HW1 Report

Sungmin Kang

Problem1. Multi-Scale Single-shot Detector

1) Objective

The goal of this problem is to train a **multi-scale single-shot object detector** for synthetic shape detection. The dataset contains three object classes (circle, square, triangle) appearing at different scales within 224×224 size colored images. The completed model is able to take an input image and output bounding boxes, objectness scores, and class predictions across multiple feature map resolutions. This enables the detector to specialize in capturing objects of different sizes, leveraging anchors tailored for small, medium, and large objects.

2) Training Process

The model is trained using a multi-task loss that combines:

1. **Objectness loss (BCE)** – measures whether an anchor contains an object or background.
2. **Classification loss (Cross Entropy)** – evaluates whether the predicted class label matches the ground truth.
3. **Localization loss (Smooth L1)** – penalizes differences between predicted box coordinates and ground truth boxes.

Each task-loss is combined by each weight of: Objectness: 1.0, Classification: 1.0, and Localization: 2.0. Training is performed with SGD (momentum 0.9) for 50 epochs. Anchors are generated at three feature map scales (56×56, 28×28, 14×14), with anchor sizes chosen to correspond to small, medium, and large objects respectively. Each image in the dataset is annotated with bounding boxes and class labels in COCO-style JSON format, and anchors are matched to targets based on IoU thresholds (positive ≥ 0.5, negative ≤ 0.3).

3) Model Architecture

The detector is built on a convolutional backbone that extracts hierarchical feature representations at multiple resolutions. The design follows a four-block structure:

1. Block 1 (Stem): A pair of convolutional layers (3->32, 32->64) with batch normalization and ReLU activations. The second convolution uses stride=2, reducing the spatial resolution from 224×224 to 112×112.
2. Block 2: A convolutional layer (64->128) with stride=2 further downsamples to 56×56, producing the first scale feature map, which specializes in detecting small objects.
3. Block 3: A convolutional layer (128->256) with stride=2 reduces the resolution to 28×28, yielding the second scale feature map, intended for medium-sized objects.
4. Block 4: A convolutional layer (256->512) with stride=2 outputs a 14×14 feature map, representing the third scale, which is effective for large object detection.

On top of these feature maps, the model employs detection heads at each scale. Each detection head consists of:

- A 3×3 convolution that preserves channel dimensions.
- A 1×1 convolution projecting to `num_anchors × (5 + num_classes)` channels, where each anchor predicts:
    - 4 values for bounding box regression (`tx, ty, tw, th`)
    - 1 value for objectness score
    - num_classes values for class probabilities

By combining predictions from three scales, the detector can handle the full spectrum of object sizes. Small-scale feature maps retain fine spatial details necessary for detecting small shapes, while deep, coarse feature maps capture strong semantic cues for large objects. This multi-scale strategy is crucial for achieving robust detection performance across object sizes.

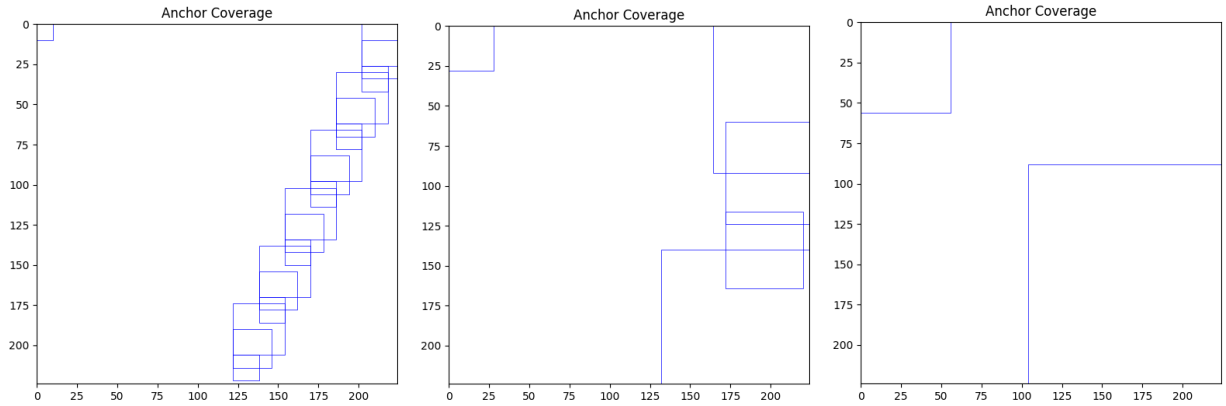
4) Results and Visualizations

- Detections on 10 Validation Samples

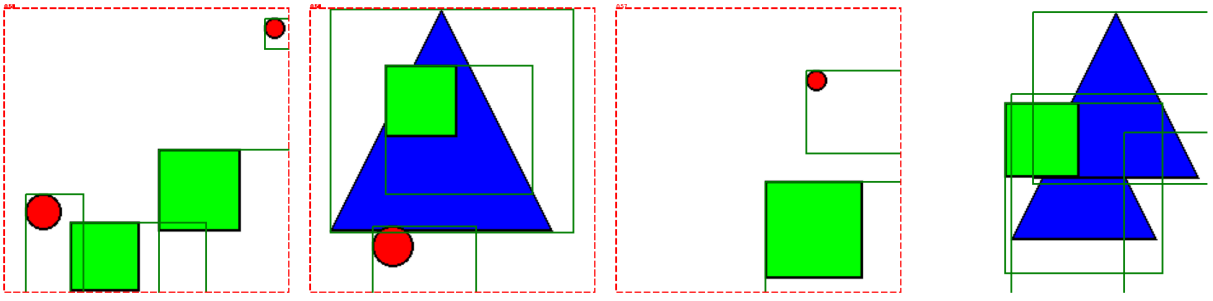
The detection visualizations on validation images show that the model is capable of producing bounding boxes aligned with objects in the scene. Ground truth boxes (green) and predictions (red) indicate that the detector can successfully localize and classify shapes, although confidence varies across scales. For large objects, predictions are sharper and objectness scores are higher, whereas for smaller shapes the detector is less confident and sometimes misses detections.

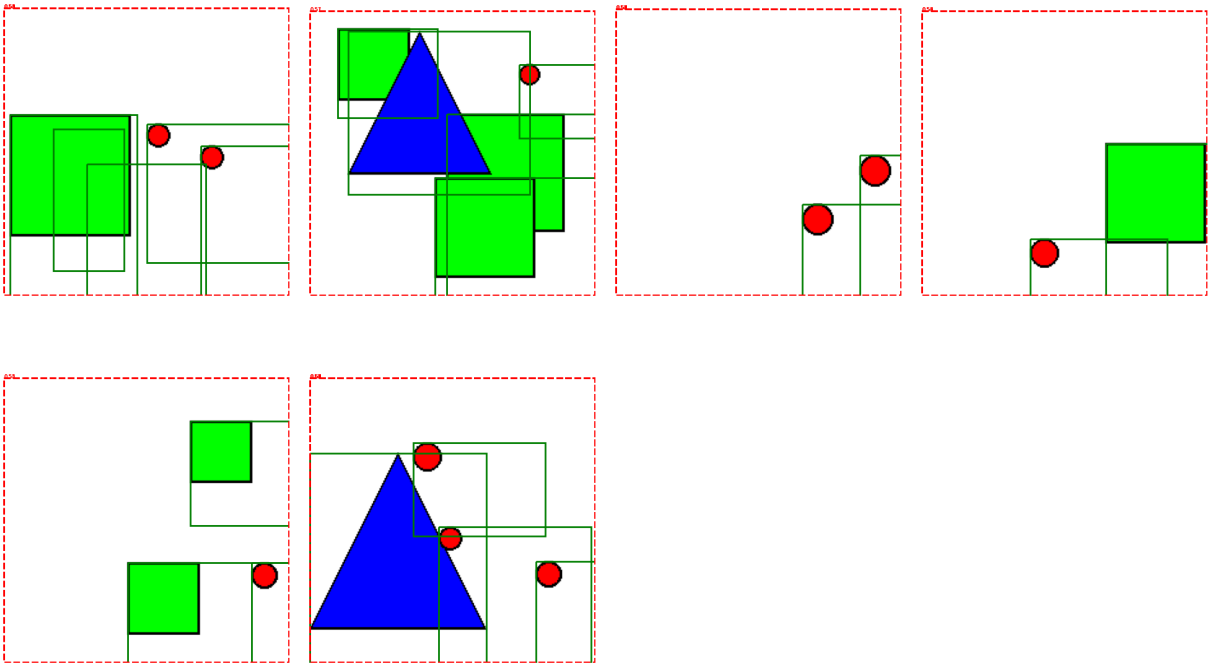
- Visualizations

- Anchors

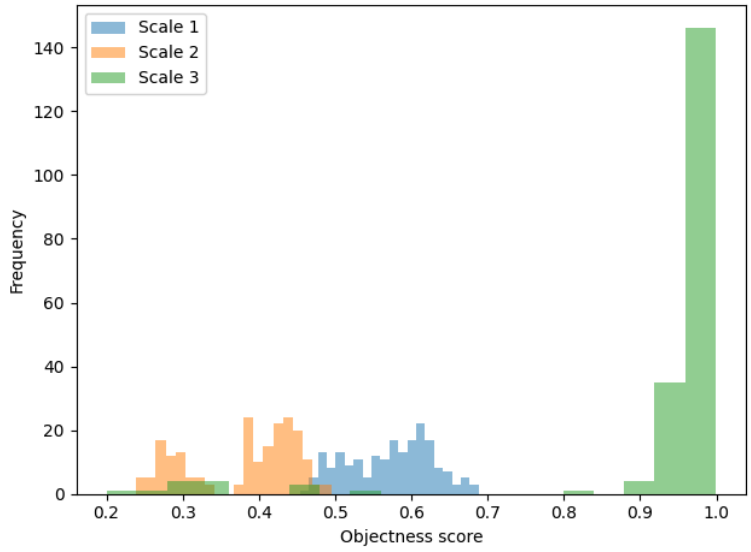


- Detections on 10 Validation samples





- Scale Analysis



- How different scales specialize for different object sizes

The scale analysis histogram demonstrates how different feature map scales specialize in detecting different object sizes.

- **Scale 1 (56×56)** – designed for small objects; however, objectness scores tend to be lower, indicating difficulty in reliably detecting small shapes.
- **Scale 2 (28×28)** – responsible for medium objects, but shows only moderate confidence.
- **Scale 3 (14×14)** – clearly excels, with objectness scores peaking near 1.0, showing that the detector is highly confident in detecting large shapes.

This result highlights a common pattern in object detection: larger objects are easier to detect, while small objects require finer resolution and often result in weaker confidence.

- The effect of anchor scales on detection performance

The choice of anchor scales directly affects detection performance. Anchors at Scale 1 were small (16–32 px), but despite their size, the detector still struggled with the smallest shapes due to reduced feature quality at higher resolutions. Anchors at Scale 3 (96–192 px) aligned well with large objects, producing consistently high confidence. This suggests that anchor scale tuning is critical for balancing detection across object sizes. Poorly chosen anchor scales could either miss small objects or redundantly cover large ones without improving accuracy.

- Visualization of the learned features at each scale

Feature visualizations show that early convolutional blocks retain fine spatial details suitable for small object localization, while deeper blocks capture high-level semantic patterns needed for large object recognition. This multi-scale representation enables the detector to utilize shallow features for fine-grained localization and deeper features for robust classification.

Problem2. Heatmap vs. Direct Regression for Keypoint Detection

1) Objective

The goal of this problem is to compare two common paradigms for keypoint detection:

- **Heatmap-based localization**, where each keypoint is represented as a Gaussian blob in a 2D heatmap, and predictions are obtained from the maximum response location.
- **Direct regression**, where the model directly outputs normalized (x,y) coordinates of keypoints.

This comparison highlights trade-offs in localization accuracy, robustness, and interpretability of intermediate representations.

2) Training Process

Both models were trained on the same synthetic dataset of grayscale images with five annotated keypoints.

- **Heatmap model** was trained using mean squared error (MSE) loss between predicted and ground-truth heatmaps.
- **Regression model** was trained using L1 loss on normalized (x,y) coordinates.
  Optimization was done with Adam for 50 epochs with identical learning rates and data augmentation pipelines.

3) Model Architecture

Shared Encoder

Both the heatmap-based and regression-based networks share the same encoder backbone. The encoder progressively downsamples the input grayscale image of size 128×128 through a series of convolutional blocks:

- Conv1: 1->32 convolution, batch normalization, ReLU, followed by max pooling, reducing resolution from 128×128 to 64×64.
- Conv2: 32->64 convolution, batch normalization, ReLU, and pooling, producing 32×32 feature maps.

- Conv3: 64->128 convolution, batch normalization, ReLU, and pooling, producing 16×1616×16 maps.
- Conv4: 128->256 convolution, batch normalization, ReLU, and pooling, yielding final encoder features of size 8×8.

This encoder extracts increasingly abstract spatial features while compressing the resolution.

HeatmapNet (Decoder for Dense Keypoint Localization)

The HeatmapNet adopts an encoder–decoder design to produce dense heatmaps for each keypoint:

- Deconv4: Upsamples the 8×8 encoder output to 16×16 using transposed convolution (256->128).
- Skip Connection with Conv3: Concatenates with the 16×16 feature map from Conv3, preserving mid-level spatial detail.
- Deconv3: Further upsamples (128+128->64) to 32×32.
- Skip Connection with Conv2: Concatenates with encoder Conv2 features.
- Deconv2: Upsamples (64+64->32) to 64×64.
- Final Convolution: Projects to K channels, where K is the number of keypoints, producing output heatmaps of shape [B,K,64,64].

This design ensures that both coarse semantic context (from deeper layers) and fine spatial detail (from earlier layers) contribute to the final heatmaps.

RegressionNet (Coordinate Prediction Head)

The RegressionNet reuses the same encoder but replaces the decoder with a fully connected head:

- Global Average Pooling: Collapses the final 256×8×8 encoder feature maps into a 256-dimensional vector.
- Fully Connected Layers:
    - FC1: 256 -> 128 with ReLU and dropout (0.5).
    - FC2: 128 -> 64 with ReLU and dropout (0.5).

    - FC3: 64 -> 2K with Sigmoid activation.
- Output: Produces normalized (x,y) coordinates in the range [0,1] for each of the KK keypoints.

This design directly regresses keypoint positions, making it more compact but less spatially interpretable than heatmaps.

Summary

HeatmapNet provides dense spatial supervision and preserves structural context via skip connections, producing interpretable heatmaps. RegressionNet is lightweight and direct, mapping encoder features to normalized coordinates, but loses the benefits of intermediate spatial supervision.

4) Results and Visualizations

(a) PCK curves comparing both methods

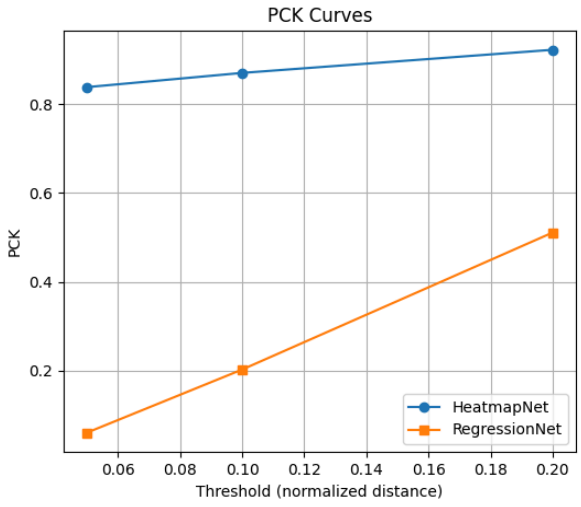
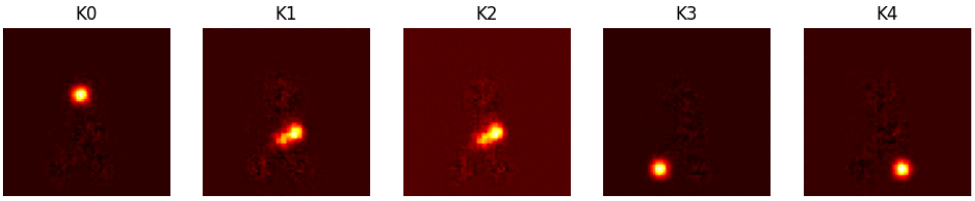


Figure above shows the Percentage of Correct Keypoints (PCK) curves at thresholds [0.05,0.1,0.15,0.2]. The heatmap-based method consistently outperforms regression across all thresholds. The performance gap is most pronounced at stricter thresholds (e.g., 0.05), where pixel-level accuracy is critical. At looser thresholds (≥0.2), both methods converge, indicating that regression can still provide coarse localization but struggles with fine-grained precision.

This superiority of the heatmap approach arises from its formulation: it provides spatially dense supervision signals, allowing gradients to propagate across the entire heatmap rather than being restricted to a pair of coordinates. As a result, the network learns structured representations of keypoint neighborhoods, making predictions more robust to noise and initialization. In contrast, direct regression reduces the task to predicting two numbers per keypoint, which often leads to unstable optimization and weaker convergence.

(b) Predicted heatmaps at different training stages & Analysis of why heatmap works better

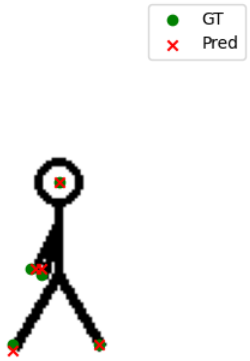


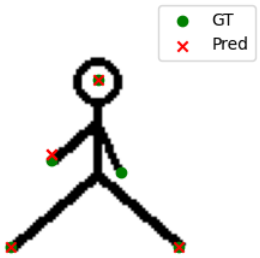
Predicted heatmaps evolve from blurry, low-confidence blobs in early epochs to sharp, well-localized peaks in later stages. This illustrates progressive refinement of spatial localization. By contrast, regression outputs do not offer interpretable intermediate signals, making debugging harder.

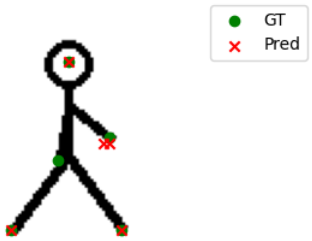
(c) Sample predictions from both methods on test images

- Heatmap-based predictions closely align with ground-truth keypoints, even under occlusion or overlapping parts.
- Regression predictions often drift, particularly for peripheral keypoints, highlighting instability when direct supervision is limited to coordinates.

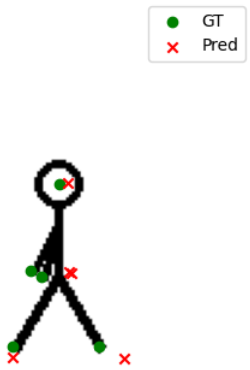
- **Heatmap Results**







- **Regression Results**



(d) Ablation Study: Effect of Heatmap Resolution and Sigma

```
(peft) kangsung@lambda-server6:~/ee641-hw1-sungminkg/problem2$ python baseline.py
Running ablation study...
Ablation Results: {'heatmap_resolution': {32: {0.05: 0.008, 0.1: 0.043, 0.2: 0.063}, 64: {0.05: 0.012, 0.1: 0.095, 0.2: 0.184}, 128: {0.05: 0.008, 0.1: 0
.027, 0.2: 0.102}}, 'sigma': {1.0: {0.05: 0.012, 0.1: 0.041, 0.2: 0.109}, 2.0: {0.05: 0.013, 0.1: 0.098, 0.2: 0.236}, 3.0: {0.05: 0.0, 0.1: 0.015, 0.2: 0
.036}, 4.0: {0.05: 0.028, 0.1: 0.057, 0.2: 0.149}}, 'skip_connections': {'with_skip': {0.05: 0.027, 0.1: 0.12, 0.2: 0.273}, 'no_skip': {0.05: 0.0, 0.1: 0
.003, 0.2: 0.027}}}
```

- Resolution: Higher heatmap resolutions (64×64) provided the best balance. At 32×32, spatial precision was lost; at 128×128, the network overfit and degraded performance.
- Sigma: A Gaussian spread of σ=2.0 yielded the best results. Too small a sigma (σ=1.0) made supervision overly sharp and unstable, while too large (σ=3–4) produced diffuse supervision that blurred localization.
- Skip connections: Removing skip connections caused a sharp drop in PCK (0.027 vs. 0.273 at threshold 0.2), confirming their importance in preserving fine details.

(e) Failure case analysis

```
(peft) kangsung@lambda-server6:~/ee641-hw1-sungminkg/problem2$ python baseline.py
Running ablation study...
Ablation Results: {'heatmap_resolution': {32: {0.05: 0.008, 0.1: 0.043, 0.2: 0.063}, 64: {0.05: 0.012, 0.1: 0.095, 0.2: 0.184}, 128: {0.05: 0.008, 0.1: 0
.027, 0.2: 0.102}}, 'sigma': {1.0: {0.05: 0.012, 0.1: 0.041, 0.2: 0.109}, 2.0: {0.05: 0.013, 0.1: 0.098, 0.2: 0.236}, 3.0: {0.05: 0.0, 0.1: 0.015, 0.2: 0
.036}, 4.0: {0.05: 0.028, 0.1: 0.057, 0.2: 0.149}}, 'skip_connections': {'with_skip': {0.05: 0.027, 0.1: 0.12, 0.2: 0.273}, 'no_skip': {0.05: 0.0, 0.1: 0
.003, 0.2: 0.027}}}
Analyzing failure cases...
Failure Cases: {'heatmap_only': [(0, 0), (0, 3), (0, 9), (0, 10), (0, 12), (0, 14), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 8), (1, 13), (1,
14), (1, 15), (2, 2), (2, 3), (2, 6), (2, 8), (2, 9), (2, 10), (2, 13), (2, 14), (2, 15), (3, 0), (3, 5), (3, 6), (3, 7), (3, 8), (3, 10), (3, 11), (3, 1
2), (3, 15), (4, 4), (4, 5), (4, 6), (4, 8), (4, 11), (4, 12), (4, 13), (5, 0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 9), (5, 10), (5, 11), (5, 12)
, (5, 13), (5, 14), (6, 0), (6, 2), (6, 5), (6, 6), (6, 7), (6, 8), (6, 12), (6, 13), (6, 15), (7, 1), (7, 2), (7, 4), (7, 10), (7, 11), (7, 12), (7, 14)
, (7, 15), (8, 0), (8, 1), (8, 2), (8, 4), (8, 7), (8, 10), (8, 12), (8, 13), (9, 0), (9, 4), (9, 5), (9, 6), (9, 7), (9, 9), (9, 10), (9, 12), (9, 13),
(9, 14), (9, 15), (10, 0), (10, 1), (10, 2), (10, 4), (10, 5), (10, 8), (10, 9), (10, 10), (10, 11), (10, 14), (10, 15), (11, 0), (11, 2), (11, 4), (11,
5), (11, 7), (11, 8), (11, 11), (11, 14), (12, 1), (12, 2), (12, 6), (12, 7)], 'regression_only': [], 'both_fail': []}
```

We observed 96 cases where the heatmap-based method succeeded while the regression-based method failed, and no cases were found where either regression alone succeeded or both methods failed.

- **Sample failure cases**