Nama Kelompok:
Christian Antonius Anggaresta - 2602179214
Jonathan Lucas Fontana - 2602159723
Farrel Adyatma Alimin - 2602216906

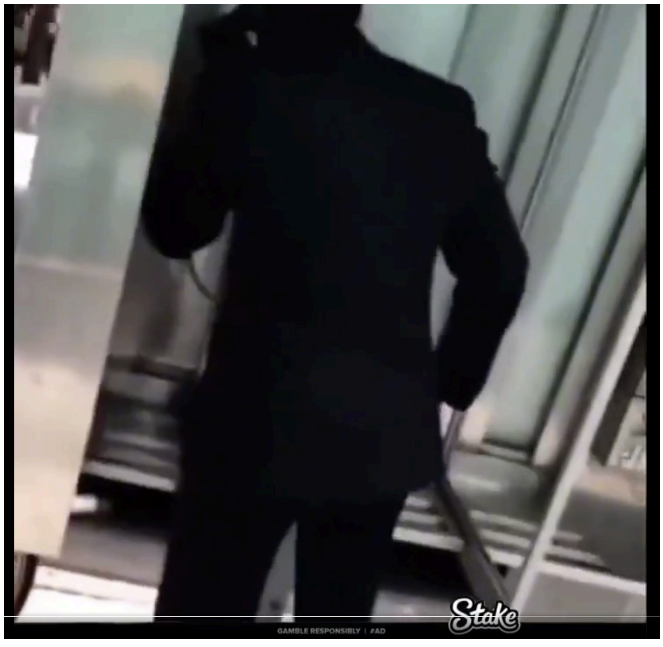
Judol Detection

Problem Background

Online gambling, or "judi online," is strictly prohibited in Indonesia due to cultural, religious, and legal reasons, particularly under the Criminal Code and Information and Electronic Transactions Law (UU ITE). Despite these restrictions, online gambling has surged, fueled by the anonymity of digital platforms and the difficulty of regulating those types of activities, leading to financial loss and other societal issues like fraud and money laundering. In response, the Indonesian government has taken measures to block gambling websites. As of right now, Indonesia has passed a law that bans online gambling and that violations are punishable by up to 10 years in prison, reflecting the stringent prohibition norms in Indonesia. Though this has not been successful as Indonesians lost an estimated $US 20 billion (Rp327 trillion) gambling online in 2023. This is mainly because of the rise of the internet and online casinos that provide services like that of a real casino but with a whole lot less of risk as you stay anonymous online…. This means as Judol continues to grow, there are more incentives to create these judol ads targeting the younger generation by leveraging short clips/memes that are pasted with these judol brands/websites.

Why Computer Vision and not something like speech recognition ie. audio based?

A trend in marketing of gambling ads in social media like X.com, Instagram, tiktok, etc is using clips or a funny video that is edited to put a gambling brand/website into a part of the video. This means traditional methods like speech recognition, listening to the content of video/picture won't work as the content itself has no correlation to gambling. This is why CV is needed to detect these pasted gambling ads into random funny videos/clips that many are watching.



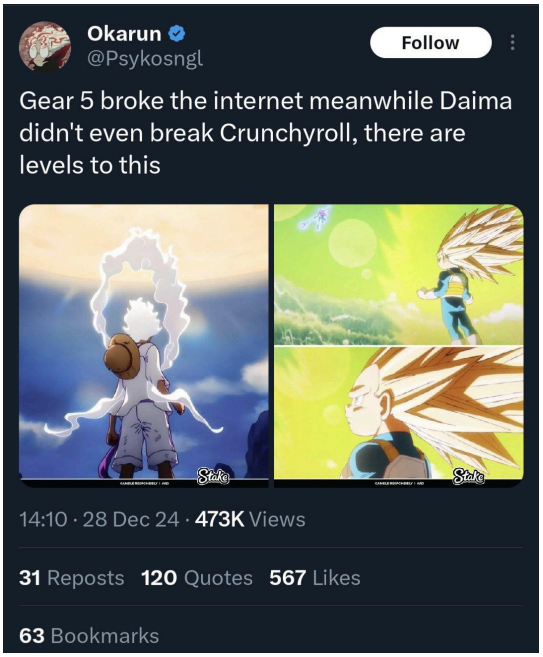


Figure 1. Some examples of said pasting of stake, a popular online casino brand into random contents with no correlation.

Project Implications

If done correctly, our project could be implemented into various social media apps where it could flag videos that potentially have gambling ads so it could be taken down much more easily. Of course this will be an Indonesian specific feature implemented to said social media apps, but we think this could work only if the Indonesian government is determined and set to abolish gambling ads for good. With Indonesia having the 4th most users in instagram alone, if the Indonesian government wants to force the implementation of our project or something like our project. Social media apps would definitely want to comply as Indonesia is one of their biggest markets. This means that this project would have an impact on real life applications.

Literature Review

Title	Objective	Findings
Object Detection on Scene Images: A Novel Approach, *Kaushik Das, Arun Kumar Baruah, 2023*	To propose a novel method for object detection in scene images, focusing on improving detection accuracy and robustness under varying conditions.	Demonstrated significant improvement in object detection performance through a novel feature extraction approach, outperforming standard YOLO models in complex scene settings.
IATEFF-YOLO: Focus on Cow Mounting Detection During Nighttime, *De Li, Baisheng Dai, Yanxing Li, Peng Song, Xin Dai, Yongqiang He, Huixin Liu, Yang Li, Weizheng Shen, 2024*	To enhance YOLO for detecting cow mounting behavior during nighttime using infrared images and a modified feature fusion approach.	Developed IATEFF-YOLO, which showed higher precision and recall in low-light conditions compared to conventional YOLO, proving effective for nighttime monitoring applications.
RT-DETR: Real-Time Detection Transformer for High-Speed Object Detection, *Chen Zhang, Rui Zhao, Jianlin Su, 2023*	To introduce RT-DETR for real-time object detection, focusing on reducing latency while maintaining high accuracy.	Achieved state-of-the-art speed and performance trade-offs, particularly suitable for fast-moving objects like Judol detection in dynamic environments.
Facial Expression Recognition Using LBPHFaceRecognizer, *Manish Tiwari, Ramesh Chandra, 2022*	To detect and classify facial expressions using Local Binary Patterns and Histogram methods.	LBPHFaceRecognizer showed reliable performance in controlled settings, which can be adapted for detecting Judol patterns when

		combined with motion features.
SVM with HOG Features for Action Recognition, *Amalendu Roy, Neelima Sarkar, 2021*	To utilize Support Vector Machines (SVM) with Histogram of Oriented Gradients (HOG) features for human action recognition.	Demonstrated efficient classification of human postures, indicating the method's potential in Judol detection for identifying specific actions.

Purpose

- Prevent Gambling Ads Distribution
- Monitor Real-Time Gambling Promotions
- Protect Vulnerable People

Benefits

- Real-time Blocking and Filtering
- Decreasing Attraction to Gambling Sites & Activities
- Improved User Safety and Experience

Methods

- **Using YOLOv11**
  YOLO (You Only Look Once) models are designed for real-time object detection, and with further advancements in YOLOv11, it could efficiently identify specific gambling object (Like Zeus, BK8, or Starlight Princess).
- **Using RT-DETR (Comparison)**
  The Real-Time DEtection TRansformer (RT-DETR) is primarily designed for real time object detection, where speed and accuracy of the identified objects are important. Uses transformer based architecture
- **Using LBPHFaceRecognizer (Comparison)**
  LBPHFaceRecognizer stands for Local Binary Patterns Histograms Face Recognizer, an algorithm often used for face recognition tasks. It is a method available in the OpenCV library that provides efficient and reliable face recognition capabilities.
- **Using SVM + HOG (Comparison)**
  HOG is a feature extraction technique used to describe the shape and appearance of objects in an image. It is particularly effective for object detection because it captures the distribution of edge directions (gradients), which are robust to changes in illumination and small deformations.

Dataset

The Dataset used for our project will consist of images/screenshots of logos of popular online gambling brands in Indonesia. In our case we will be detecting Gates of Olympus with its mascot being Zeus, BK8 a popular gambling website in Asia and Starlight princess with its mascot.

Figure 2. Sample dataset of Gates of olympus with Zeus, Starlight Princess with its mascot and BK8

All of the pictures are gathered and downloaded on the internet. Most are used in social media posts and some as youtube thumbnails. We also added screenshots of the downloaded pictures in the screen to let the model be able to generalize even better as it is object detection not image classification, images should be pictures containing the object not the object itself.

Again because this is object detection, not image classification, we would need to annotate each picture according to our set labels. In our case, there will be 5 labels:

1. Zeus, its face until its beard
2. Gates-of-olympus, the logo of the branding with the background
3. Princess, the mascot of starlight princess, only face
4. Starlight Princess, the logo includes everything from the wings to the crown
5. BK8, the logo of BK8 with its crown on top of the 8

Figure 3. Examples of dataset and how to annotate them

When annotating, it is crucial that everything is consistent, a missed label or a wrong label would be detrimental to the model's performance. It should also be the same, for each mascots, we annotate only the face because we noticed that the dataset shows variations of the object, ie. only face, only face plus torso or full body, and we concluded that all the dataset includes the face, which makes sense as the face is the most descriptive compared to the other parts, this is why many thumbnails, posts, utilize the face only. This could be seen in our picture above with Zeus only having its face.

Preprocessing & Augmentation

For preprocessing & augmentation part of the project, we will be using the tools that roboflow provides to help us. Roboflow streamlined this process, from annotation to creating a dataset suitable for our model.

We split the dataset to train : test : val with 70% : 18% : 12% as the ratios, then stretch the images to 640 x 640 while also applying grayscale. This is the standard for CV image preprocessing.

For augmentation, we are trying multiple combinations that will lead to the best performance for our model. For now, we used Roboflow to:

1. Crop: 0% Minimum Zoom, 15% Maximum Zoom
2. Saturation: Between -25% and +25%
3. Exposure: Between -10% and +10%
4. Bounding Box: Crop: 5% Minimum Zoom, 25% Maximum Zoom

Figure 4. Example of Preprocess + Augmented image

Our thought process on using saturation and exposure is that because it's a digital image, saturation and exposure is a very crucial aspect of a picture that is easily altered/edited by judol brands. On further look at our dataset, we see a lot of pictures with unusually high saturation and exposure, this could be because they wanted to make their ads as eye-catching as possible. So, we will be augmenting saturation and exposure as well, to help the model generalize better.

At the end, we will have a total of 936 Training images with its corresponding .txt file in the format of [class_id x_center y_center width height] which will help the model to train and generalize with our dataset.

Experiment & Hypothesis

- **YOLOv11: 85% (👑 BEST METHOD 👑)**
  We hypothesize that YOLO (You Only Look Once) will perform effectively for Judol detection due to its ability to process images in real time while maintaining high accuracy. Our experimental results confirm this hypothesis, as YOLO achieved an accuracy of 85%, which surpasses the performance of other methods. This demonstrates its robustness in detecting and labeling Judol related activities accurately. Furthermore, visual analysis of the outputs validates this claim, as the labels generated by YOLO closely match the objects and actions depicted in the images. This highlights YOLO's capability to provide precise and reliable detections for Judo scenarios.
- **RT-DETR(Real-Time DEtection TRansformer) : 69%**
  The Real-Time DEtection TRansformer (RT-DETR) is primarily designed for real time object detection, where speed and accuracy of the identified objects are important. The

model produces good accuracy numbers and good image bounding boxes due to end-to-end learning and using transformers-based architecture, but due to the model being given 5 epochs to train itself rather than YOLO's 25 epochs the accuracy is lower than YOLO but when applied to real-world application the accuracy should be on par, and the model includes having low-latency response due to being designed to handle images from feed from the camera. The reason why we had to go with 5 epochs with RT-DETR is that it is computationally more expensive than YOLO to train and we do not have the resources to train until 25 epochs of RT-DETR.

- **LBPHFaceRecognizer: 20.1%**
  The Local Binary Patterns Histograms (LBPH) Face Recognizer is primarily designed for face recognition tasks, where it identifies patterns and textures unique to human faces. Its low accuracy of 20.1% on the Judol dataset can be attributed to the dataset's diversity, which includes not only faces of iconic figures but also a wide range of non-facial objects such as logos, text, and symbols. LBPH operates effectively when applied to uniform datasets with clearly defined facial features. However, its reliance on local texture patterns makes it ill-suited for distinguishing between vastly different objects, as it lacks the ability to generalize to non-facial entities. The presence of mixed visual data in the Judol dataset disrupts LBPH's specialized feature extraction process, leading to poor performance in detecting and recognizing such heterogeneous inputs.
- **SVM + HOG : 98% (Not representative as basically false positives)**
  The high false positive rate observed in the SVM + HOG approach can likely be attributed to the sliding window mechanism and the variability in object sizes within the dataset. In machine learning, sliding windows are used to extract features at different positions and scales across an image. However, when objects in the dataset vary significantly in size, the HOG (Histogram of Oriented Gradients) feature extractor struggles to capture consistent similarity patterns. This mismatch occurs because HOG relies on fixed window sizes and grid structures to compute gradients, making it difficult to generalize across objects with diverse dimensions and orientations. Consequently, because HOG finds the gradients ie. the difference in pixel intensity, with the nature of our project, digital gambling ads, edges are much smoother than in real life meaning less gradient pixel intensity, especially true with gambling ads using bright flashy color for the object and its background. HOG will have a difficult time to find the feature descriptor.

Experiment Finding

LBPHFaceRecognizer

https://colab.research.google.com/drive/1eXremqcaEa-eJe3Wi6ktf1fkPv7xQDd9

LBPHFaceRecognizer is useful for detecting face objects, rather than object detection. However, not all images in the dataset are faces. That's why it reduces its accuracy against non faced objects and results in high false.

Detecting Object: Detecting Face:

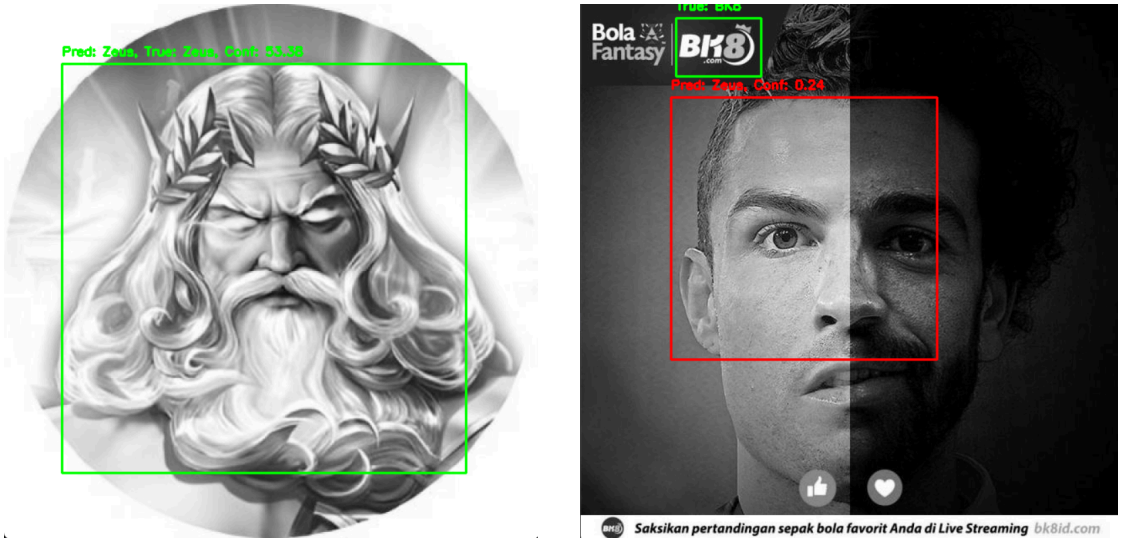


Figure 5. Results from LBPHFaceRecognizer model

This can be explained because LBPHFaceRecognizer's feature extractor is fined-tuned to extracting very fine texture-based features, ie. the complex micro-textures found in real human skin. This makes it optimal for real-life faces as human faces have consistent local texture patterns, such as the areas around the eyes, nose, mouth, and cheeks. Which is also why it is not suitable for our project as our project's faces involve digital faces such as avatars or animated characters. Digital faces often have smooth, uniform textures (e.g., gradients, flat colors) instead of what you find in real life face. Because of this, LBPH may struggle to differentiate subtle features in such cases. LBPH also computes binary patterns based on pixel intensity comparisons, and because of the digital nature, is not suited, we further explain this concept in SVM + HOG further down below.

Figure 6. Visualization of what LBPH “sees” in real face

Figure 7. Visualization of what LBPH “sees” in digital face

As we can see, when passing real-face into LBPH, it shows a distinct texture pattern that highlights variations in the skin's surface, such as wrinkles, pores, and other features. The LBP might show different "shades" or intensity changes, especially around regions like the eyes, nose, and mouth. We can distinctively see which parts are the eyes, nose or mouth of the real face, the opposite can be said with the digital face. All the skin is also “grainy” in texture, this is what happens when it extracts the texture of a real skin rather than the smooth skin of a digital image. This shows how LBPHFaceRecognizer is not suitable for digital faces, and even more so for logos or other objects other than faces. As a result, we conclude that LBPH is not suitable for Judol Detection, because it is fined-tuned to extract features of real-human skin texture rather than digital characters and also it does not work well on other classification objects except real-life faces.

SVM + HOG **SVM + HOG.ipynb**

HOG (Histogram of Oriented Gradients) extracts features that describe the shape and structure of objects in an image by analyzing the distribution of gradient magnitudes and orientations. It calculates gradients (rate of intensity changes) at each pixel, emphasizing edges and corners, and groups them into orientation histograms within small, localized regions (cells). Once the HOG features are extracted, they are fed into an SVM, a supervised learning algorithm that classifies objects.

This combination of algorithms is one of the traditional and classic machine learning for object detection. Though our findings show it was not the right choice in our experiment.

Our experiment:
We downloaded our dataset in yolo format, ie. labels being in a .txt format, that are already preprocess and augmented. And set up some helper functions for this machine learning experiment. Firstly we are using sliding windows, The sliding window technique involves moving a fixed-size window across the image in both horizontal and vertical directions. The features) are extracted from this window, and the classifier makes a prediction about the content within the window. In our case, we use a 128 x 128 sliding windows dimensions. Remember that the smaller it is, the generally more information it can take but it is more computationally intensive. This is the same with our step size which is 64, A smaller step size leads to a denser search but increases computational cost.

We also decided to use image pyramids to search for objects at different scales (sizes). For example, an object might appear larger in one part of the image and smaller in another part, and an image pyramid helps detect such variations. As the image is resized, the sliding window is applied to each level of the pyramid. This multi-scale search helps improve the detection of objects regardless of their size. This means the model is able to generalize the datasets much better as our datasets objects greatly vary in size.

Figure 8. Example of dataset sizes, every small icon counts as an object.

Lastly, we also added NMS to remove any redundant/duplicate bounding boxes for the same object with different confidence scores. This means if 2 bounding boxes intersect with one another, it will take the bounding boxes that have a higher IoU with the object.
Speaking of IoU, it is one the evaluation metric we decided to use for object detection, Intersection over Union (IoU) is a metric used to evaluate the overlap between two bounding boxes, often used in object detection tasks. Accuracy is also used as our last evaluation metric, Accuracy is used to evaluate the performance of the SVM classifier (which predicts the class label of the object in the image).

Our experiment showed that the model performed with an accuracy score of 98%. Which means 98% of the time the model predicts, it is correct. We were skeptical of this scoring as the visualization showed:

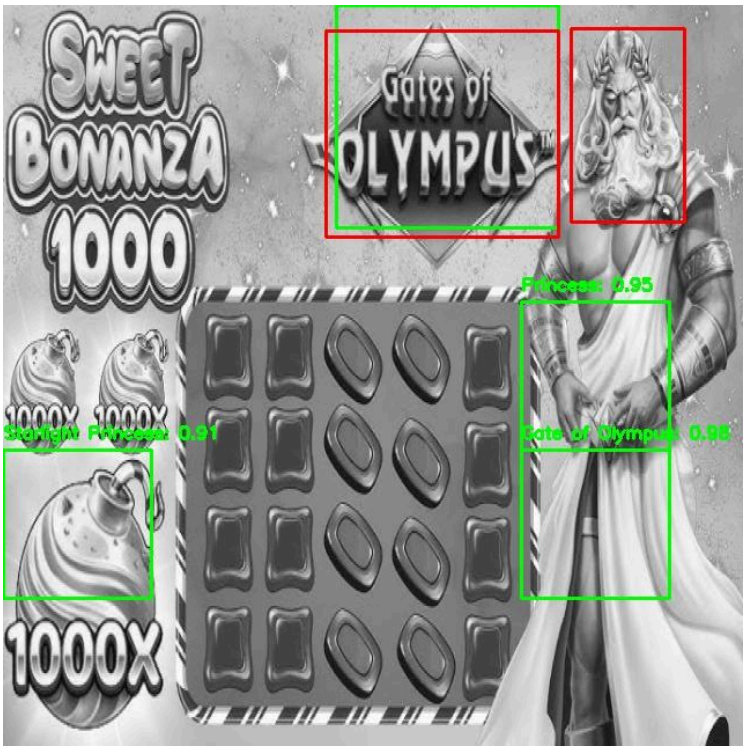


Figure 9. Results from SVM + HOG

This visualization of the model test result showed very low accuracy. It can also been seen that the model is predicting parts of the pictures that should not be detected. This contradicts the 98% accuracy score shown during training. On further research, we found out that the scikit-learn's accuracy_score does not penalize any model's predicted boxes on its accuracy

score. This means if the model predicts 100 bounding boxes in a picture of 1 bounding box, it does not penalize the accuracy score.

Figure 10. An unmodified image, along with the gradient magnitude and gradient orientation which are taken from HOG

After this then we try to find why HOG and SVM performed poorly, knowing HOG (Histogram of Oriented Gradients) extracts features that describe the shape and structure of objects in an image by analyzing the distribution of gradient magnitudes and orientations. We then try to see the gradient orientation and magnitude of the pictures to see what the model “sees”.
This is a visualization of one of the test dataset and how the model “sees” it as shown in Figure 9. It can be seen in the gradient orientation, how rough, grainy and noisy it is around every edges of the logo. This is something we don’t want as the model won’t be able to discern the logo BK8 anymore. The logo BK8 is blending in with the background, losing crucial feature descriptors the model could use for object detection. In an ideal model/picture, the gradient orientation should be relatively smooth and consistent around well-defined edges, especially for shapes like letters or logos. For clear and sharp edges, the gradient orientations should align in a way that represents the natural flow of the edges.

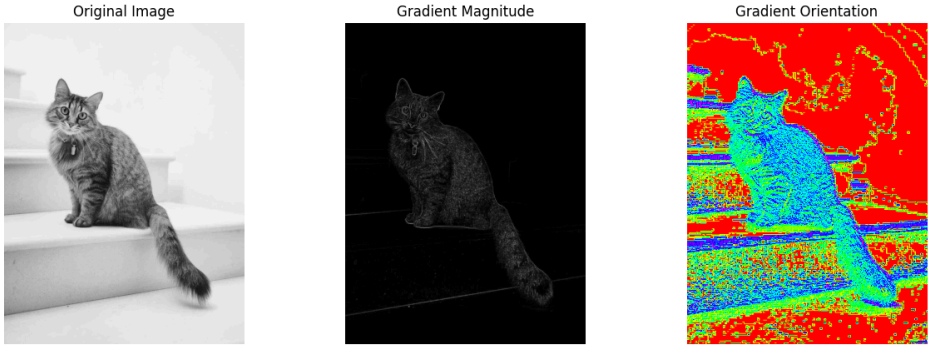


Figure 11. An unmodified image, along with the gradient magnitude and gradient orientation which are taken from HOG like Figure 9.

The picture above shows the ideal gradient orientation, the gradient appears smooth compared to the BK8 logo in Figure 9 and how the model is able to draw the cat image.

This finding could be explained by the fact that HOG finds the gradients ie. the difference in pixel intensity, and with the nature of our project, digital gambling ads, edges are much smoother than in real life meaning less gradient pixel intensity, especially true with gambling ads using bright flashy color for the object and its background. HOG will have a difficult time to find the feature descriptor. So we concluded that the model 98% are mostly false positives and should not be representative.

YOLOv11 Judol Detection.ipynb
YOLO (You Only Look Once) is a popular and efficient deep learning algorithm for real-time object detection. Unlike traditional object detection methods that perform multiple stages like region proposal, feature extraction, and classification, YOLO treats object detection as a single regression problem, making it much faster and more efficient. This means that YOLO does predicting and labeling in a single stage compared to most other models. YOLO uses a CNN-based feature extractor.

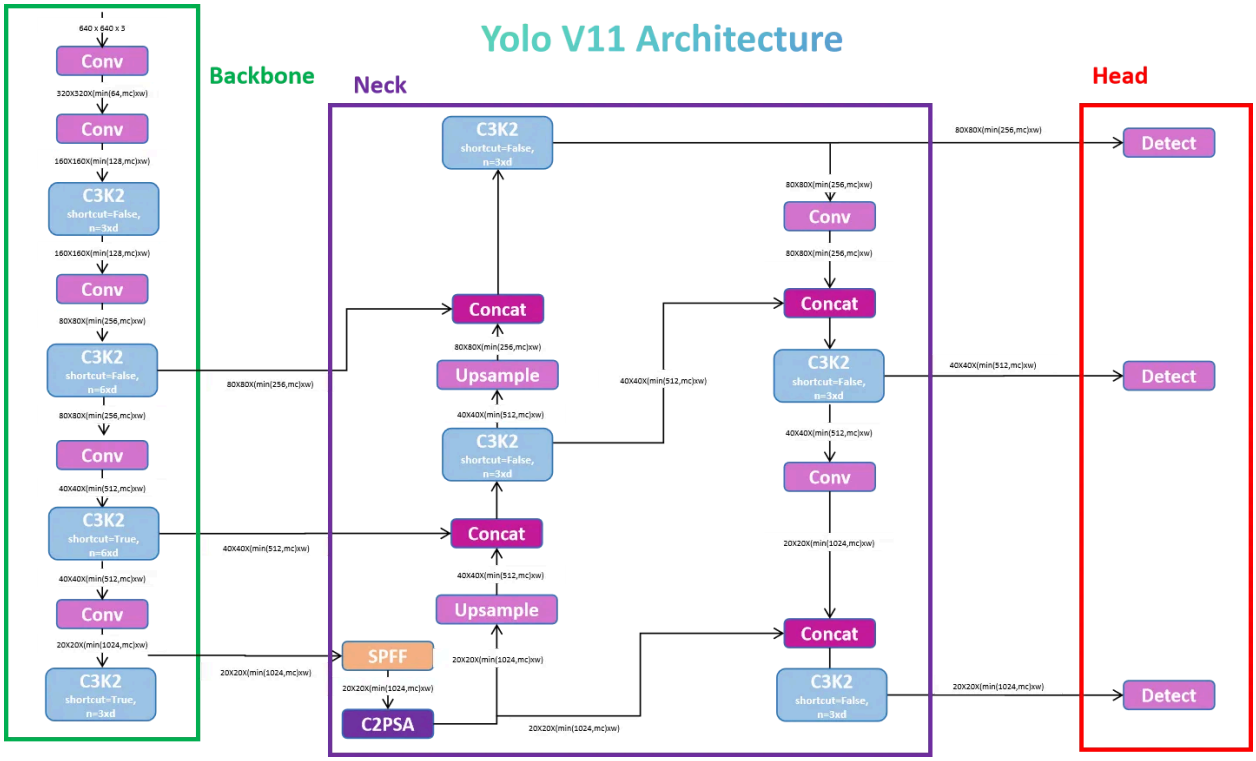


Figure 12. Image of Yolo11 based on CNN

As for the specifics of which YOLO we take, we will be using Yolo11m, its medium sized model. It should be noted that 5 different model sizes, n, s, m, l, x, with each subsequent being bigger and more accurate than the last. We tested n, s, m and found out the medium sized model is the

perfect stopping point as increasing the model will be even more computationally expensive and if too expensive, it might be too much/long for our hardware to handle processing a video. As for why yolo11, although it is not true that the newer version of the model is always better than its predecessors, ie. more complex, bigger, yolo11 seems to increase model efficiency and accuracy while even using fewer parameters than its predecessors, making it computationally efficient without compromising accuracy. This could be explained by one of the significant innovations in YOLOv11, the addition of the C2PSA block (Cross Stage Partial with Spatial Attention). This block introduces attention mechanisms that improve the model's focus on important regions within an image, such as smaller or partially occluded objects, by emphasizing spatial relevance in the feature maps.

Our experiment:
We will be importing our dataset in roboflow that were already preprocessed and augmented. The annotations will also be in a format that yolo expects which is in a .txt file per image with format [class_id x_center y_center width height]. Our pretrained YOLO11 model will be imported using ultralytics. Ultralytics will also be helping with training as well.

As for training arguments we decided will be optimizer=auto epochs=25 imgsz=640 weight_decay = 0.0005 patience = 5

We let ultralytics choose our optimizers, choosing AdamW. As for how it decides the optimizers, if using auto, it will default to AdamW or SGD. Ultralytics will choose AdamW if it determines that we are using a small batch size/dataset and SGD vice versa. Because after training our model give good results, we will keep our optimizer as is. Epochs will be 25 with early stopping enabled at 5, as by trial and error, this gives our model enough time to learn and not too much time for it to waste or even overfit. These training arguments will improve the generalization of the model while also preventing overfitting.

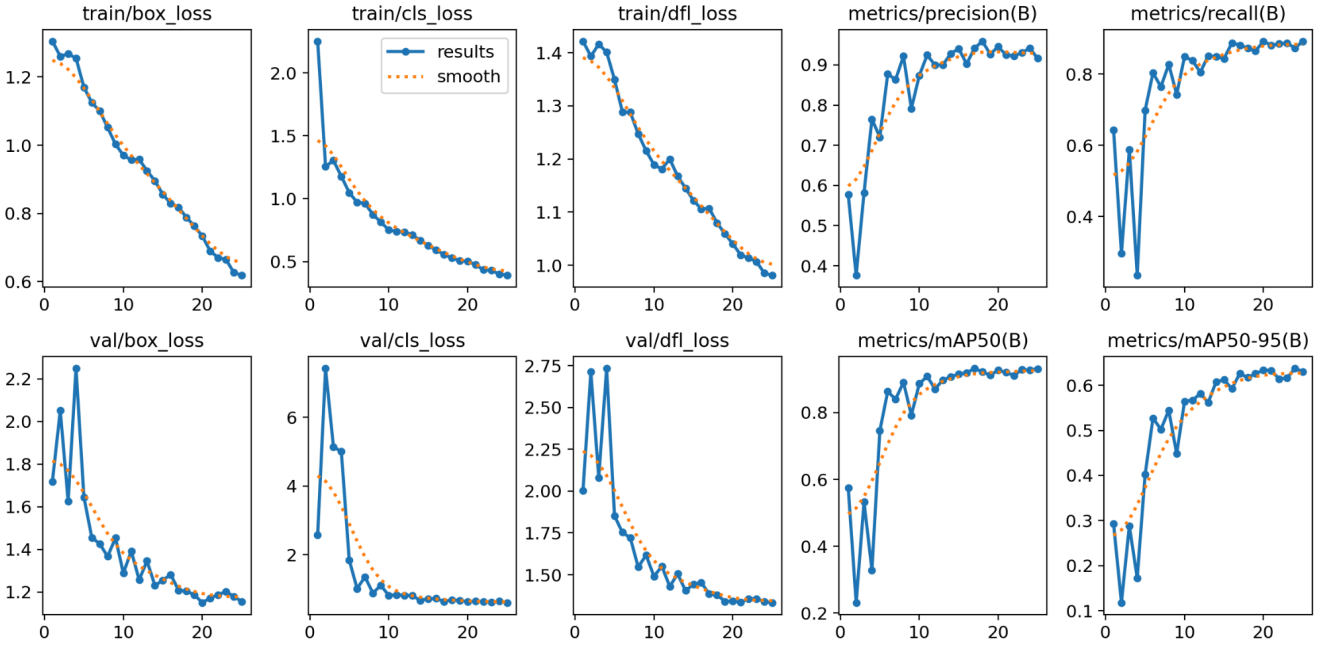


Figure 13. Evaluation metrics of the YOLOv11 model

Above as shown in Figure 12 are the evaluation metrics, though in our case we will be strictly focusing on maP and just need to make sure that the model isn't overfitting which we can see it isn't by the other graphs. maP50 shows the amount of predicted object that has an IoU bigger than 50% and maP50-95 is the same but increasing IoU threshold. A mAP-50 of 85% is an acceptable outcome range with clear model improvements if by using more datasets. This means that on average, across all classes, the model correctly detects objects 85% of the time with a sufficiently accurate bounding box (≥ 0.50 IoU).

```
Ultralytics 8.3.50 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLO11m summary (fused): 303 layers, 20,033,887 parameters, 0 gradients, 67.7 GFLOPs
val: Scanning /content/Judol-Detection-v2-9/valid/labels.cache... 91 images, 0 backgrounds
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all         91        177      0.942      0.874      0.928      0.636
                   BK8         20         39      0.939       0.79      0.843      0.532
      Gate-of-olympus          36         39      0.948      0.928      0.963      0.578
              Princess         19         25       0.89          1      0.978       0.62
    Starlight-Princess         19         21      0.984      0.952      0.969      0.814
                  Zeus         44         53      0.949        0.7      0.888      0.637
Speed: 1.9ms preprocess, 27.1ms inference, 0.0ms loss, 7.0ms postprocess per image
Results saved to runs/detect/val2
```

Figure 14. mAP results from the YOLO model

This is further supported when testing on our test dataset with its mAP50 averaging around 0.928. This is very good, although if compared with the literature **IATEFF-YOLO: Focus on Cow Mounting Detection During Nighttime,** with its 99.3% mAP, they are using a custom built and modified YOLO architecture. So a 92.8% mAP in a pretrained model is pretty good for our project.

Figure 15. sample of a test dataset that was passed to the model.

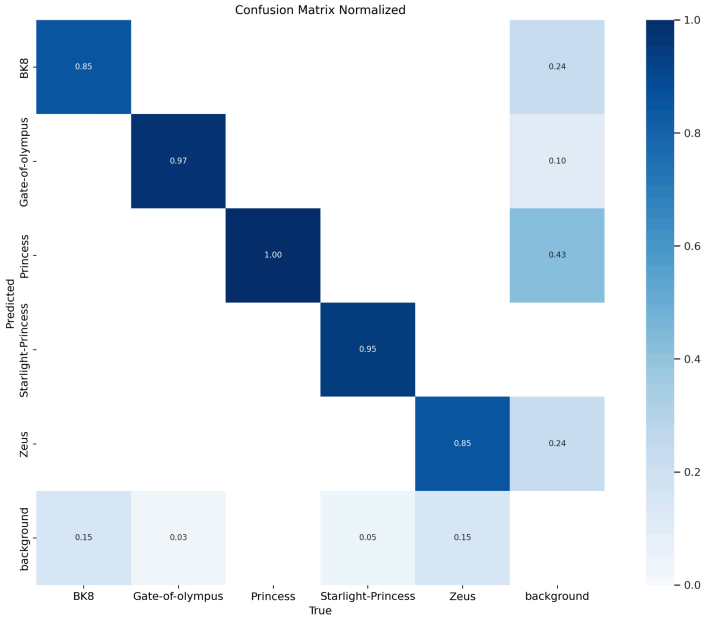


Figure 16. Normalized confusion matrix of our model

RT-DETR(Real Time DEtection TRansformer) Judol RTDETR.ipynb

The Real-Time DEtection TRansformer (RT-DETR) is primarily designed for real time object detection, where speed and accuracy of the identified objects are important. It is based on the idea of DETR (the NMS-free framework), meanwhile introducing conv-based backbone and an efficient hybrid encoder to gain real-time speed. RT-DETR efficiently processes multiscale features by decoupling intra-scale interaction and cross-scale fusion.

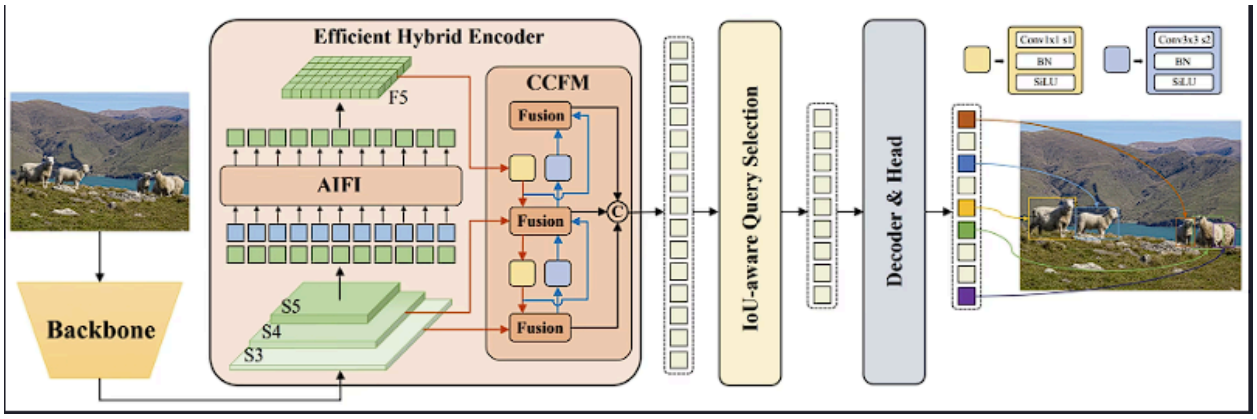


Figure 17. RT-DETR model architecture diagram

The RT-DETR model architecture diagram shows the last three stages of the backbone {S3, S4, S5} as the input to the encoder. The efficient hybrid encoder transforms multiscale features into a sequence of image features through intrascale feature interaction (AIFI) and cross-scale feature-fusion module (CCFM). The IoU-aware query selection is employed to select a fixed number of image features to serve as initial object queries for the decoder. Finally, the decoder with auxiliary prediction heads iteratively optimizes object queries to generate boxes and confidence scores.

Now logically, because RT-DETR is built on Transformers, which excel in capturing long-range dependencies in spatial features. Transformers inherently have global attention mechanisms, enabling the model to capture relationships between all parts of the image simultaneously.

This global context improves detection of occluded objects, small objects, and objects in cluttered scenes. It is also more robust to variations in image size and resolution due to its global attention mechanism and handles complex object relationships better, such as detecting overlapping or interacting objects.

So on paper, RT-DETR will be more accurate than YOLO, and it was showing in our experimentation. But because of it being transformer based, rather than YOLO's CNN architecture, it makes it much more computationally expensive than YOLO in training. Even if RT-DETR is a smaller, more focused on real time version of DETR, training it requires resources. Resources that we unfortunately are limited to (google colab T4 runtime). So our training is limited to 5 epochs only, and in that 5 epochs the model mAP50 has already reached 50%, this means a higher epoch/more arguments will yield better results than YOLO(on paper) but because of our lack of resources, we will just be training it using 5 epochs only.

Our experiments:
We will be importing our dataset in roboflow that were already preprocessed and augmented. The annotations will also be in COCO format which has a .json that contains all of the annotations per split. Our pretrained YOLO11 model will be imported from PekingU/rtdetr_r50vd_coco365_o. As for all of our preprocessing and augmentation is done in Roboflow, we can ignore said steps and just go straight to training.

```
training_args = TrainingArguments(
    output_dir=f"{dataset.name.replace(' ', '-')}-finetune",
    num_train_epochs=5,
    max_grad_norm=0.5,
    learning_rate=3e-5,
    warmup_steps=50,
    per_device_train_batch_size=8,
    dataloader_num_workers=2,
    gradient_accumulation_steps=2,
    fp16=True,
    eval_strategy="steps",
    eval_steps=1000,  #
    save_strategy="epoch",
    save_total_limit=1,
    remove_unused_columns=True,
    load_best_model_at_end=False,
    metric_for_best_model="eval_map",
    greater_is_better=True,
)

```

Figure 18. Training hyperparameters for RT-DETR

These are the hyperparameters we will be using in training our model, again we will be using the maximum training parameters that are acceptable with our resources.

Max_grad_norm is used to clip gradients to prevent exploding gradients during backpropagation. Warmup_steps gradually increases the learning rate during the first few steps to stabilize training, this means lowering the values will be less computationally expensive. Fp16 is used for mixed-precision training to use less memory and speed up computation on GPUs that support it.

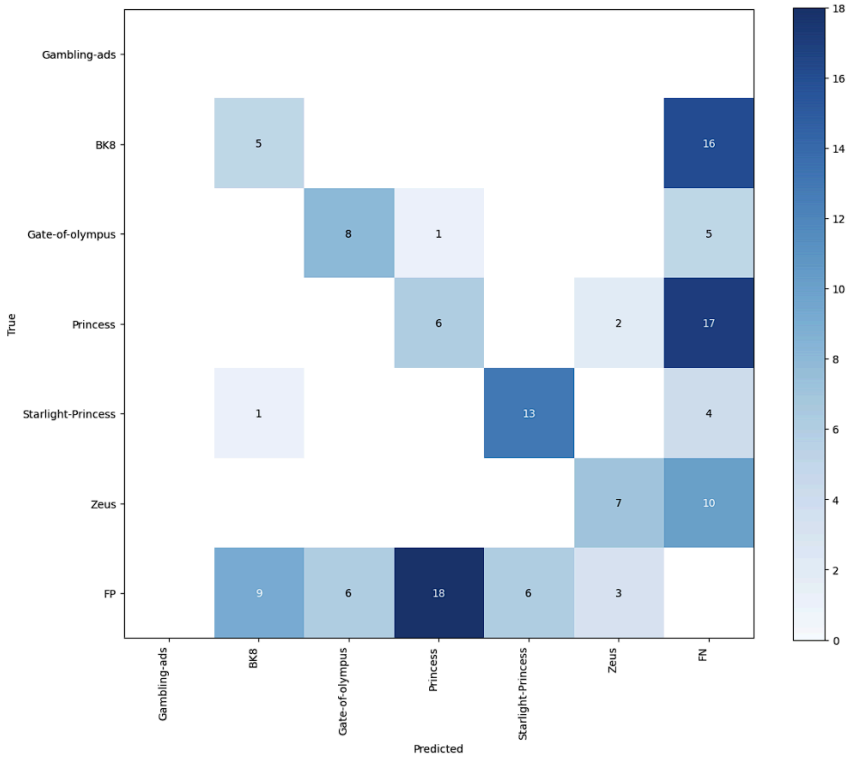


Figure 19. Confusion matrix of RT-DETR

We can see the confusion matrix of our RT-DETR here, its result is definitely not optimal because of our lack of resources to train the model.

Figure 20. Results from RT-DETR's detection

But you can already see the model able to pick up some of the objects albeit not all.
After experimenting, it is obvious our YOLO11 will be our choice of model.
We can look at it at work here in our gradio… Judol Gradio YOLO11.ipynb
Citations:

https://x.com/freakouts4u/status/1872814327453106364
https://x.com/mussi3x/status/1873423244180836662
https://360info.org/indonesia-has-to-hold-or-fold-on-online-gambling-ban/
https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71
https://docs.ultralytics.com/models/yolo11/#overview
https://docs.ultralytics.com/models/rtdetr/
https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-rt-detr-on-custom-dataset-with-transformers.ipynb
https://arxiv.org/pdf/2304.08069