CS5242 Mini-project: Weapon Detection

Group 16: GRACE NGU SOOK ERN (A0268281X), LI YUNFAN (A0162493B), LIU TIEDONG (A0189717L), WANG TIANCHI (A0227468U), ZHOU ZIYI (A0262680A)

Objectives

- **Objectives**:
  a. Classify **images** if weapon is present or not
  b. Classify **videos** if weapon is present or not
  c. Draw **bounding boxes** on weapons for images and videos
- **Employment of the obtained knowledge:**
  - Data Preprocessing
  - Data Augmentation: traditional methods (noise, flipping, etc) + image synthesis (GAN)
  - Image Classification: MLP & CNN
- **Exploration and exploitation of the state-of-the-art methods:**
  - Data Augmentation: CNN-based diffusion model with U-net architecture
  - Object Detection: YOLOv7 & YOLOv8

EDA & Pre-processing

- Observations from EDA
  - Unbalanced samples (Weapons > Normal)
  - Different sizes and aspect ratios
  - Smallest height/width: 240x240
  - Weapons mostly in centre parts of the images
- Image Preprocessing (only for CNN & MLP)
  - Crop to the largest centre square
  - Resize to 240x240
- Data Loading
  - Load part of samples from each category
  - Shuffle and further resizing

Preprocess image (before and after):

Image Synthesis

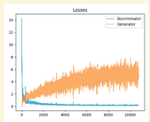
- Image Synthesis using DCGAN and Diffusion Model
- ***Objective***: To balance the dataset by synthesizing image for underrepresented images/ weapons
- ***Similarities & Differences***: Both models are unsupervised generative models that synthesize image based on noise, differences include:

Dissimilar elements	DCGAN	Diffusion Model
Networks Trained	A pair of network	A single network
Synthesizing Mechanism	Min-max game	Maximum likelihood estimation
Sampling process	Non-iterative	Iterative
Image Resolution	Fixed	Adjustable (Depend no. of diffusion steps
Image Interpretability	Non-abstract, focuses on well-defined features	Abstract, focuses on pixel-level data distribution

DCGAN Algorithm

Standard deep convolutional GANs algorithm involving 2 neural networks:

- Generator
- Discriminator
- **Framework**: Tensorflow
- Optimize parameters to achieve optimal min-max game



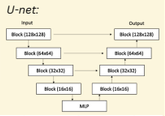
Diffusion Algorithm

Diffusion model comprise of a 1 neural network with 2 processes:

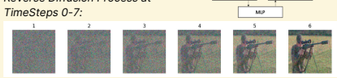
- Forward Diffusion: Add noise to image iteratively, and train U-net, the predicted noisy image is then compared to a true noise.
- Reverse Diffusion: Noise is ran through U-net to predict noise, the difference between noisy image and predicted noise becomes the denoised image.
- Framework: **Tensorflow**

Forward Diffusion Process:

U-net:



Reverse Diffusion Process at TimeSteps 0-7:



Model Architecture & Components

The model components to achieve optimal result for both DCGAN and diffusion model is as follows:

Model Components	DCGAN	Diffusion Model
Architecture	Generator-Discriminator	U-net
Layers	• Generator ○ Input: FC Dense(128), Leaky Relu, Batch Normalization ○ 5 upsampling blocks (Strided Conv2D - 8x8, 16x16, 32x32, 64x64, 128x128), LeakyReLu, Batch Normalization ○ Output: Strided Conv2D(3), Tanh • Discriminator ○ 5 downsampling blocks (128x128, 64x64, 32x32, 16x16, 8x8) ○ Output: Conv2D(1), Sigmoid	• Input: FC Dense(100) • 4 downsampling blocks (Conv2D - 128x128, 64x64, 32x32, 16x16), ReLu, Layer Normalization, Max Pooling • MLP layer: Dense(128), Dense(16x16x128) , ReLu, Layer Normalization • 4 upsampling blocks (16x16, 32x32, 64x64, 128x128) • Output: Conv2D(3)
Padding	All 'Same'	All 'Same'
Optimizer	Adam(LR_D=0.0004, LR_G=0.0005, BETA1=0.8, BETA2=0.9)	Adam(LR=0.0008)
Loss Function	Sigmoid Cross Entropy with Logits, Increase loss weights on discriminator	Mean Absolute Error
Final Epochs	1950 (Training Time: 48 hours)	500 (Training Time: 30 hours)

Model Performance Comparison

3 images were selected to compare the performance between DCGAN and diffusion model. Diffusion model outperforms DCGAN:

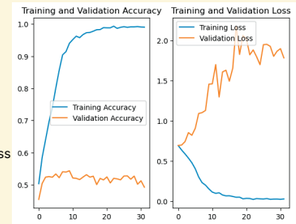
Original Image	DCGAN Output	Diffusion Model Output
MSE	1414.67	136.80
MSE	2344.99	501.34
MSE	1279.23	258.84

Methodology - MLP & CNN

MLP Model Overview

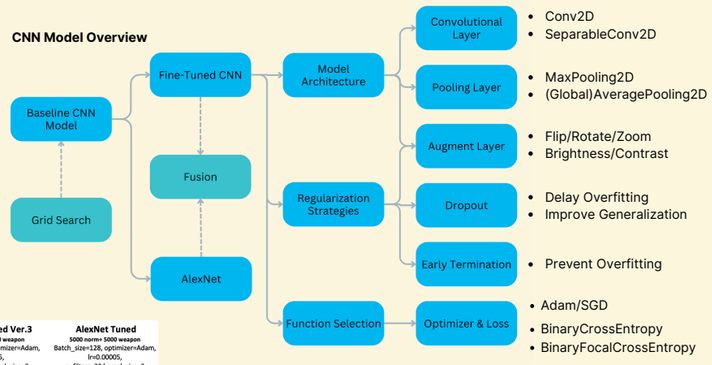
- Training data
  - Resize
  - Random horizontal flips
- Image Patches
  - Size/Strides: 5x5, Rate: 2x2
  - (180, 240, 3) to (1645, 75)
- Dense layers on patches
  - 75 => 36 => dropout => 3
  - 2847 parameters shared across patches
- Flatten
- Dense layers on the flattened array
  - 4935 => 600 => 100 => 1
  - Dropouts of 50% and 20%

- Training Data Size: 4000 images
- Final Accuracy: 54%



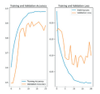
Best Hyperparameters	Baseline Tuned Ver.2 7500 norm+ 7500 weapon	Baseline Tuned Ver.3 7500 norm+ 7500 weapon	AlexNet Tuned 5000 norm+ 5000 weapon
	Batch_size=128, optimizer=Adam, lr=0.001, n_filters=32, kernel_size=3	Batch_size=128, optimizer=Adam, lr=0.0005, n_filters=32,kernel_size=3	Batch_size=128, optimizer=Adam, lr=0.00005, n_filters=32,kernel_size=3

CNN Model Overview



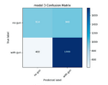
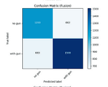
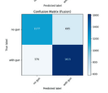
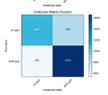
Result - CNN Models

Training & Validation Loss, Accuracy

CNN Models	Loss & Accuracy Graphs	Accuracy
Baseline Tuned Ver.2 7500 norm+ 7500 weapon		0.66
Baseline Tuned Ver.3 7500 norm+ 7500 weapon		0.67
Mid-level Fusion on Ver.3 and AlexNet		0.69
AlexNet Tuned 5000 norm+ 5000 weapon		0.67

- Mid-level fusion shown the highest accuracy with 0.67, which means the model correctly classifies 67% of the images, regardless of the class.
- The mid-level fusion method on Ver.3 and AlexNet has slightly improvement, and the accuracy has increased by 2%.
- After few epochs of training, the CNN model's validation loss starts to increase, and the validation accuracy increase slightly.

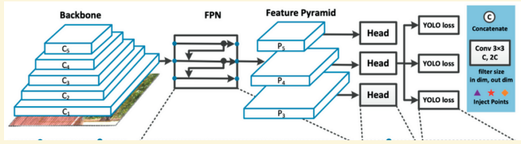
Classification Report & Evaluation Matrix

CNN Models	Precision 0: norm 1: weapon	Recall 0: norm 1: weapon	F1-score 0: norm 1: weapon	Confusion Matrix
Baseline Tuned Ver.2 7500 norm+ 7500 weapon	0: 0.68 1: 0.65 Overall: 0.67	0: 0.49 1: 0.81 Overall: 0.65	0: 0.57 1: 0.72 Overall: 0.65	
Baseline Tuned Ver.3 7500 norm+ 7500 weapon	0: 0.64 1: 0.69 Overall: 0.67	0: 0.64 1: 0.69 Overall: 0.67	0: 0.67 1: 0.69 Overall: 0.67	
Mid-level Fusion on Ver.3 and AlexNet	0: 0.67 1: 0.70 Overall: 0.69	0: 0.63 1: 0.74 Overall: 0.68	0: 0.65 1: 0.72 Overall: 0.69	
AlexNet Tuned 5000 norm+ 5000 weapon	0: 0.67 1: 0.68 Overall: 0.67	0: 0.57 1: 0.76 Overall: 0.66	0: 0.62 1: 0.71 Overall: 0.66	

- Ver.3 and AlexNet has a better performance on class: weapon compared to class: norm.
- Mid-level fusion' s performance is better for both classes.
- The Ver.2 is better at identifying class weapon instances but is more accurate when predicting class norm instances. It is worth noting that the model correctly identified 81% of all the images that have a gun.

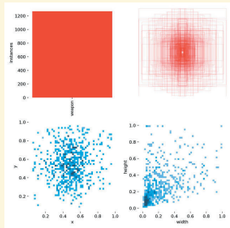
YOLOv7 & YOLOv8

- State-of-the-art object detection algorithms that build upon the YOLO (You Only Look Once) family of models.
- Applied to both images and videos, even real-time streaming due to low latency!
- **YOLOv7: (Jul 2022)**
  - Extended efficient layer aggregation networks (E-ELAN)
  - Several trainable bag-of-freebies methods: planned re-parameterized convolution (RepConvN), Coarse-to-fine lead head guided label assigner, etc.
  - Reduce about 40% parameters and 50% computation and achieve faster inference speed and higher accuracy.
- **YOLOv8: (Jan 2023)**
  - A new backbone network, a new anchor-free split head, and new loss functions.
  - Supports a full range of tasks, including object detection, segmentation, pose estimation, tracking,
- **Framework:** PyTorch



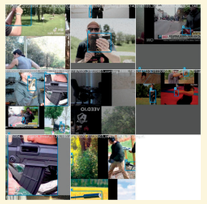
Annotation:

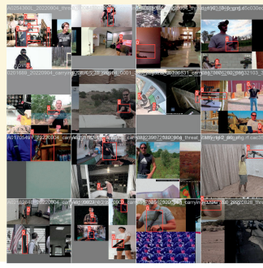
- We annotated ~3000 images in total, 600 images each person, using RoboFlow. Only weapon images are used for training.
- Both guns and knives are labeled as weapons, so we only have one class: weapon (class 0)
- No preprocessing applied.
- Data augmentation: x2
  - Gaussian noise, blur
  - Saturation, brightness
- YOLO has its own data augmentation implemented as bag-of-freebies, i.e., Mosaic, etc.



Training configurations:

- YOLOv7 (left)
  - Initial weight: coco
  - Match size: 8
  - No. of epochs: 500
  - Image resize: 640
- YOLOv8 (right)
  - Model: yolov8s
  - Initial Weight: coco
  - Batch size: 16
  - No. of epochs: 300
  - image resize: 640

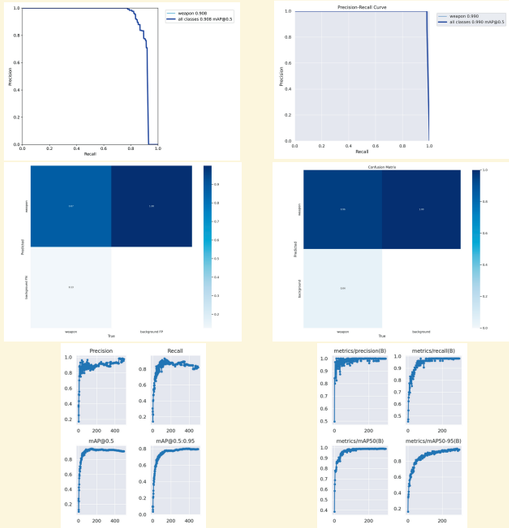




Results - YOLOv7 & YOLOv8

- **Metric**:
  - precision
  - recall
  - Intersection over Union (IoU)
  - mean Average Precision (mAP), mAP@0.5
  - speed/inference time (FPS)

- Training statistics (YOLOv7 left, YOLOv8 right):



- **Showcase of the result (YOLOv7 left, YOLOv8 right, VAL)**:

- **Test set result (best):**

	Images			Videos		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
YOLOv7	0.73	0.74	0.72	0.76	0.78	0.74
YOLOv8	0.84	0.85	0.84	0.86	0.88	0.85

For video detection, we manually set a threshold 20% of total frames that has weapons detected as positive (conf=0.01).
Inference speed: 30-40fps

Innovations

- **Transfer learning**.
  - Coco dataset -> Collections of several Kaggle gun detection datasets -> training dataset

- **Semi-supervised learning:**
  - Train a teacher model on ~1000 manually annotated images from training set.
  - Generate pseudo labels on other images from training set.
  - Enlarge dataset using data augmentation
  - Train a student model on the union of labeled dataset.

Evaluation

- We show that our CNN / MLP tends to use background information for classification instead of focusing on small objects like gun / knife.
- This demonstrates the advantage of YOLO algorithm, as the backbone and feature pyramid networks (FPN) can capture both fine-grained and coarse-grained features at multiple scales, which is important for detecting objects of different sizes.

Conclusion

- We have applied what we have learned in class: CNN, MLP, which serves as a baseline.
- We have also explored GAN as a data augmentation method. Due to its low resolution, we continued to explore the popular diffusion model for data augmentation.
- We applied one of the sota methods in object detection, YOLOv7 and YOLOv8. The methods can also be used for videos.
- We compared and evaluated their performances.

References

[1] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). *Image-to-image translation with conditional adversarial networks*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 5967-5976).
[2] Dhariwal, P. and Nichol, A. (2021) *Diffusion models beat gans on image synthesis, arXiv.org*. Available at: https://arxiv.org/abs/2105.05233 .
[3] Xiao, Z., Kreis, K. and Vahdat, A. (2022) *Tackling the generative learning trilemma with Denoising Diffusion Gans, arXiv.org*. Available at: https://arxiv.org/abs/2112.07804.
[4] Alom, M.Z. et al. (2018) *The history began from Alexnet: A comprehensive survey on Deep Learning Approaches, arXiv.org*. Available at: https://arxiv.org/abs/1803.01164.
[5] Terven, J. and Cordova-Esparza, D. (2023) *A comprehensive review of Yolo: From Yolov1 to yolov8 and beyond, arXiv.org*. Available at: https://arxiv.org/abs/2304.00501.
[6] Wang, C. Bochkovskiy, A. and Liao, H. (2022) YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, arXiv.org. Available at: https://arxiv.org/abs/2207.02696.
[7] https://github.com/ultralytics/ultralytics