Instructions for Formatting JMLR Articles:  
A Microsoft Word Template

Anthony J. Vasquez avasque1@jh.edu

Research Data Scientist

Inficon Inc.

2 Technology Pl,

East Syracuse, NY, 13057

**Editor:** Anthony J. Vasquez

Abstract

This document investigates performance metrics of object detectors that are no longer cutting edge but are still relevant as they are still widely used solutions. The metrics considered for this study include time to train per epoch, precision, recall, and mean-average precision (MAP) scores of the validation set at the end of the tenth epoch. The two architectures considered are the Single-Shot YoloV8 object detector with CSPNet backbone, and the region proposal network FasterRCNN with a Nested Hierarchical Transformer (NesT) vision transformer (ViT) backbone.

# Introduction

Object detection is a widely used machine learning area of interest that has grown exponentially over the last ten years. Due to the increase in compute power, as well as the increase in optimization techniques used, many open-source off-the-shelf solutions have been developed such as the Hugging Face framework for object detection (ViTDet, 2023), and YOLOV8 (Glenn, 2023). The latter has been around since Joseph Redmond first published You Only Look Once: Unified, Real-Time Object Detection (Redmon Joseph, 2016). Since then, YOLO has gone from being an object detection algorithm, to an object detection framework expanded upon by Alexey Bochkovskiy (AlexeyAB) and turned into a framework by Ultralytics.

The transformer architecture was first conceived in the now infamous “Attention is All You Need” (Vaswani, 2017). In this hallmark paper, the authors showed that attention mechanism for a new transformer-based model could be used in lieu of the then cutting-edge process intensive CNN encoder/decoder architectures. Though the architecture was initially used for natural language processing, the architecture found use in other fields, including computer vision, for classification and object detection. One example is the Hugging Face framework that has an extensive and robust library of vision transformer networks that was helpful in understand how to implement a pretrained ViT backbone. For that reason, the NesT ViT pretrained weights were used that were used to for network finetuning.

The question that should be answered by this study is, does the vision transformer inference faster then a similar sized CNN? YoloV8 is well known for inferencing exceptionally fast when compared to other CNN based object detectors, but can it come close to the inference speed of ViT? Furthermore, can ViT perform as good or better on common performance metrics, such as recall, precision, and mAP scores. The final question is, what are the implications on hardware requirements for integration?

# Method

Experiment methods are discussed in this section.

## Data Acquisition

The Numbers MNIST (Modified National Institute of Standards and Technology) dataset (Deng, 2012) was downloaded from the popular data science website Kaggle (KhodaBakhsh, 2018). This dataset was first developed by LeCun et al., in 1994 and is commonly used as a machine learning benchmark dataset (Chavi, 2019). The train set includes 60k samples each with 784 features in a at 8-bit depth in CSV file. The test set includes 10k samples in the same format as the train set. The dataset was selected because of its extensive use over the past 25 years, and because the quality of the data is well established. Furthermore, the size of the dataset was sufficient yet small enough to overlay on larger noisy images, a task that is covered in the data preprocessing section.

Figure . Shown here are samples from the Numbers MNIST dataset (Yashwanth, 2020).

A number and a number

Description automatically generated with medium confidence

## Data Preprocessing

The train digits were opened in CSV format, converted to a Numpy array (Millman, 2020) and each reshaped to 28x28 pixels. The samples were randomly selected from the train set and had no limits to the number of times the samples could be used. A random number between 0-10 was selected, and that number of random images were sampled from the train set. A larger 256x256 pixel image was created and populated with random gaussian noise. The smaller randomly selected images were put at random locations throughout the images; however, no overlap was allowed at the image boundaries to avoid padding complications. However, overlapping was allowed between the number images, this was allowed to preserve the simplicity of the code. However, there is one logic step that ensures that the center of the images do not overlap, though it was observed that most of the large images were absent of overlapping samples.

The locations of each of the samples were recorded along with the class name, which were all written to a text document in the form of the class, upper left x-coordinate pixel, upper left y-coordinate pixel, width, and height of image. For example, each large image with overlaid samples would contain a list of class and coordinates like “0 200 50, 28, 28”. The example shows that a “0” is located at location (x=200, y=50) and is of size (w=28, h=28). Each of the text files could have from 1 to 10 such annotations. YOLOV8 requires that the coordinates be normalized according to the size of the image, which translates to dividing the coordinate values by the size of the large noisy image of 256. Once the samples were overlaid on the larger image, SciPy’s implementation of the single-dimension-Gaussian filter (Virtanen, 2020) was applied to the pixel values to smooth any obvious boundaries around the samples. One text file for each dataset (test and train sets), was written to disk and had the path to each image that could be read by the Custom Pytorch data-loader (Paszke, 2019).

Figure . Shown here is a large noisy 8-bit 256x256 image overlaid with multiple 28x28 MNIST samples.

A number in the sky

Description automatically generated with medium confidence

## Dataset Split

The dataset was pre-split.

## Training YOLOV8 (YV8)

The YV8 network used the yolov8n.pt pretrained weights downloaded directly from Ultralytics (Jocher, 2023). The pretrained weights were trained on the well-known COCO dataset (Microsoft, 2017) composed of 330k images, 1.5 million object instances, and 80 unique object classes.

## Training NesT (ViT)

A small YOLOV8 set of pretrained weights called

## Evaluation

Appendices, if included, follow the acknowledgments. Each appendix should be lettered, e.g., ``Appendix A''. If online appendices are submitted, they should not be included in the final manuscript (see below), although they may be referred to in the manuscript. They will be published online in separate files. The online appendices should be numbered and referred to as Online Appendix 1, Online Appendix 2, etc.

# Results

To ready your work for publication, please typeset it using software such LATEX that produces PostScript or PDF output (LATEX is preferred.) A LATEX style file is available at http://www.jmlr.org/format/jmlr2e.sty or from the editor. There is also a template for Microsoft Word, which can produce PDF files via the Acrobat product. We hope to eventually have macros/samples for other document preparation systems as well. We recommend working from the LATEX source of the sample article (at http://www.jmlr.org/format/sample.tex), which has been annotated to simplify use of the macros in the style file. If you must use a document preparation system other than LATEX, please discuss this with the editor prior to submitting your final document. If you do not have the software necessary to produce acceptable PostScript or PDF files, the editor will recommend a professional service for formatting your article. (Authors will be responsible for paying for this service).

## YOLOV8

The LATEX source for the sample paper, at http://www.jmlr.org/format/sample.tex, details the use of most of the macros in jmlr2e; we describe a few of the macros here for illustration.

## ViT

The recommended citation style file, natbib, is included in jmlr2e.sty. It supports the citation styles described in Section 2.7 with macros such as \citep{} and \citet{}. The basic uses of \citep{} and .

# Conclusion

This section summarizes the experiment and results of this work.

## Conclusion

Dashes should be used--with care--to set off interjections in a sentence. They should be long and there should not be spaces between them and the preceding and following words. Thus, in LaTeX, the input should look like this: