Appendix

A1. YOLOv5 structure

The concept of *YOLO* suggests utilizing an end-to-end neural network for simultaneous predictions of both bounding boxes and class probabilities. This sets it apart from earlier methods in object detection that repurpose classifiers for detection tasks. Through its distinctive approach to object detection, *YOLO* achieved exceptional outcomes, surpassing the performance of competing real-time object detection techniques by a substantial margin [43]. *YOLOV5* builds upon the foundations of *YOLOV1* through *YOLOV4*. Consistent enhancements have propelled it to achieve exceptional results on two authoritative object detection datasets: Pascal *VOC* (visual object classes) [44] and Microsoft *COCO* (common objects in context) [45].

The network architecture of *YOLOV5* is illustrated in Fig. A1 and comprises three essential components: the backbone, the neck, and the head. Initially, input data is fed into *CSPDarknet*, serving as *YOLOV5*'s backbone, for the extraction of crucial features. These extracted features are then passed on to path aggregation network (*PANet*), which constitutes the neck of the structure. *PANet* intelligently merges and combines these features. Finally, the *Yolo* layer, acting as the head of the architecture, is responsible for generating vital detection outcomes. These outcomes entail class identification, score assignment, precise location pinpointing, and size determination [46].

This specific architectural choice was made for several significant reasons. First and foremost, it incorporates the cross-stage partial network (*CSPNet*) into *Darknet*, resulting in *CSPDarknet* as its core structure. This innovative integration effectively tackles the challenge of redundant gradient information within extensive backbones. By incorporating gradient variations into the feature map, it streamlines the model's parameters and reduces floating-point operations per second (*FLOPS*). As a result, this optimization not only enhances inference speed and accuracy but also reduces the overall model size. This is particularly helpful for the precise detection of nanoparticles, as it mitigates potential adverse effects on results while minimizing computational costs [47].

Furthermore, within *YOLOV5*, the *PANet* is seamlessly integrated as its neck component, aimed at optimizing data flow. *PANet* introduces a pioneering feature pyramid network (*FPN*) architecture, thereby bolstering the foundational path for feature extraction. This enhancement serves to amplify the transfer of information from lower-level features. Concurrently, adaptive feature pooling establishes intricate connections between feature grids across all levels, efficiently guiding essential data from each level towards subsequent subnetworks. Through the heightened utilization of precise localization cues in the lower layers, *PANet* substantially advances the accuracy of object localization [48].

Finally, at the core of *YOLOV5* is the *Yolo layer*, responsible for producing feature maps of three different dimensions (18×18, 36×36, 72×72). This exceptional feature empowers the model to efficiently process objects of diverse sizes, entaling everything from tiny elements to substantial entities. This versatility proves crucial when identifying objects that vary greatly in scale [49].

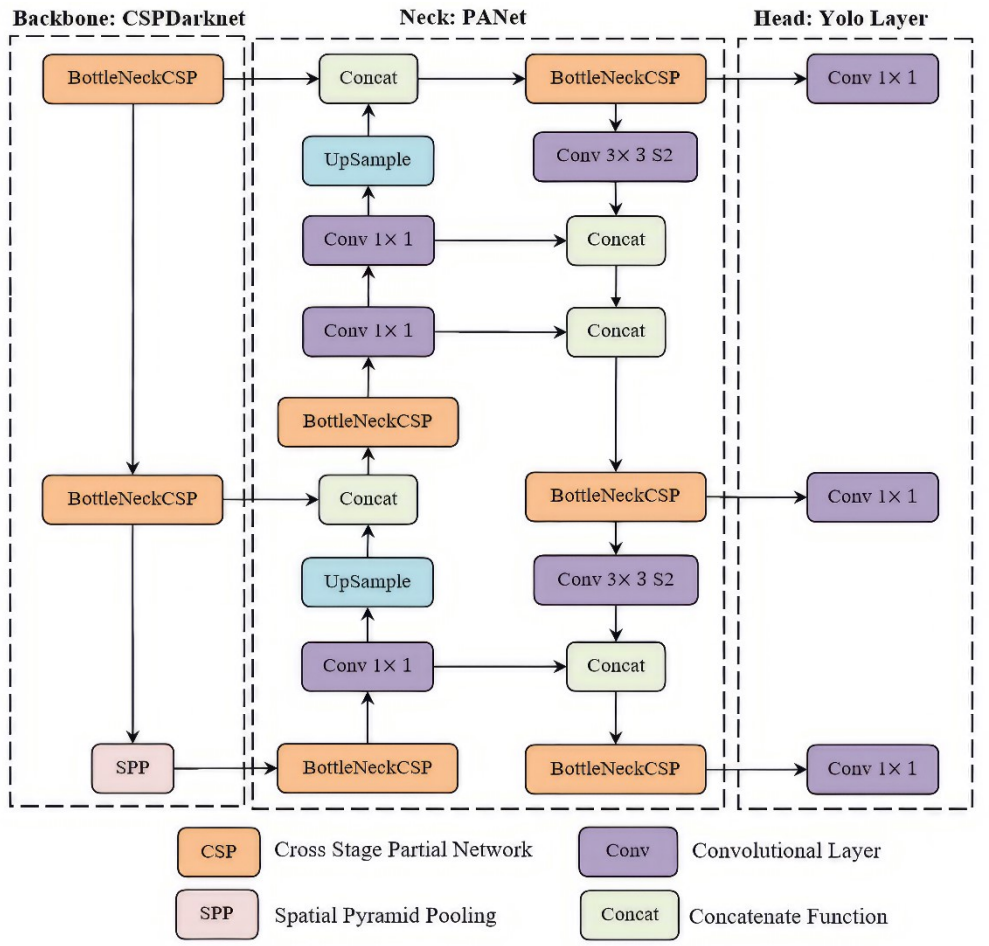

Fig. A1. A schematic wrokflow of *YOLOV5*

A2. Nicolais and Narkis formulation

In the *Nicolais and Narkis* model, the point of yielding is anticipated to transpire within the smallest cross-sectional area fraction of the polymeric matrix (denoted as f_{A_m}), perpendicular to the direction of stress. This is expressed as follows [50]:

$$f_{A_m} = 1 - f_{A_f} \tag{A1}$$

Here, f_{A_f} quantifies the nanoparticle surface area as a proportion of the entire sample area at the cross-section of the cube that is perpendicular to the loading direction. In scenarios where there is no interfacial adhesion, the strength of the nanocomposite is contingent upon the effective area of the load-bearing matrix, which can be expressed as:

$$\sigma_{yc} = \sigma_{ym}\left(1 - f_{A_f}\right) \tag{A2}$$

In Eq. (A2), σ_{yc} represents the yield strength of the composite, while σ_{ym} pertains to the yield strength of the matrix. This equation can be adapted as follows:

$$\sigma_R = 1 - f_{A_f} \tag{A3}$$

In this context, σ_R signifies the relative strength, calculated as $\sigma_R = \sigma_{yc}/\sigma_{ym}$. In the presence of a robust interfacial interaction or adhesion between the polymer matrix and nanoparticles, it gives rise to a transitional region known as the interphase. This interphase possesses distinct physical and mechanical properties compared to both the polymer and the nanoparticles themselves. In this context, stress can efficiently propagate from the matrix to the nanoparticles via the interphase, consequently redistributing a fraction of the initially applied stress that primarily impacted the matrix. To accommodate this interaction effect, the *Nicolais and Narkis* model was adapted, as delineated below [51]:

$$\sigma_{yc} = \sigma_{ym} f_{A_m} + \sigma_{yint} f_{A_{f+int}} \tag{A4}$$

In Eq. (A4), σ_{yint} represents the interphase strength, and $f_{A_{f+int}}$ signifies the combined surface area fractions for both the interphase and nanoparticles. By taking into account Eq. (A1) and replacing f_{A_f} with $f_{A_{f+int}}$, Eq. (A4) can be reformulated as below:

$$\sigma_{yc} = \sigma_{ym} - \sigma_{ym} f_{A_{f+int}} + \sigma_{yint} f_{A_{f+int}} \tag{A5}$$

$f_{A_{f+int}}$ is defined as follows:

$$f_{A_{f+int}} = f_{A_f} + f_{A_{int}} \tag{A6}$$

Hence, the total surface area fraction of the nanoparticle can be expressed as:

$$f_{A_{f+int}} = f_{A_f} + i f_{A_f} = f_{A_f}(1+i) \tag{A7}$$

Here, i represents the interaction factor, which quantifies the ratio of interfacial area to that of the corresponding particle.

A3. Tensile properties of nanocomposites

The stress-strain curves for UHMWPE and its nanocomposites are shown in Fig. A2. Additionally, the tensile data derived from these curves are presented in Table A1.

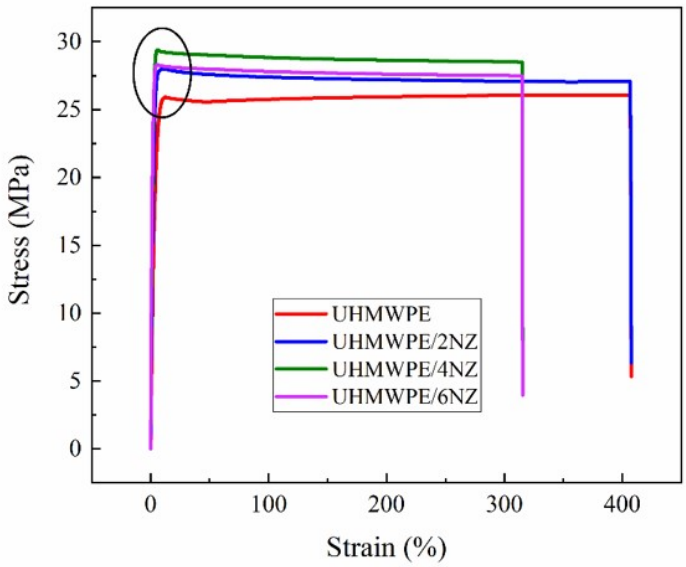

Fig. A2. Stress-strain curves for neat UHMWPE and its nanocomposites

Table A1. Mechanical properties of *UHMWPE* for different compounds

Compounds	Tensile modulus (GPa)	Tensile strength (MPa)	Tensile strain (%)
UHMWPE	0.91 ± 0.082	25.9 ± 0.87	407 ± 25
UHMWPE/2NZ	1.093 ± 0.091	27.8 ± 0.92	407 ± 20
UHMWPE/4NZ	1.213 ± 0.089	29.4 ± 0.63	315 ± 15
UHMWPE/6NZ	1.305 ± 0.096	28.3 ± 1.13	315 ± 10

A4. Dataset Partitioning

After completing the annotation and polygon drawing process described in Section 2.4, the dataset was divided into three distinct sets: 80% for training, 10% for validation, and 10% for testing. This partitioning was designed to ensure robust model training and evaluation. Each set included the annotated polygon coordinates along with the original image data, which were subsequently input into the neural network model for further analysis. For more detailed information on the dataset partitioning procedures, please refer to Section 2.4.

A5. Implementation Details and Hyperparameters

Python 3.8 with PyTorch 1.7.0 was utilized for implementing YOLOv5. The hyperparameters were set to a learning rate of 0.001 and a batch size of 16. The anchor sizes were selected based on the average dimensions of nanoparticles in the annotated datasets, determined through an analysis of the dataset characteristics.

A6. Mathematical description of loss functions

While implementing YOLOv5 for segmentation, three primary loss functions were employed: box loss, objectness (obj) loss, and segmentation (seg) loss. Box loss measures the discrepancy between the predicted and ground truth bounding box parameters (center coordinates, width, and height), expressed as:

$$Box\ Loss\ =\ \lambda_{loc}\sum_{i=1}^{N}((x_i-\bar{x}_i)^2+(y_i-\bar{y}_i)^2)+\lambda_{size}\sum_{i=1}^{N}((\sqrt{w_i}-\sqrt{\bar{w}_i})^2+(\sqrt{h_i}-\sqrt{\bar{h}_i})^2) \tag{A8}$$

where (x_i,y_i) and $(\bar{x}_i,\bar{y}_i)$ are the ground truth and predicted center coordinates, and (w_i,h_i) and $(\bar{w}_i,\bar{h}_i)$ are the ground truth and predicted width and height of the bounding boxes. λ_{loc} and λ_{size} are weighting factors. Objectness (Obj) Loss quantifies the error in predicting the objectness score, indicating the presence of an object within a bounding box, as described below:

$$Obj\ Loss\ =\ -\sum_{i=1}^{N}[y_i\,.\,log(p_i)\ +\ (1-y_i)\,.\ log(1-p_i)] \tag{A9}$$

Here, y_i is the ground truth objectness score (1 if an object is present, 0 otherwise), and p_i is the predicted objectness score. **The Segmentation (Seg) Loss assesses the accuracy of the predicted segmentation masks. This evaluation combines Binary Cross-Entropy (BCE) loss and Dice loss to capture both pixel-wise accuracy and overall shape similarity, resulting in the following formulations**:

$$BCE\ Loss\ =\ -\frac{1}{N}\sum_{i=1}^{N}[t_i\,.\,log(s_i)\ +\ (1-t_i)\,.\ log(1-s_i)] \tag{A10}$$

$$Dice\ Loss\ =\ 1-\frac{2\sum_{i=1}^{N}s_i\,t_i}{\sum_{i=1}^{N}s_i\ +\ \sum_{i=1}^{N}t_i} \tag{A11}$$

$$Seg\ Loss\ =\ \lambda_{BCE}.BCE\ Loss\ +\ \lambda_{Dice}\ .Dice\ Loss \tag{A12}$$

where s_i is the predicted segmentation score for pixel i, t_i is the ground truth segmentation label for pixel i, and λ_{BCE} and λ_{Dice} are the weights for the BCE and Dice losses, respectively. The total loss used for training the model is calculated as a weighted sum of the box loss, objectness loss, and segmentation loss as follows:

$$Total\ Loss\ =\ \lambda_{Box}.Box\ Loss\ +\ \lambda_{Obj}.Obj\ Loss+\ \lambda_{Seg}.Seg\ Loss \tag{A13}$$

where λ_{Box} , $\lambda\lambda_{Obj}$, and λ_{Seg} are hyperparameters used to balance the contributions of each loss component.