Exploring YOLO Encoder Transfer from Detection to Segmentation

Karim ElSedfy
Department of Computer Science
Bowdoin College
Brunswick, ME, USA
kelsedfy@bowdoin.edu

***Abstract*—Object identification algorithms like YOLO, which are tailored for sparse predictions in the form of bounding boxes, sometimes omit fine-grained spatial information that is essential for dense vision applications [1], [2]. The idea of repurposing a detection-trained YOLO encoder for segmentation without retraining the backbone is investigated in this work. Three approaches to that were investigated in this project. In order to produce pixel-level item likelihood maps at the original image resolution, it first investigates the use of a modified stride-1 inference algorithm. Second, a YOLO model is initially trained for bounding box recognition. The same YOLO encoder is then used as the foundation for an encoder–decoder architecture for semantic segmentation on the CUB dataset. Third, to reduce the loss of spatial detail caused by the encoder's low-resolution bottleneck, skip connections are used between early encoder layers and the decoder. The results show that YOLO encoders may learn transferable semantic representations to handle dense prediction tasks; nevertheless, architectural changes are needed to reclaim precision.**

I. Introduction

Object detection and semantic segmentation are two closely related tasks, typically solved using different types of models. Whereas object detection involves predicting a small set of outputs (e.g., bounding boxes and classes), semantic segmentation involves predicting an output for every pixel [2]. As a result, detection models are forced to downsample spatial information, often in favor of speed. A one-stage detector like YOLO is one example of this; it downsamples the feature maps heavily in order to make fast, global predictions [1].

YOLO has long been regarded as a network designed exclusively for bounding box prediction. Its final feature maps are of extremely low resolution, leading to the assumption that the network is ill-suited for dense prediction [1]. However, YOLO employs a deep convolutional encoder tasked with learning the semantic characteristics of objects and the background. So, a question arises: regardless of YOLO being trained for detection, how much spatial information can be extracted from the encoder, and will those representations be helpful in other dense prediction tasks?

This study examines that question by incrementally stretching a YOLO model beyond its intended task of detecting bounding boxes. It first starts by training a YOLO model on a bus detection dataset. Rather than running the standard inference, inference is performed using a stride of 1. This yields outputs at the original input image resolution where every pixel directly indicates the model's confidence that some object exists. While this isn't a real segmentation mask, it meets the need to visually determine how detection-oriented features react to operating in a dense output environment.

Following on from this idea, the same YOLO encoder is then repurposed in an encoder-decoder configuration for semantic segmentation on the CUB dataset. The encoder is first trained on box prediction using the same YOLO architecture, then frozen, and a second-stage decoder is trained using the segmentation masks. By freezing the encoder, any segmentation improvements can be attributed to the box-prediction features themselves and are not confounded by an additional round of end-to-end training. The prediction is correct globally and the overall silhouette is correctly obtained. However, there are block-like holes and jagged boundaries.

In order to alleviate this problem, skip connections are introduced between the early encoder layers of the YOLO model and the decoder layers, resulting in a U-Net-style architecture. This enables higher-resolution features to contribute directly to the segmentation map and results in a significant improvement in performance. In particular, the shapes of objects and their boundaries are much more well-defined compared to the network that does not include a skip connection. This result differentiates the effect of network architecture on the utility of detection-trained encoders for the task of dense prediction.

In essence, this paper explores the re-tasking of YOLO encoders for both detection and segmentation, through a set of experiments. The main contributions are dense stride-1 inference with a detection model, segmentation training using frozen detection-trained YOLO encoders, and skip connections to recover spatial information. Overall, the presented work suggests that semantic information is shared across tasks for detection-trained encoders, but also shows the constraints imposed by architectures not designed for dense output.

II. Related work

Object detection and semantic segmentation have been considered as distinct tasks which are implemented using different model architectures. One-stage detectors like YOLO have become increasingly popular as they are easier to implement and efficient.

A. Object Detection and YOLO

YOLO treats object detection as a single regression task to spatially separated bounding boxes and class probabilities in a single evaluation of the neural network [1]. This architecture leads to a high inference rate, but it is established on extreme downsampling of feature maps, producing a low spatial resolution in the later layers. Consequently, the YOLO family of detectors is trained for coarse localization instead of final interpretation. Although newer versions of YOLO applied better design choices and multi-scale prediction, the architecture still perceives sparse output, rather than pixels [1], [6].

B. Encoder–Decoder Architectures for Segmentation

Semantic segmentation is usually implemented with encoder-decoder architectures, where the encoder extracts features and the decoder upsamples them to make pixel-wise predictions [2]. U-Net is one such architecture, notable for its skip connections from encoder to decoder of the same depth [3]. These connections let early layers' high-resolution features affect the final prediction, enhancing boundary precision. Encoder-decoder architectures proved useful with encoders pre-trained on similar problems, such as image classification.

C. Transfer Learning and Feature Reuse Across Tasks

Transfer learning is commonplace in computer vision, such as using a pretrained model on a larger dataset for a downstream task [3], [4]. Most transfer learning literature involves transferring the features of image classification to detection or semantic segmentation, for which the spatial resolution is progressively downsampled but remains retained at various levels [5], [7], [8]. In contrast, transferring the features of detection networks for dense prediction tasks has not long been studied, especially when the detection network uses high spatial downsampling during the training process. Previous works propose that detection networks are potent feature extractors since they capture semantically salient features in a compressed manner; however, the transferability of these features to dense prediction tasks is uncertain, and the requirement of end-to-end retraining is largely unknown [3].

In contrast to previous works that concern classification pretraining or full end-to-end transfer, here the transferability of a detection pretrained YOLO encoder in a frozen-backbone manner is studied. Thus, potential of transferring the features of detection networks to dense prediction tasks can be studied in isolation, as well as investigating the architectural elements that play a role.

III. Methodology

A. YOLO-Based Detection Model

The base architecture used in this paper, illustrated in Fig. 1, is a YOLO-style single-stage object detector. YOLO forumlates object detection as a single unified regression problem and outputs bounding box coordinates as well as objectness scores directly from the image in a feedforward manner. This architecture prioritizes inference speed by aggressively downsampling the spatial dimensions of the input at a high factor.

The architecture consists of a deep convolutional encoder that progressively reduces the input resolution, followed by a detection head that operates on the final low-resolution feature map. For the input sizes considered in this paper, the encoder outputs a 7×7 feature map, which acts as the bottleneck from which all other experiments are run.

In the original YOLO formulation, the detection head is implemented using fully connected layers applied to this bottleneck feature map [1]. In this study, the fully connected detection head is replaced with an equivalent fully convolutional head. This preserves the detection-specific layers while removing the architecture-dependent fully connected layer and allowing the network to run on an unconstrained resolution at inference time. Most notably, this allows the use of stride-1 inference at evaluation time, producing outputs that maintain spatial correspondence with the input image.

The modified YOLO architecture is trained on a bus detection dataset with the same object detection supervision as before: bounding box regression and objectness score prediction. An end-to-end architecture that combines both types of detection loss is trained. The final realized detection model is used as the base object detection model for the rest of the experiments. In particular, the convolutional encoder is reused for dense stride-1 inference and as the frozen encoder in encoder–decoder architectures for semantic segmentation. During dense evaluation, downsampling operations are temporarily disabled by using a convolution/pooling stride of 1, resulting in dense per-cell outputs from the detection head. This alters the inference calculation and does not correspond to a sliding-window-based evaluation of the original detector.

B. Dense Stride-1 Inference

Standard YOLO inference only predicts at a low spatial resolution due to downsampling operations being heavily used throughout the network architecture. This is advantageous in detection tasks where speed is a high priority, but it comes at the cost of not being able to provide density over possible spatial locations. To evaluate the spatial nature of features learned from a detection task, a modified procedure for inference is described.

While evaluating the network, all downsampling layers from the convolutional backbone are disabled. Specifically, convolutional layers with stride greater than one are executed with unit stride, and max-pooling layers are replaced with 1×1 pooling operations. In this way, downsampling is avoided while still maintaining learned weights, allowing for dense inference to be run without additional training.

Using this approach, the fully-convolutional detection head results in a high-resolution prediction tensor of size H×W×(C+5B), where H and W are both approximately equal to the input image height and width. This resulting tensor is a spatial prediction of the presence of an object at each location. While only a coarse objectness heatmap is determined, this

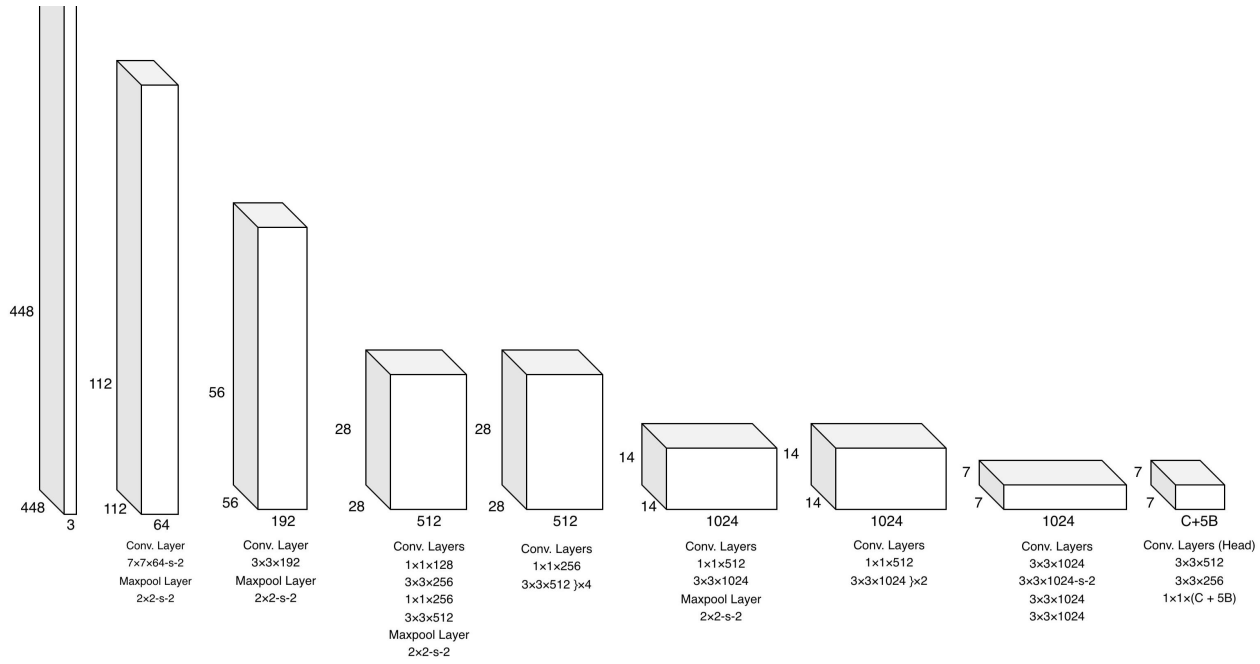

Fig. 1. Modified YOLOv1 architecture used in this study. The backbone follows the original YOLOv1 design, while the detection head is replaced with a fully convolutional head producing per-cell predictions.

allows for dense inference to be viewed without an actual spatial mask.

It is worth noting that this dense inference is not the same as sliding-window inference where the entire detector is run multiple times on subregions of the input image. Instead, it is a mechanism used to explore the spatial information preserved in a YOLO encoder that is trained strictly for a detection task.

C. YOLO-Encoder-Based Segmentation Network (No Skip Connections)

For segmentation, the detection-trained YOLO encoder is used as a fixed feature extractor in an encoder-decoder framework. The encoder feature map is from the adapted YOLOv1 model illustrated in Fig. 1. It maps a 448×448×3 input image to a 7×7×1024 bottleneck feature map. In the original architecture, this bottleneck is directly handled by a task-specific head for detection, but here it is treated as a compact latent representation for dense pixel prediction.

To evaluate the detection-trained representations, the encoder is not fine-tuned for the segmentation task. In particular, the YOLO model is first trained for bounding box regression and objectness prediction, and its backbone is then fixed (i.e., weights are frozen). A pixel-wise segmentation decoder is trained using the extracted bottleneck features and pixel-wise segmentation masks. This setup guarantees that any benefits observed in the semantic segmentation task arise from features learned via detection supervision, rather than from end-to-end fine-tuning.

The segmentation network, YoloSegNet, employs a segmentation decoder comprised of a stack of transposed convolution upsamplers that coherently upsample the 7×7 bottleneck to match the input resolution. Specifically, six 2x2, stride-2 transposed convolution layers are used to upsample by a factor of 64 (i.e. 7→14→28→56→112→224→448). Dimensionality reduction is performed, and BatchNorm + ReLU activations are included after each upsampling step, until the number of channels reaches the number of segmentation classes. Finally, the output is a set of 448×448×C, where C is the number of segmentation classes, allowing optimization based on cross-entropy loss.

D. Adding Skip Connections

In the skip-connected variant, intermediate feature maps outputted from the YOLO encoder are passed to the decoder using lateral skip connections. In particular, feature maps from the YOLO backbone are obtained directly after each of the max-pooling operations, yielding different spatial resolutions. These encoder features are recorded at each time step in the forward pass and concatenated with decoder activation at the corresponding spatial resolution.

The final encoder feature map at 7×7 resolution is used as a bottleneck input to the decoder and is upsampled without a skip connection. In the subsequent decoder layers, we have skip connections at 14×14, 28×28, 56×56, and 112×112 spatial resolutions, respectively. In each of these layers, the upsampled decoder feature map for the layer channel-wise is concatenated with encoder feature maps of the corresponding spatial resolution. The final upsampling to full resolution (448×448) is done without a skip connection.

This design is a mirror image of the decoding layers in U-Net-style architectures and enables the decoder layers to splice semantically dense feature information from the deeper layers

Fig. 2. Qualitative detection results of the YOLO-based bus detector on the test set. The model achieves a mean Average Precision (mAP) of 0.88 at an IoU threshold of 0.5.

of the encoder with pixel-accurate feature representations from the earlier layers. Notably, the YOLO encoder is frozen and does not change during training, allowing the use of YOLO's detection-trained feature maps in this decoder architecture without updating the encoder parameters.

IV. Results

A. Detection Results

The adapted YOLOv1 architecture was first trained for 200 epochs and tested on the bus detection dataset to ensure that the modifications applied do not degrade detection capabilities. Although the original fully connected detection head was replaced by a fully convolutional layer and batch normalization was applied, the architecture performs very well on detection.

Qualitative results on bus detection are presented in Fig. 2. The model detects buses in a variety of challenging scenarios, including different scales, viewpoints, lighting, occlusion, and background. Detection outputs are given with bounding boxes, even in complex urban and traffic scenarios with occlusions and clutters.

From a quantitative perspective, the performance of the detector is 0.88 mean Average Precision (mAP) on the validation set. This indicates that the proposed changes to the architecture did not affect the model's performance and that the features learned are very well suited for the detection task.

B. Stride-1 inference

To investigate the extent of spatial structure preserved within a detection-trained YOLO encoder, an evaluation in a dense stride-1 inference mode is conducted. Here, downsampling in the backbone is turned off, leading to the high-dimensional fully-convolutional-head prediction, H×W×(C+5B). Then, the per-pixel objectness response is overlayed on the original image as a heatmap.

Fig. 3. Dense stride-1 objectness response generated by the detection-trained YOLO model. The response is diffuse with low contrast across the scene, high activity throughout the spatial extent, and perceptible border artifacts, rather than sharply-focused spatial masks for objects.

Figure 3 presents an example of such a dense map. Unlike a segmentation model, an explicit foreground is not observed. Instead, the response is diffuse and low-contrast. The bus segment is only faintly identified, and there exists significant global color casts and edge effects. This indicates that while detection could potentially implicate spatial sensitivity, the intrinsic YOLO structure and detection loss function made for object detection do not lend themselves to dense feature prediction. A main conclusion is that the naive dense stride-1 model evaluation within the realm of a box-detection scenario led to spatial structure that, while informative, could not map to a clean segmentation head.

C. Segmentation Results

This section explores the feasibility of reusing YOLO encoder model pre-trained for detection for semantic segmentation. A YOLO encoder maps a 448×448×3 image to a 7×7×1024 bottleneck. It is first pre-trained for detection on the CUB dataset for 50 epochs and then is frozen, resulting in the segmentation score reflecting segmentation features learned solely from detection supervision.

Two encoder-decoder settings are tested on the CUB dataset. The first one retains the original SegNet decoder with six stride-2 transposed convolution layers to upsample the bottleneck back to the input size. The second one uses the same decoder, but adds U-Net's skip connections concatenating intermediate encoder features with decoder activations at the same spatial resolution.

Segmentation is evaluated at the intersection over union (IoU) of the bird class for all CUB test images (5,794) [9]. The quantitative comparisons are in Table I. The model without

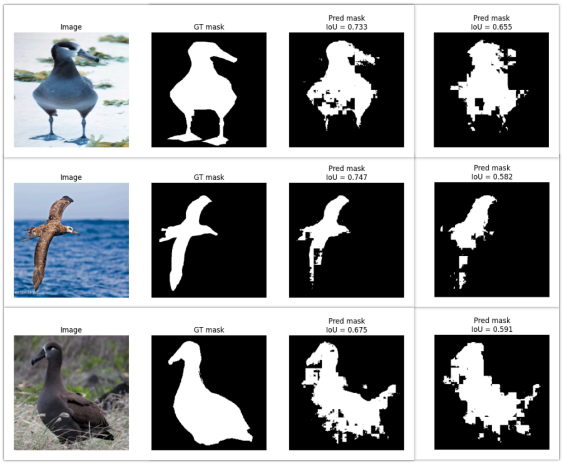

Fig. 4. Qualitative segmentation results on CUB dataset. The images in each row show the input image and the ground truth mask, followed by two masks predicted by networks with different decoder architectures feeding off a frozen YOLO encoder backbone. The networks being compared are skip (left) and non-skip (right) as described in the main text. IoU scores are provided for each of the predicted segmentation masks.

TABLE I
SEGMENTATION PERFORMANCE ON THE CUB TEST SET (5,794 IMAGES) USING A FROZEN DETECTION-TRAINED YOLO ENCODER.

Model	Mean IoU	Median IoU	Max IoU
YoloSegNet (no skip)	0.600	0.636	0.924
YoloUNet (skip)	0.684	0.722	0.958

skip connections scores a mean IoU of 0.600, while the skip connection model scores a mean IoU of 0.684 with an absolute gain of 0.084 IoU. The rise in median IoU aligns with the model's performance in segmenting less typical examples with boundary IoUs.

Qualitative comparisons in figure 4 illustrate how the skip connection model achieves more definite object boundaries with fewer holes in the segmentation. It compensates for spatial detail lost through the downsampling from the detection training on YOLO encoder. These results signify that detection-trained encoders keep some usable spatial details for segmentation; they state the necessity of skip connections in the first place.

V. CONCLUSION

This paper explores the ability to transfer representations from object detection to dense prediction tasks. A single stage detector based on YOLOv1 was modified to use a fully convolutional head to perform dense stride-1 inference and was then used as a fixed encoder for segmentation on the CUB dataset. Using a frozen detection-trained encoder with a separately trained decoder enables analysis of how detection-derived features contribute to pixel-level tasks.

The experiments show that the knowledge learned from detection can transfer to segmentation even in the absence of direct supervision. Furthermore, using skip connections on the decoder side improves segmentation results in terms of both mean IoU and contour smoothness, compared to using a plain upsampling decoder. These results indicate that spatial information learned from detection-trained YOLO encoders can be useful for dense prediction if combined with the proper decoder head.

In summary, this paper shows that single-stage detection networks can be used for more than bounding box detection. It suggests that even when performing dense tasks such as segmentation, the object detection supervision can lead to meaningful representations for dense visual prediction problems.

ACKNOWLEDGMENT

The author would like to thank Professor Jeova Farias for his guidance and mentorship throughout this independent study. His feedback, support, and insight were invaluable in shaping the direction of this work.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, 2015, pp. 3431–3440.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Proc. Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Munich, Germany, 2015, pp. 234–241.

[4] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How Transferable Are Features in Deep Neural Networks?" in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 3320–3328.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[6] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, 2017, pp. 7263–7271.

[7] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, 2015, pp. 1440–1448.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 91–99.

[9] M. Everingham *et al.*, "The PASCAL Visual Object Classes (VOC) Challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.