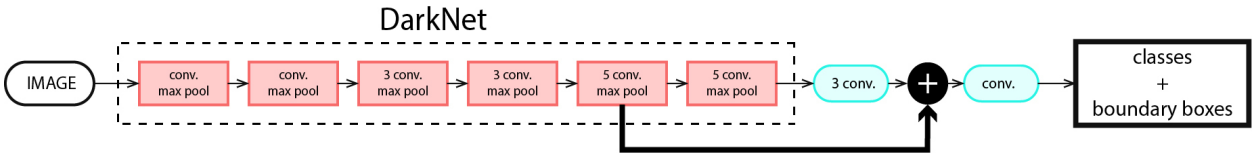
TASK 1

- **Link to the code:**

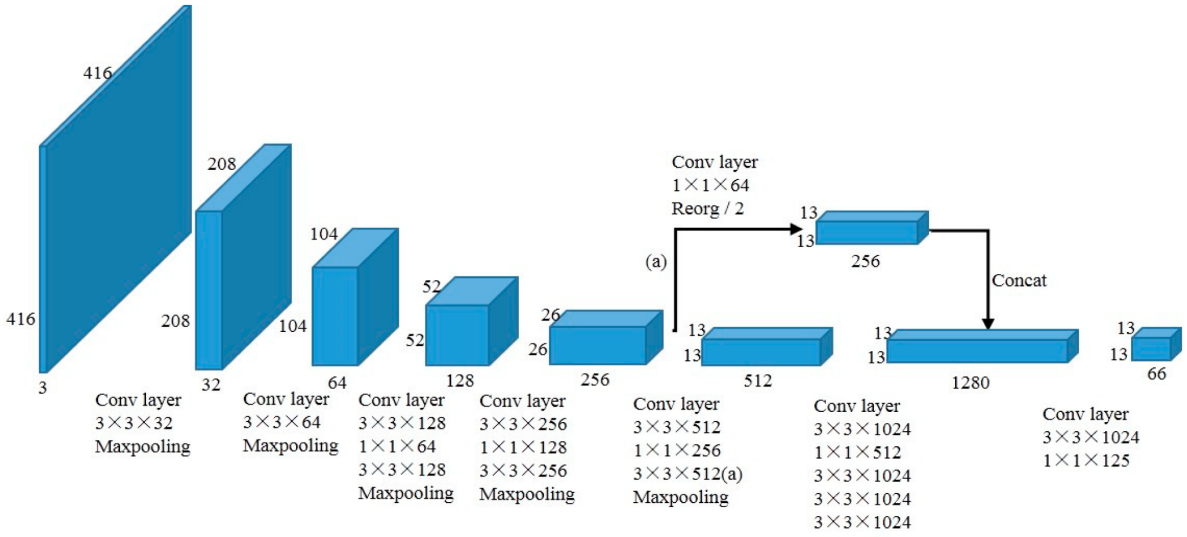
  https://drive.google.com/file/d/1b3l4eak3BzYzdnCwTC-EJ1RFx3-GZ-VX/view?usp=sharing

- **Dataset**:
  - COCO dataset contains 330k images having 80 classes, for example, ‘dog’, ‘car’ etc.
  - It has annotations of each image in a JSON file that consists of the image id, the object classes and the bounding box coordinates, the area of objects, and segmentation masks.
  - There are separate images that belong to train data, validation data, and test data, respectively.

- **Model Architecture:**



  - I chose to implement the YOLO v2 (You Only Look Once) model architecture for object detection in images.
  - The model consists of 19 convolutional layers and 5 max-pooling layers throughout the architecture.
  - The convolutional layers are responsible for feature extraction from the input image.
  - Max-pooling layers are used to downsample the feature maps and reduce the spatial dimensions.



- **Loss function:**
  - The total loss consists of localization, classification, and confidence loss.
    - localization loss (errors between the predicted boundary box and the ground truth)
    - classification loss (errors between predicted class and true class)
    - confidence loss (the objectness of the box)
  - **Localization Loss**
    - It measures the errors in the predicted boundary box locations and sizes.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

where

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

  - **Classification loss**
    - If an object is detected, the classification loss at each cell is the squared error of the class conditional probabilities for each class.

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

where

$\mathbb{1}_i^{obj} = 1$ if an object appears in cell i, otherwise 0.

- $\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i.

- **Confidence loss**
  - If an object is detected in the box, the confidence loss (measuring the objectness of the box) is:

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

where

$\hat{C}_i$ is the box confidence score of the box j in cell i.

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

  - If an object is not detected in the box, the confidence loss is:

$$\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

where

$\mathbb{1}_{ij}^{noobj}$ is the complement of $\mathbb{1}_{ij}^{obj}$.

$\hat{C}_i$ is the box confidence score of the box j in cell i.

λ_{noobj} weights down the loss when detecting background.

- The **total loss** adds localization, confidence, and classification losses together.

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

- **Modifications made to the model architecture to suit the dataset:**
  - The input size of the model is 416 * 416 pixels. Therefore, the images were transformed to the given size.
  - 5 anchor boxes were chosen, and the dimensions were given based on the objects (object sizes and aspect ratios) in the coco dataset.
  - The hyperparameters of the model, like batch size, learning rate, and the number of epochs, were chosen based on the size and complexity of the dataset.
  - Deep neural network, i.e., 24 layers to handle the diverse object categories of the data.
  - The model uses softmax activation function for class prediction, allowing it to predict multiple object classes for each bounding box.
  - The custom loss function is designed as such that takes object detection, including bounding box accuracy, object class prediction, and confidence scores, into account. The loss function is designed to balance these components and consider cases where bounding boxes have no objects.

- **Dataset Preparation:**
  - The dataset used for object detection is the COCO dataset, containing 330k images in the dataset.
  - In the code, all images and their associated annotations in the training dataset are processed.
  - The code iterates through the annotations and extracts relevant information such as category ID and bounding box coordinates.
  - It creates a dictionary to map category IDs to class indices, for assigning class labels to objects.
  - Images are converted to RGB color space (if not already) and resized to the target values - 416 * 416 pixels.
  - Pixel values of the image are normalized to the range [0, 1] by dividing by 255.0.
  - Bounding box dimensions are normalized with respect to the target image size.

- The class index for the object in the image is retrieved from the mapping.
- The preprocessed image and label, including class index and bounding box coordinates, are appended to the respective lists.
- Finally, the lists of preprocessed images and labels are converted to NumPy arrays for efficient handling during model training.

- **Training Details**
  - Number of epochs: 100
  - Optimizer: Adam optimizer with a learning rate of 0.001
  - Early Stopping with a patience of 3 epochs
  - Model Checkpoint that saves the best model weights based on validation loss
  - The training of the YOLO v2 model can take a few days to weeks.

TASK 2

- **Link to the code:** https://drive.google.com/file/d/1qQFkVkFy5Dn71BXNomRj06vyWFEG7WWP/view?usp=sharing
- Generate 5 sentences - product-based description for the given product using Open AI API.