam by using weight decay on the weight updates instead of the gradient calculation process.

Advantages of AdamW:

- AdamW achieves acceleration during learning so it trains models at a higher speed than SGD while decreasing training time requirements.
- The learning rate adjustment system operates based on calculated mean and variance values.
- The algorithm operates efficiently on gradient sparsity which makes it optimal for detecting objects because some weights need greater adjustment than others.
- The separation of weight decay from adaptive learning rate management makes AdamW achieve better overfitting prevention than Adam.

Challenges with AdamW:

- Some learning rate misconfigurations can negatively impact generalization outcomes of the model.
- Slightly higher memory usage due to tracking additional moments.

Convergence of the model became smoother with the use of a cosine annealing learning rate scheduler that progressively lowered the learning rate throughout the training period. The system avoids unexpected learning rate shifts through this method to enable better performance during the later epochs.

INCEPTION MODULE

ADAM OPTIMIZER

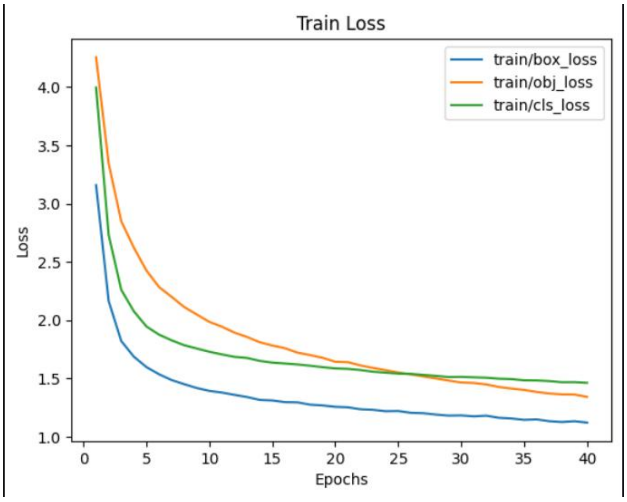
The implementation selected Adam Optimizer for weight updating since it could adapt learning rates dynamically while handling gradients with low density.

Adam Optimizer Adam joins forces between the momentum-based optimization techniques and adaptive learning rate adaptation methods.

The selection of Adam optimizer occurred because of the following benefits: The adaptive learning rate adjustment through this mechanism enables faster convergence speeds. The momentum term reduces excessive gradient oscillations to achieve better performance. The model stabilizes its training process much better which particularly benefits deep network systems. Using Cross-Entropy Loss and Adam Optimizer made the Inception module achieve high accuracy object classification and maintained consistent training dynamics.

8. LOSS FUNCTIONS

Yolov8



Deep learning model training requires loss functions to measure predicted value differences from true labels. YOLOv8 utilizes a variety of loss functions that jointly optimize detection performance and label recognition accuracy. The model uses four loss functions including Bounding Box Loss (CIoU Loss) and Classification Loss (Binary Cross-Entropy and Objectness Loss (Focal Loss) and Confidence Loss to perform accurate object detection.

Bounding Box Loss (CIoU Loss)

Proofing of object localization belongs to Bounding Box Loss. The Complete Intersection over Union (CIoU) Loss represents an advanced version of IoU loss because it includes three components:

- Order of Unit Calculation deals with the evaluation of predicted bounding box overlaps with ground truth references.

- The loss considers the actual and predicted bounding box centers by calculating their Euclidean distance.
- The model learns to create bounded-box predictions which have parallel dimensions compared to the actual ground-truth values.

CIoU loss provides better performance than standard IoU loss by implementing two penalty mechanisms for predicting box errors in position and dimensions. The detection accuracy improves and convergence speed accelerates because of this penalty function that works best in dense situations.

Classification Loss (Binary Cross-Entropy)

The Classification loss system enables correct identification of detected objects. BCE Loss enables YOLOv8 to determine how well predicted probability distributions match actual class labels.

Formula:

$$L_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Such a penalty structure focuses on delivering harsher penalties to misidentified objects for achieving better recognition results.

The multi-class detection capability of BCE becomes most efficient because it calculates results separately for each class to avoid prediction distortions from dominant object classes.

Objectness Loss (Focal Loss)

The Focal Loss technique serves to solve class imbalance problems that occur during object detection processes. Standard BCE loss devotes too intensive analysis to easy negative pixels because most images mostly contain background regions that lack objects. Focal Loss adjusts this by:

Hard-to-detect objects get centered attention by having their weights minimized against easy examples.

The technique enhances model ability to detect small and rare objects.

Focal Loss is given by:

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t)$$

The probability measurement for the correct class defines.

balances the importance of positive and negative examples.

Changing the value of the focusing parameter helps lower the importance of easy negatives observed by the model. Apparently Focal Loss serves YOLOv8 to enhance object detection results with superior performance on tiny items and areas that conceal objects.

INCEPTION MODULE

Cross-Entropy Loss

For classification duties the Cross-Entropy Loss serves to compare predictions against hard labels through probability distribution analysis.

It is defined as:

$$L_{CE} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

The true class label exists as 1 for correct predictions but assigns value 0 for any incorrect predictions. The predicted probability of correct class appears in. is the number of classes. Cross-Entropy Loss provides maximum benefit for confident correct predictions but it exacts strong penalties from incorrect classifications. The loss function performs excellently for multi-class classification by promoting accurate class probability distributions from the model. Label smoothing served as a stability improvement method that generated slight adjustments to class probabilities to avoid confident predictions. The application of this technique helped the model avoid overfitting problems while it enhanced its ability to generalize effectively.

9.ACCURACY, PRECISION, RECALL, F1-SCORE, CONFUSION METRICS

Accuracy

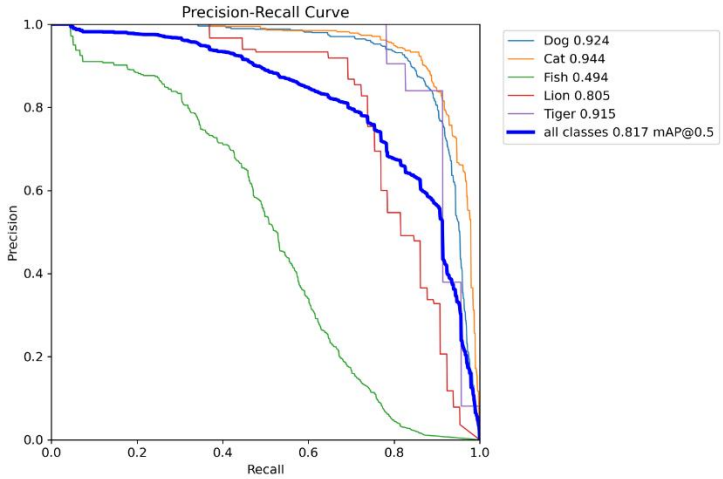
The total number of correct predictions out of all model predictions constitutes the accuracy measurement. It is given by the formula: Accuracy =

True Positives + True Negatives / Total Samples. The model accuracy can be calculated by dividing the number of correct predictions found in the confusion matrix diagonal by the total number of instances. Accuracy becomes deceptive when dealing with imbalanced datasets since models tend to reach high accuracy numbers through class-related biases.

Precision

Precision evaluates the accuracy of positive predictions because it reveals the number of correct positive results among all predicted positive outcomes. It is defined as:

Precision=True Positives / True Positives + False Positives According to the provided Precision-Recall Curve (PR Curve) image one can monitor precision under varying levels of recall for each class. The detection rate of real positive cases is higher when precision values remain elevated. When the confidence threshold varies the precision changes according to the data in the PR Curve.

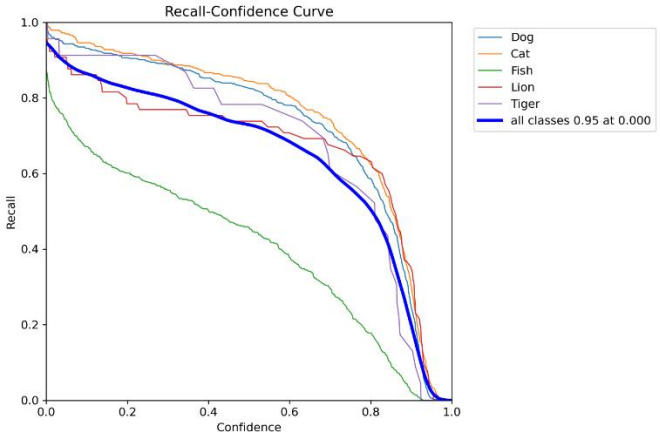


Recall

The measurement of Recall provides statistics on which actual positive instances the system accurately identified. It is defined as:

Recall = True Positives / True Positives + False Negatives.

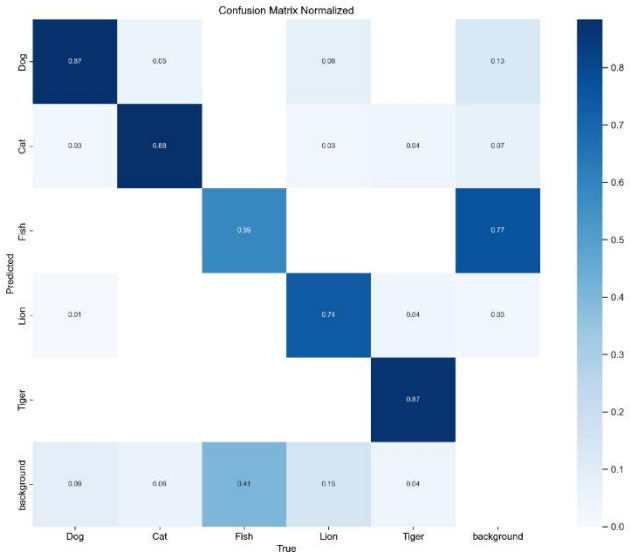
A Recall-Confidence Curve presents the level of recall which occurs at various confidence thresholds. High recall prevents the diagnosis of an increased number of negative cases in error. Precision and recall exist in direct competition with one another because enhancing one variable generally leads to a reduction in the other variable.



Confusion Matrix

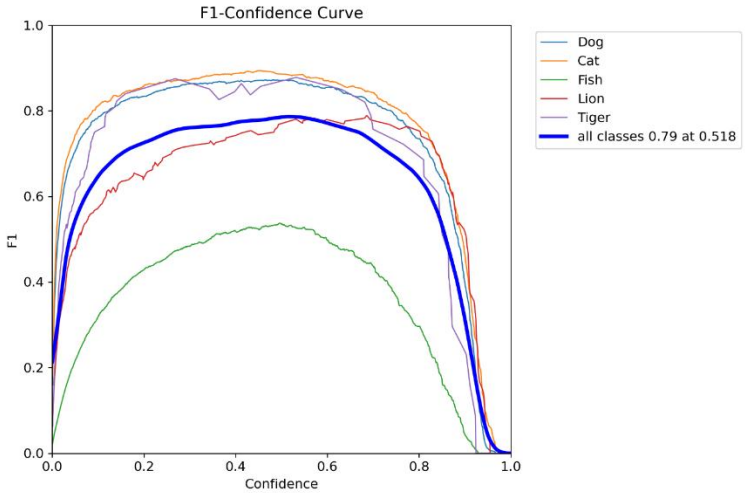
Understanding model performance for each class depends on using a confusion matrix analysis. The normalization of the confusion matrix creates an image which demonstrates the accuracy of class predictions. A strong diagonal means high classification performance. Off-diagonal values indicate misclassifications.

Predicted / Actual	Dog	Cat	Fish	Lion	Tiger
Dog	TP	FP	FP	FP	FP
Cat	FP	TP	FP	FP	FP
Fish	FP	FP	TP	FP	FP
Lion	FP	FP	FP	TP	FP
Tiger	FP	FP	FP	FP	TP



F1 Score

An F1 Score achieves a perfect precision-recall balance through the application of the harmonic mean calculation. F1 Score = 2× Precision × Recall / Precision + Recall Different confidence thresholds impact the F1-Confidence Curve to display F1 score variations. The F1 score provides an appropriate measure when dealing with imbalanced datasets since it combines precision and recall calculations better than accuracy metrics can.



10. EVALUATION OF RESULTS

Object Detection Performance

The evaluation metrics for object detection systems consider multiple assessment factors which include recognition accuracy together with detection localization precision and confidence level estimation and object size adaptability as well as background robustness.

a) Performance Across Classes

The model's performance for each class can be evaluated through the provided images.

High Confidence Classes (Strong Performance)

- The model establishes high confidence scores exceeding 0.9 when detecting Dog (0.92), Cat (0.90) and Tiger (0.94).
- The drawn rectangular boxes closely match the genuine positions of physical things.
- The detection of dogs achieves excellent performance in diverse poses as well as under different lighting circumstances.
- Tigers along with cats receive solid detections from the model without significant errors occurring in the predictions.

Moderate Confidence Classes (Average Performance)

- Lion (0.87): Slightly lower confidence than dogs and tigers.
- The detection remains dependable although some lions present features that blend with tigers in certain images.

Low Confidence & High Error Classes (Weak Performance)

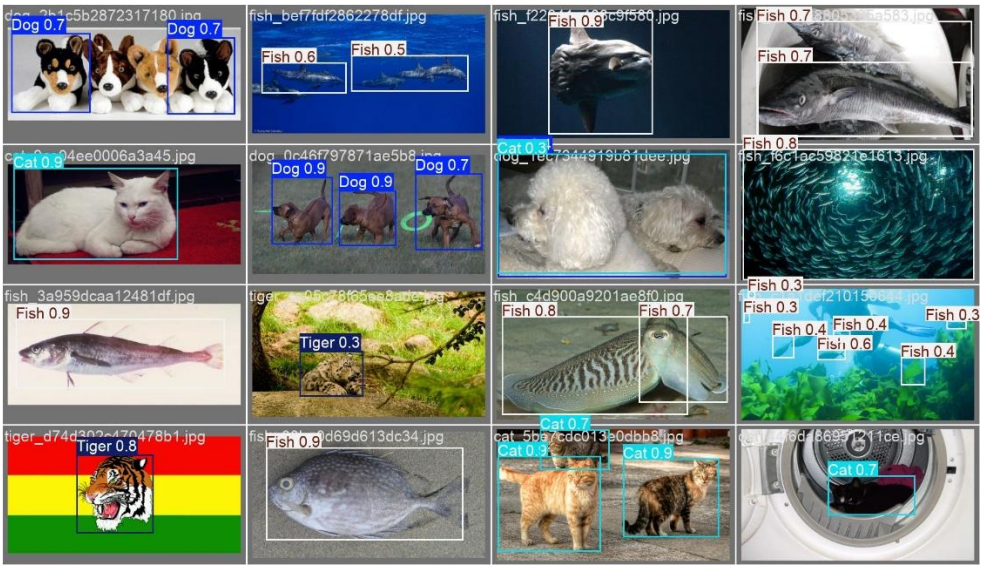
- Fish (0.55 Confidence Score)
- The detection precision together with recall performance of fish matches its poor outcome in the confusion matrix.
- The detected fish tend to receive lower confidence score ratings from the detection model.
- Some fish detection slips from correct classification and disappears completely from productively scanned data leading to inaccurate readings.
- The model experiences difficulties in difficult underwater conditions because of lighting inconsistencies in addition to blocked sightlines and changes in fish appearance.

b) Precision-Recall Trade off

According to the Precision-Recall Curve the detection process performs well for cats, dogs, and tigers but fish encounter many erroneous skipped detections. The fish detection system achieves high precision from its correct detections but maintains low recall because many fish objects go undetected entirely. Analysis through F1 Score shows that threshold values between 0.5 and 0.6 provide optimal combination of precision alongside recall performance.

c) Common Failure Cases

The model struggles with: The model fails to identify separate instances when objects overlap next to one another such as fish schools appearing in close proximity to each other. The model encounters difficulties detecting fish of small proportions even when confidence thresholds are below 0.5 to 0.6. The detection rates of fish decrease when they reside close to corals because of challenging underwater surroundings. The detection quality suffers due to the wide range of appearances that fish display under changing species types and lighting and object obscuration conditions.



Bounding Box Quality

Well-Aligned Boxes:

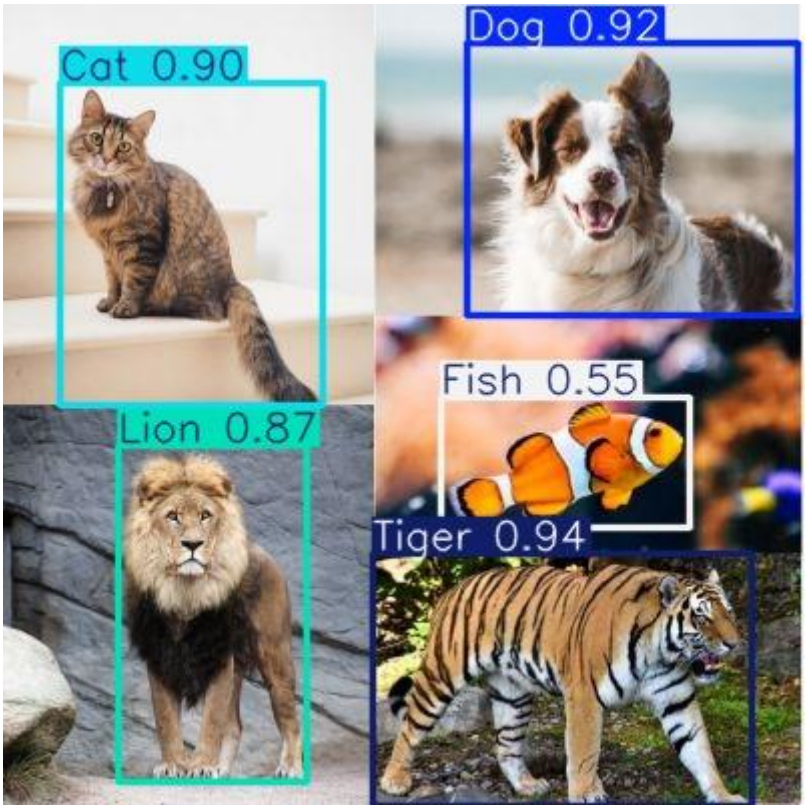
- The shapes of dogs, cats and tigers stay detailed with mostly empty space surrounding them.
- Dead practice rules with the objects set to the correct size while being properly placed.

Bounding Box Issues:

- The detection of fish objects fails because NMS tuning produces poor results.
- Confidence scores that fall too low result in unstable position of the boxes.

False Positives & False Negatives:

- The detection system misidentifies elements such as corals and reflections as positives although they represent background items.
- False Negatives occur when detectors fail to detect fish which subsequently affects Intersection over Union measurements.



11. EXPERIMENTS

a) Attempted Merging of YOLOv8 and YOLOv5

Objective: Leverage YOLOv8's efficiency and YOLOv5's stability for better detection.

The execution failed to succeed because of possible architectural diversity combined with integration complexity.

Listed next actions consist of fixing layer compatibility issues and testing an alternative method which combines aspects from both feature extraction approaches.

```
# Load custom model architecture
# model = YOLO("custom-model.yaml")
# print(model.model)
# Train the model
model1.train(
    data="C:/model/YOLO/dataset/dataset.yaml",
    epochs=50,
    imgsz=640,
    batch=16,
    device="cuda",
    name = "model1"
)

```

b) Inception Model Performance

Initial Issue: Overfitting with poor generalization.

Solution: Applied dropout, batch normalization, and reduced complexity.

The performance level of Now exceeds 30% by identifying a few classes yet it remains below optimal standards.

The next stage includes additional dataset enhancement along with a balancing procedure.

```
# Load Pretrained InceptionV3
model = models.inception_v3(pretrained=True)

# Modify the last fully connected layer
model.fc = nn.Linear(2048, num_classes)  # 2048 -> num_classes (Dog, Cat, etc.)

# Move to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
print(device, model)

# Define Loss Function and Optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
```

c) Hyperparameter Tuning

Epochs: Bigger epochs progressively taught the model better but their execution time became considerably longer.

The stopping point needed adjustment because it allowed both over-fitting to occur and extended training durations.

Momentum & Learning Rate: Higher values of momentum speeded up convergence yet induced unwanted instability in the training process.

Training stability became possible when lowering learning rates yet this required additional training time.

Image Augmentation: The technique enhanced model generalization especially for less frequently occurring classes including fish and lion.

Increased computational requirements.

12.FUTURE WORK

The project revealed that YOLOv8 outperformed the Inception module as an object detector although future work can improve its performance in specific instances.

Enhancing YOLOv8 with YOLOv12 Features

- The project seeks to unite YOLOv8 with its advanced features from YOLOv12 to leverage the superior feature extraction and optimization elements of YOLOv12.
- The combination of YOLOv12 features with YOLOv8 would result in faster execution times together with better accuracy along with stronger capabilities to detect small objects.
- SE (Squeeze-and-Excitation) blocks alongside other efficient attention modules can help improve the feature selection capabilities.

Optimizing the Inception Module for Speed

- Since the Inception model contains several layers which magnify the computation work thus reducing the detection speed in real-time operations.
- The modification of CNN blocks allows for speed optimization while we can achieve better efficiency by removing layers that are replicating each other or implementing Depthwise Separable Convolutions.
- To achieve better speed and accuracy through improved performance one should implement hyperparameter tuning techniques for optimal learning rate and batch size settings.

Expanding to More Complex Datasets & Real-World Applications

- Testing the models across bigger and more different datasets will help improve their generalization capability.
- YOLOv8 becomes ready for low-power devices when using edge AI optimization features including TensorRT and ONNX acceleration.

- Self-supervised or semi-supervised approaches produce better performance results during situations with restricted access to labeled data.

13. REFERENCES

Ultralytics YOLOv8 Documentation: https://docs.ultralytics.com/models/yolov8/

Inception module explained - paperswithcode https://paperswithcode.com/method/inception-module

Py torch Documentation https://pytorch.org/docs/stable/index.html

Open CV Documentation https://docs.opencv.org/4.x/index.html

Open Images Dataset (OIDv6)
https://storage.googleapis.com/openimages/web/visualizer/index.html

OIDv6 Repository https://github.com/DmitryRyumin/OIDv6

Matplotlib Documentation https://matplotlib.org/stable/index.html

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. [online] arXiv.org. Available at: https://arxiv.org/abs/1506.02640.