JUDOL DETECTION

PRESENTATION

Presented by :
Christian Antonius Anggaresta - 2602179214
Jonathan Lucas Fontana - 2602159723
Farrel Adyatma Alimin - 2602216906

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.

Literature Review

- Object Detection on Scene Images: A Novel Approach, *Kaushik Das, Arun Kumar Baruah*, 2023
- 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

Purpose & Benefits

1. Prevent Gambling Ads Distribution
2. Monitor Real-Time Gambling Promotions
3. Protect Vulnerable People

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

Methods

Deep Learning

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

Machine Learning

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, and images that are used alongside many online gambling ads to promote them.

Sample Zeus

Sample Starlight Princess

Sample BK8

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.

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

At the end, we will have a total of 936 Training images

Annotating

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

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.

Dataset split

COLOR	CLASS NAME	COUNT
	BK8	203
	Gate-of-olympus	171
	Princess	179
	Starlight-Princess	150
	Zeus	229

Experiment

YOLOv11: 85% (👑 BEST METHOD 👑)

RT-DETR(Real-Time DEtection TRansformer) : 50%

LBPHFaceRecognizer: 20.1%

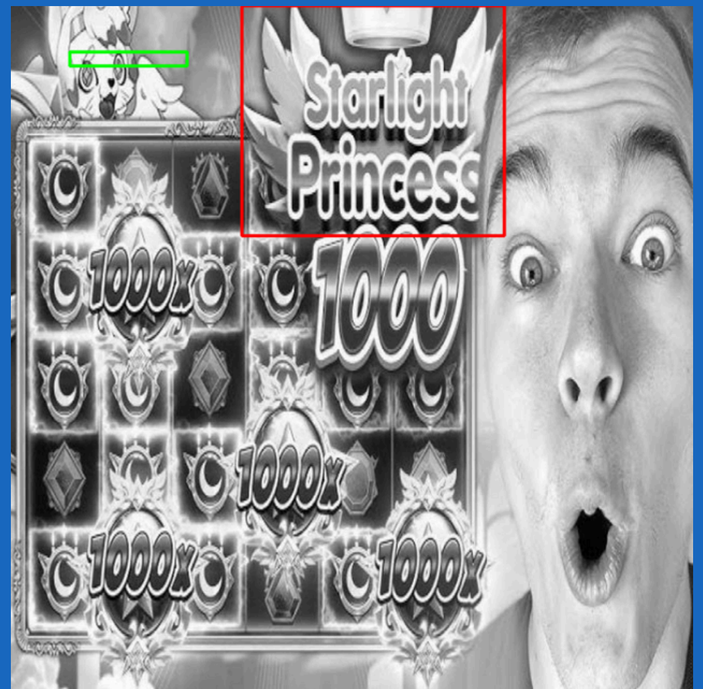
SVM + HOG : 98% (Not representative as basically false positives)

Metric Evaluation =
mAP50 = Mean average Precision 50% (Only true if predicted bounding box covers 50% of label bounding boxes

EXPERIMENT HYPOTHESIS

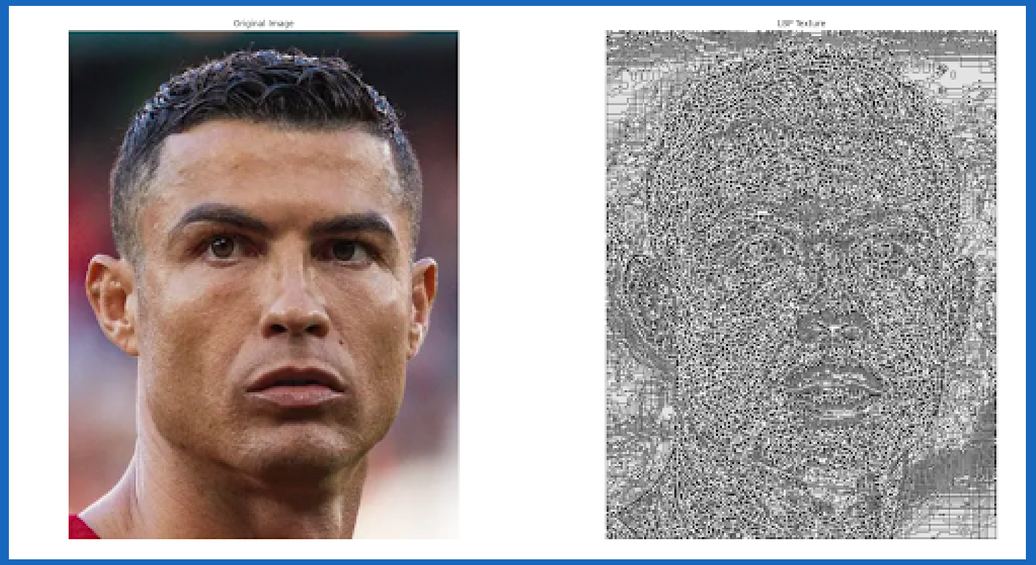
LBPHFaceRecognizer: 20.1%

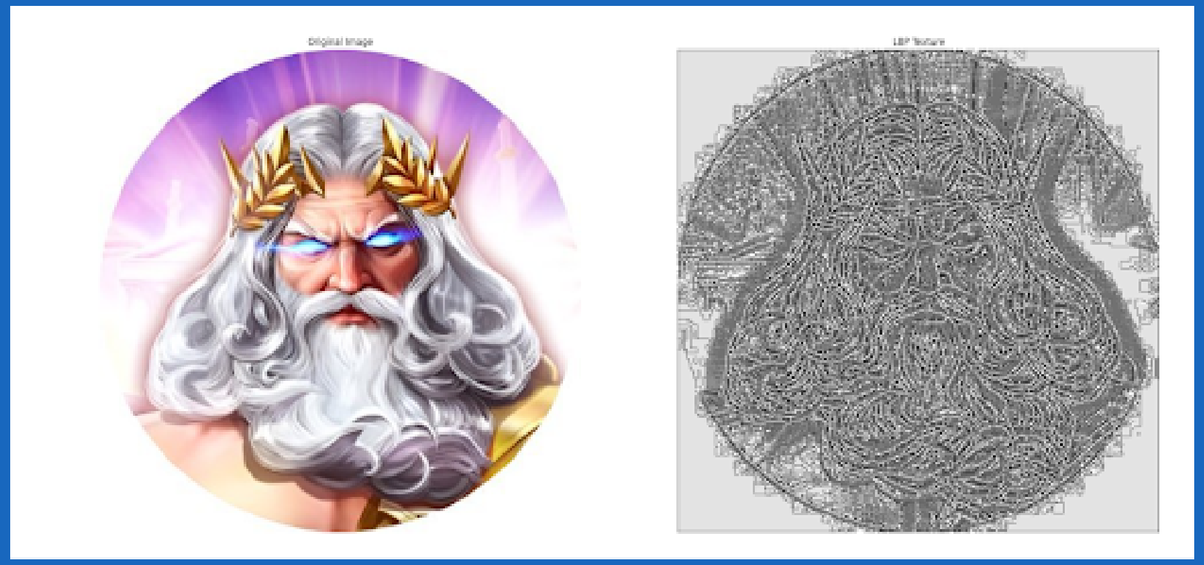
The Local Binary Patterns Histograms (LBPH) Face Recognizer is primarily designed for face recognition tasks, where it identifies patterns and textures unique to human faces. Its low accuracy of 20.1% on the Judol dataset can be attributed to the dataset's diversity, which includes not only faces of 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.



LBPHFaceRecognizer: 20.1%

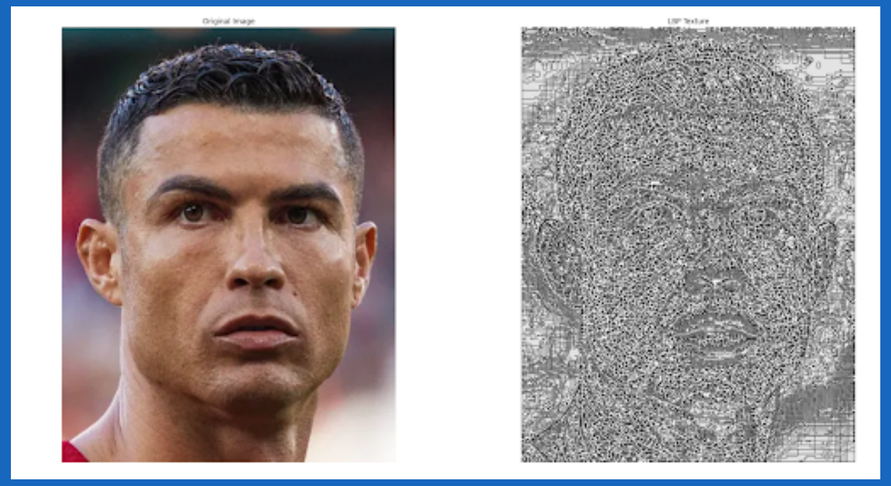
This can be explained because LBPHFaceRecognizer’s feature extractor is fined-tuned to extracting very fine texture-based features, ie. the complex micro-textures found in real human skin. This makes it optimal for real-life faces as human faces have consistent local texture patterns, such as the areas 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.

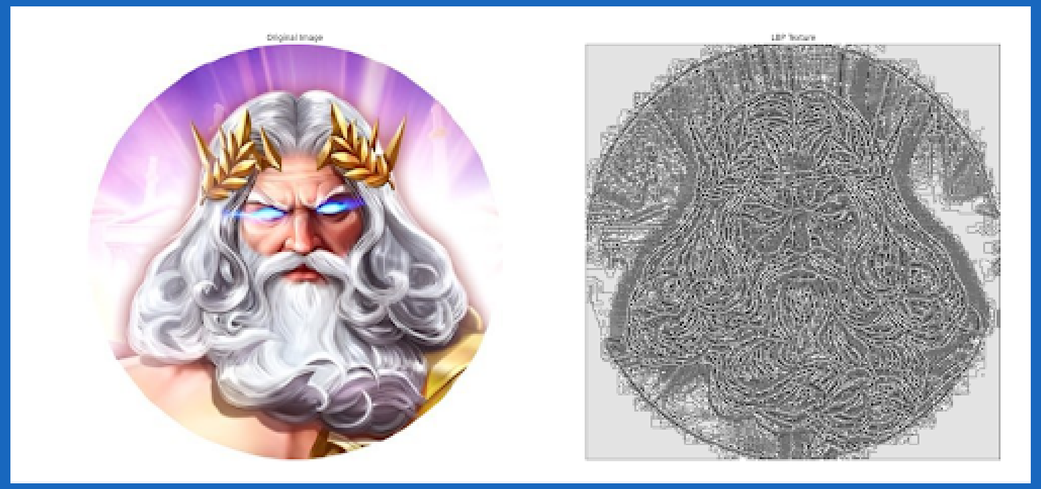




LBPHFaceRecognizer: 20.1%

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.





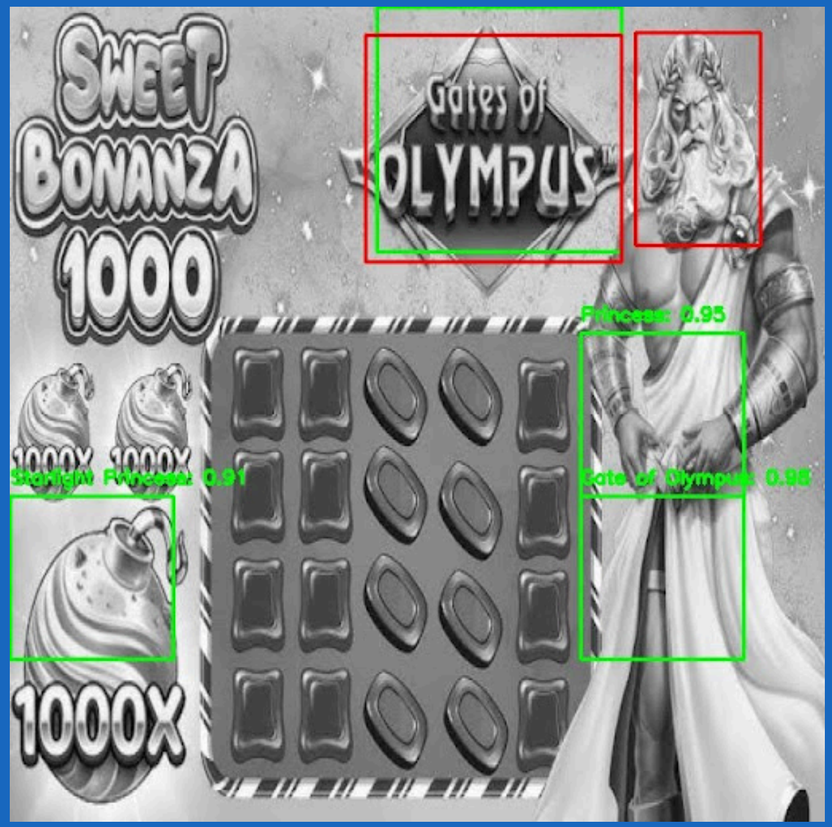
EXPERIMENT HYPOTHESIS

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.

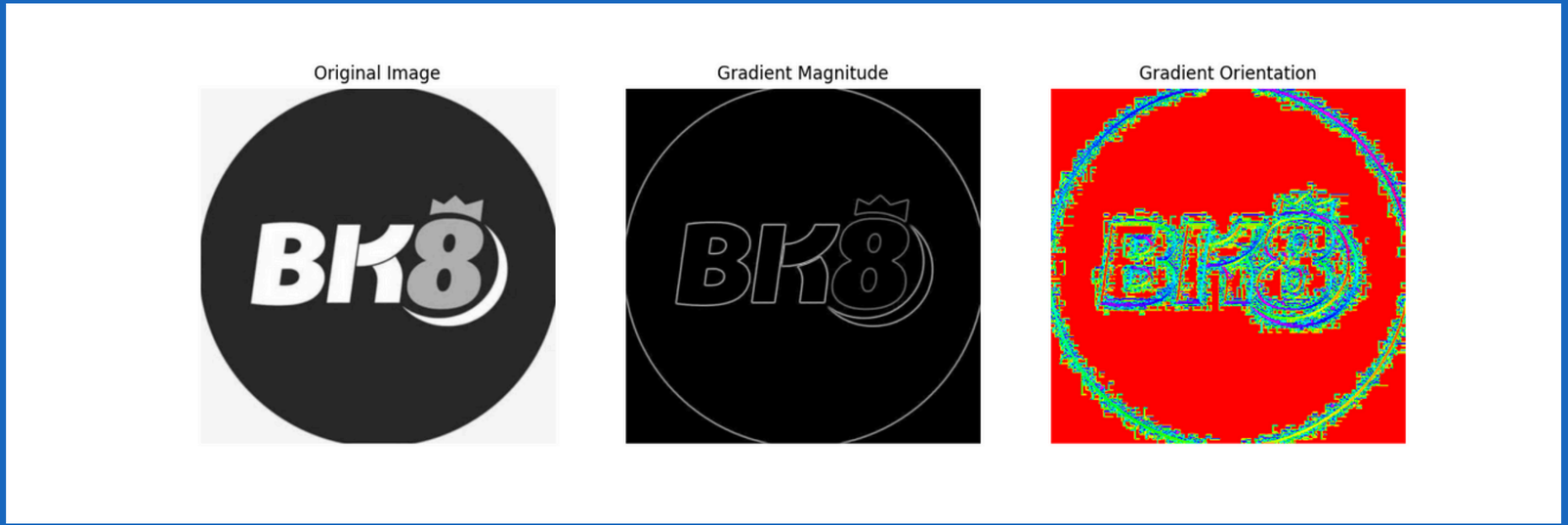
SVM + HOG : 98% (Not representative as basically false positives)

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.



SVM + HOG : 98% (Not representative as basically false positives)

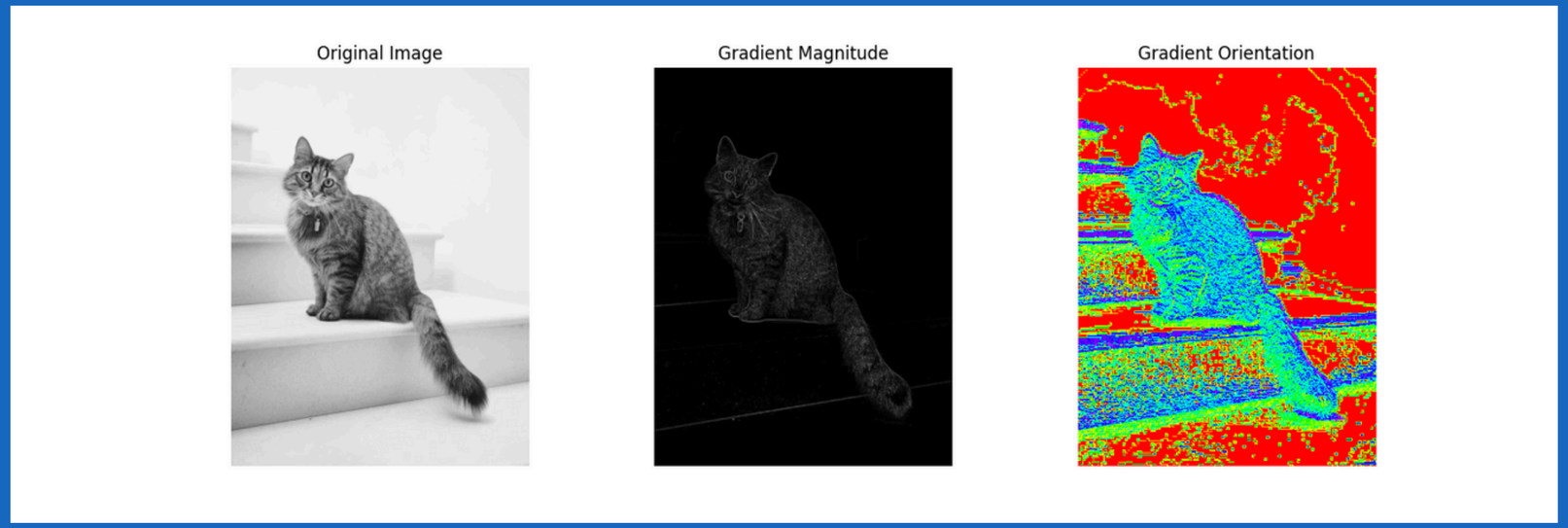
After this then we try to find why HOG and SVM performed poorly, knowing HOG (Histogram of Oriented Gradients) extracts features that describe the shape and structure of objects in an image by analyzing the distribution of gradient magnitudes and orientations. We then try to see the gradient orientation 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 below. 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.



SVM + HOG : 98% (Not representative as basically false positives)

The picture below shows the ideal gradient orientation, the gradient appears smooth compared to the BK8 logo in last page 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.



EXPERIMENT HYPOTHESIS

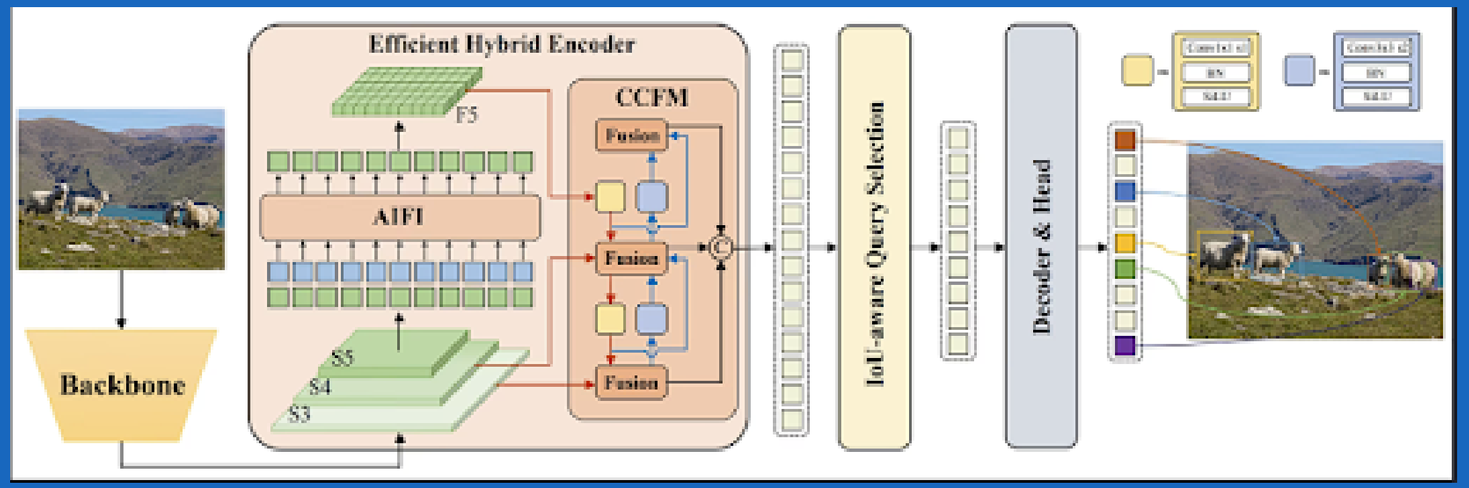
RT-DETR (Real-Time DEtection TRansformer) with transformers: 50%

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.

RT-DETR (Real-Time DEtection TRansformer) with transformers: 50%

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.

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.



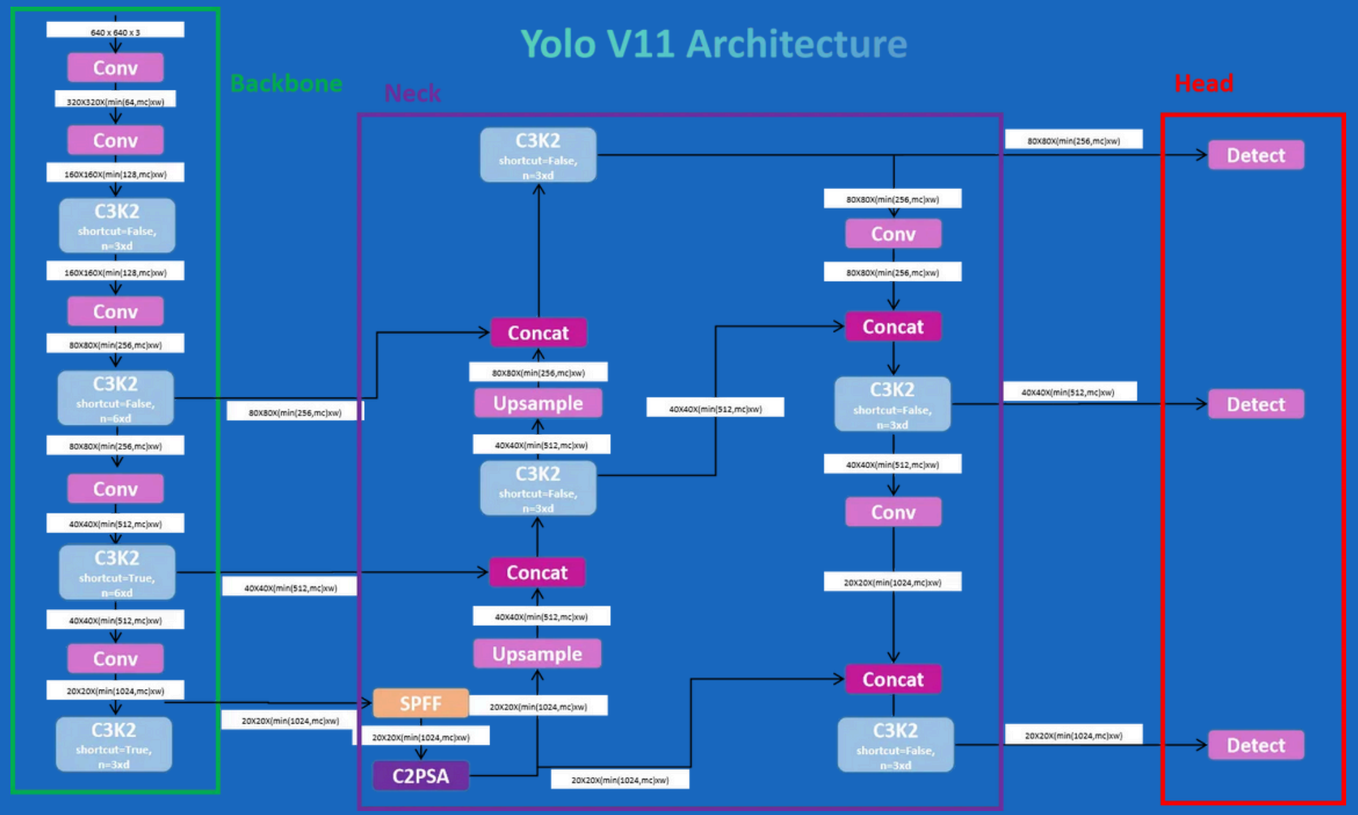
EXPERIMENT HYPOTHESIS

YOLOv11: 85%

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.

YOLOv11: 85%

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.



Yolo Experiment

Model = Yolo11m

The parameters we are currently using:

Optimizer = AdamW

Epochs = 25

Image Size = 640px

Learning Rate for model = 0.001111

weight_decay = 0.0005

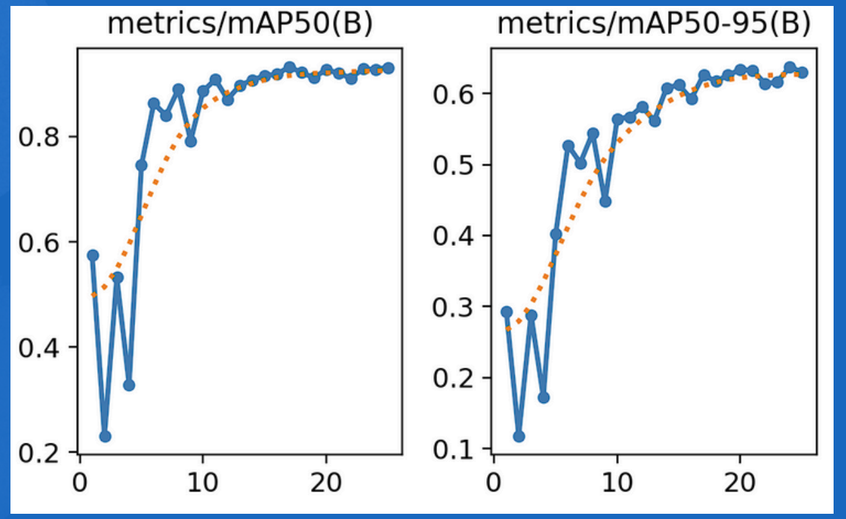
Normalize = /255

```
Validating runs/detect/train2/weights/best.pt...
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
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all         91        177      0.942      0.873      0.928      0.638
                   BK8         20         39      0.939      0.788      0.842      0.534
       Gate-of-olympus         36         39      0.948      0.928      0.963      0.579
              Princess         19         25      0.891          1      0.978       0.62
    Starlight-Princess         19         21      0.984      0.952      0.969      0.818
                  Zeus         44         53      0.949      0.699      0.887      0.637
Speed: 0.2ms preprocess, 12.6ms inference, 0.0ms loss, 3.5ms postprocess per image
```

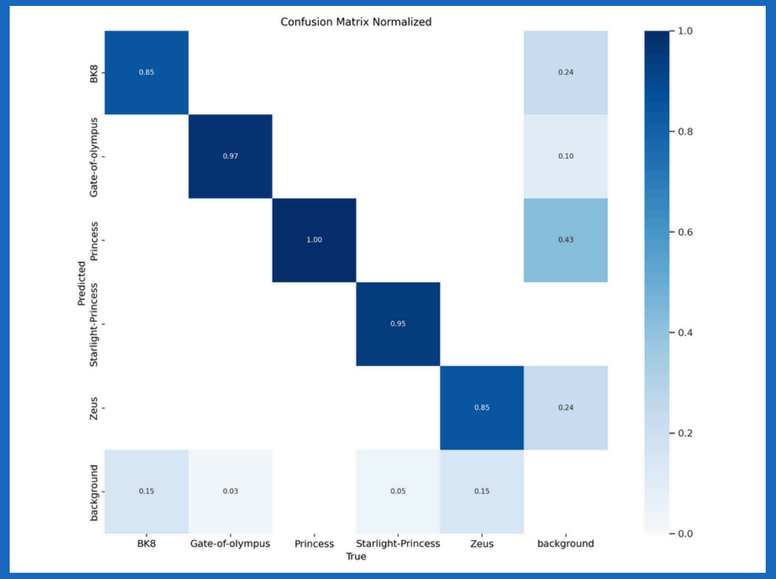
Training evaluations

mAP50 = Mean average Precision 50% (Only true if predicted bounding box covers 50% of label bounding boxes

mAP50-95 = The average mAP from 50 - 95% threshold with a 5% step



Yolo Evaluation




```
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-v2-8/valid/labels.cache... 91 images, 0 backgrounds,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 16
                   all         91        177      0.916      0.899      0.939      0.646
                   BK8         20         39      0.902      0.795      0.855      0.526
      Gate-of-olympus         36         39      0.958      0.949      0.992      0.589
              Princess         19         25      0.821          1       0.97      0.664
    Starlight-Princess         19         21      0.984      0.952      0.971      0.812
                  Zeus         44         53      0.914        0.8       0.91      0.638
Speed: 2.8ms preprocess, 28.6ms inference, 0.0ms loss, 8.2ms postprocess per image
Results saved to runs/detect/val
💡 Learn more at https://docs.ultralytics.com/modes/val
```

Testing evaluations

CONCLUSION

In this project we compare 2 machine learning and 2 deep learning algorithms/model and concluded that for object detection, Deep learning methods are definitely much suited for this task. With YOLO11 being the better pre-trained model with an mAP50 of 85% compared to RT-DETR's mAP50 of 50% in our findings. 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.

For a more detailed explanations of our models, trainings and findings, please refer to the documentation

LINKS

LINK COLAB:

https://colab.research.google.com/drive/1gWZlhKi6S-7mRJtrKyWr3YG9_ejN3Jj6#scrollTo=wpM50m8qW6Mc

LINK VIDEO DEMO:

https://drive.google.com/file/d/1TmRffC8RZKvHr6ZJHfVc_6EhnZxsxZ9H/view?usp=drive_link

LINK DEMO HOST IN HUGGING-FACE:

https://huggingface.co/spaces/JrEasy/Judol_Push_model

LINK DOKUMENTASI YANG LEBIH DETAIL

https://docs.google.com/document/d/1sRhai6kxePchqMiQZSr8qkD4HzZNdu1G4qUgvAjcoGM/edit?tab=t.0

LINK GITHUB:

https://github.com/JrEasy1/Binus-Judol-Detection-Yolo11

Thank you!