Project CI

- Topic: Yoga poses detection

  Why? The fundamental goal of Yoga pose detection and correction is to provide standard and correct yoga postures using computer vision. If the yoga posture is not done properly, it can result in serious injuries and long-term issues.

- Data collection

  1. Downloaded around 30 images each for 10 different classes.
  2. Annotated using Makesense.ai
     a. label_name: Class Name
     b. bbox_x: Top left X coordinate of bounding box
     c. bbox_y: Top left Y coordinate of bounding box
     d. bbox_width: Bounding Box width
     e. bbox_height: Bounding box height
     f. image_name: Image name
  3. Size of dataset: 75 mb

- Data preprocessing

  1. One hot encoding: To represent categorical variables as a binary vector. It is commonly used to convert categorical data, such as class labels, into a format that can be input into a machine learning model.
  2. 3 separate arrays: Image, bounding box, class
     a. Loop over each annotation
     b. Fill imageInfo array with annotated values
     c. Look for image and compute its dimension
     d. Scale bounding box wrt image
     e. Scale the image

f. Apend into corresponding arrays

3. Normalization:
    a. Convert each data lists to NumPy arrays.
    b. Used to scale the values of dataset between 0 and 1 to ensure that all variables in the dataset have the same scale and to prevent any one variable from having too much influence on the final result.
4. Split data into test and train using train_test_split

▼ Model selection

1. VGG16:
    a. High accuracy
    b. Large number of pre-trained models available so it can be easily fine-tuned for a specific task.
    c. Simple architecture: it has a stack of convolutional and max pooling layers with a small number of parameters, making it easy to understand and implement.
    d. Good performance on small datasets

    VGG16 is a heavy model and requires a lot of computation power and memory. It's not always suitable for real-time or mobile applications.

2. Mobilenetv2
    a. Efficient: in terms of computation and memory usage, making it suitable with limited resources.
    b. High accuracy
    c. Small model size: MobileNetV2 has a small model size, which is important for mobile and embedded devices that have limited storage capacity.
    d. Inverted residual structure: Uses an inverted residual structure(design pattern used in the MobileNetV2 architecture to reduce computation and memory usage without sacrificing accuracy), which helps to reduce computation and memory usage without sacrificing accuracy.

e. Transfer learning: MobileNetV2 can be used as a feature extractor for transfer learning, which can help to improve the performance of other models on small datasets.Transfer learning is a technique where a model that has been trained on one task is used as a starting point for a model on a second, related task

Even though MobileNetV2 is efficient, it may not be the best choice for all tasks, especially when high accuracy is the primary concern, other architectures such as VGG or ResNet should be considered.

3. Resnet50
    a. High accuracy
    b. Good performance on large datasets
    c. Residual connections: ResNet50 uses residual connections, which allow the network to learn residual functions and make it easier to optimize the network. Residual functions are a key component of ResNet. They are used to make it easier for the network to optimize very deep architectures with hundreds or thousands of layers.
    d. Transfer learning: ResNet50 can be used as a feature extractor for transfer learning, which can help to improve the performance of other models on small datasets. Transfer learning is a technique where a model that has been trained on one task is used as a starting point for a model on a second, related task

    ResNet50 is a heavy model and requires a lot of computation power and memory.

▼ MobilenetV2 vs VGG16

1. Efficiency: MobileNetV2 is designed to be efficient in terms of computation and memory usage, while VGG16 is a more computationally expensive model.
2. Model size: MobileNetV2 has a smaller model size(14mb) compared to VGG16(528 mb). This is important for mobile and embedded devices that have limited storage capacity.

3. Accuracy: VGG16 was one of the top-performing models. MobileNetV2 also achieves high accuracy, but it may not be as high as VGG16, especially for high resolution images.
4. Number of parameters: VGG16(138.4M) has much more parameters than MobileNetV2(3.5M), which makes it more prone to overfitting.
5. Real-time applications: MobileNetV2 is designed to be suitable for real-time applications, such as object detection and recognition on mobile devices, while VGG16 may not be suitable for real-time applications.
6. Transfer Learning: Both can be used as a feature extractor for transfer learning, but MobileNetV2 is more efficient and can be used with smaller datasets