Knowledge Distillation on YOLO with KAN

Dr. Nabeel Mohammed[1], Sahil Yasar[2], Shahriar Ratul[3], Sarwar Alam[4], and Rafiqul Bari[5]

North South University, Dhaka, Bangladesh

Saturday 7th December, 2024

Abstract

In this paper, we investigate the application of knowledge distillation on the one stage object detection model, You Only Look Once (YOLO) implemented using Kolmogorov-Arnold Networks (KAN). We examined several versions of YOLO, including both the large YOLO architectures (Yolo8x, Yolo5x) and their smaller counterparts (Yolo8n, Yolo5n) with and without KAN-based modifications (Yolo8n-kan and Yolo8n-kan-d); and also with and without knowledge distillation. The models were analyzed across a range of datasets, including COCO, COCO128, Crack-seg, Brain-tumor, and African-wildlife, which enabled us to measure their performance in both general and specialized object detection tasks. Our research concentrated on several essential performance metrics such as mean Average Precision (mAP) at different IoU thresholds, precision, and recall. These results showed how the KAN models far outperformed their non-KAN counterparts and even excelled. These findings highlight the advantages of integrating KAN with knowledge distillation for improving the performance of smaller, more efficient models without sacrificing accuracy, especially in specialized object detection tasks. We have two git hub repository, one for YOLOv5 implemented with Fine-grained feature imitation knowledge distillation, `https://github.com/BunnyWarlock/YOLOv5_with_KAN` and one where we forked the ultralytics repository to allow us to implement KAN on any YOLO version `https://github.com/BunnyWarlock/ultralytics_with_KAN`.

1 Introduction

With the growing demand for machine learning models that can function efficiently in resource- constrained environments, the need for innovative model compression techniques has never been more crucial. And among these machine learning models, object detection is one of the most talked about, used in various appliances like self-driving car, robots, etc. However, the large and powerful models like BERT or YOLO usually demand a lot of processing power, which restricts their use in situations like embedded systems and mobile devices and makes it expensive to put them into production. Which is why model compression techniques are being tested with object detection models very frequently. Although they have shown some promise, traditional model compression techniques like pruning and quantization frequently fall short in their capacity to preserve model performance while drastically cutting down on resource use. That is where knowledge distillation comes in.

Knowledge Distillation (KD) is a machine learning technique that transfers knowledge from a large model to a smaller model. The large model is called the teacher model, and the smaller model is called the student model. The goal is for the student model to mimic the behavior of the teacher model. Deep learning Knowledge Distillation can be categorized into three types

- In response-based knowledge distillation, the student model learns to mimic the predictions of the teacher model by minimizing the difference between predicted outputs.
- In feature-based knowledge distillation, the student model is trained to mimic the internal representations or features learned by the teacher model.

- In relation-based distillation, a student model is trained to learn a relationship between the input examples and the output labels.

In particular, Object Detection has been mainly studied in Response-based and Feature-based methods, but not much research has been done in Relation-based methods.

Kolmogorov-Arnold Networks (KAN), is a novel method based on the Kolmogorov-Arnold representation theorem, offer a viable substitute to the standard Multi Layer Perceptron (MLP) model. The Kolmogorov-Arnold representation theorem asserts that any multivariate continuous function can be represented as the composition of a finite number of continuous functions of a single variable. The Universal Approximation Theorem (UAT), on which MLP are based, deals with approximating functions using neural networks. In contrast, the Kolmogorov-Arnold theorem provides a way to exactly represent continuous functions through sums of univariate functions, which is expected to make them more accurate. KAN models have been shown to outperform MLP models with significantly fewer parameters and is more interpretable than MLP models. All of these makes KAN the perfect candidate to act as a student model for Knowledge Distillation, as we can make the student model smaller than ever.

This paper thus explores the application of Knowledge Distillation to enhance KAN-based models, specifically within the context of object detection. By transferring knowledge from a larger, pretrained model (teacher) to a smaller KAN-based student model, we seek efficient, compact models which are better suited to being deployed on resource-constrained devices. Our approach leverages KAN Convolutional Layers (KANConv) for its potential to model complex relationships with fewer resources compared to traditional CNN. To evaluate the effectiveness of the proposed method, we conducted experiments using well established datasets for object detection, including COCO, which are commonly used for benchmarking computer vision models. In addition, we tested the models on specialized datasets such as Crack-seg, Brain-tumor, and African-wildlife, targeting specific domains in medical imaging and wildlife conservation. We present experiments that compare the performance of models distilled from teacher models using KAN against models such as Yolo5x and Yolo8x on key metrics including mAP, precision and recall across these varied datasets.

2 Related Work

2.1 Kolmogorov-Arnold Networks (KANs)

The Kolmogorov-Arnold representation theorem states that any multivariate continuous function $f(x_1, \ldots, x_n)$ can be expressed as:

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right)$$

Here, $\varphi_{q,p}$ are univariate functions mapping each input variable x_p, such that $\varphi_{q,p} : [0, 1] \to R$, and $\Phi_q : R \to R$ are also univariate functions.

The integration of the Kolmogorov-Arnold representation theorem into neural network design has led to the development of Kolmogorov-Arnold Networks (KANs), which differ from traditional MLPs by replacing linear weight matrices with learnable B-spline functions.

KANs structure each layer as a matrix of these learnable 1D functions:

$$\Phi = \{\varphi_{q,p}\}, \quad p = 1, 2, \ldots, n_{\text{in}}, \quad q = 1, 2, \ldots, n_{\text{out}}$$

Each $\varphi_{q,p}$ is a B-spline, defined as a linear combination of basis splines:

$$\text{spline}(x) = \sum_i c_i B_i(x), \quad \text{where } c_i \text{ are trainable coefficients.}$$

The authors have presented proof showing KANs consistently outperform MLPs, achieving significantly lower test loss across a range of parameter, and at much lower network depth (number of layers). KANs also possess continual learning ability. It is commonly observed that when a neural network is trained on a particular task and then shifted to being trained on task 2, the network will soon forget about how to perform task 1, but this is not a problem in KAN.

KAN layers can also be interpreted as functions, giving rise to many possibilities in interpretability on large machine learning models. KANs perform transformations in a way analogous to traditional MLPs, but with the addition of spline-based functional mappings. The overall network can be represented as:

$$\text{KAN}(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_0)(x)$$

At each layer, the transformation Φ_l acts on the input x_l to produce the next layer's input x_{l+1}, described as:

$$x_{l+1} = \Phi_l(x_l) = \begin{pmatrix} \varphi_{l,1,1}(\cdot) & \cdots & \varphi_{l,1,n_l}(\cdot) \\ \vdots & \ddots & \vdots \\ \varphi_{l,n_{l+1},1}(\cdot) & \cdots & \varphi_{l,n_{l+1},n_l}(\cdot) \end{pmatrix} x_l$$

Each activation function $\varphi_{l,j,i}$ is a spline, providing a rich and adaptable response surface to model complex interactions within the data.

The authors did notice that KANs are usually 10x slower than MLPs when both have the same number of parameters, but the way lower parameters in KAN go head to head with MLP may reduce the impact of this downside.

2.2 Convolutional Kolmogorov-Arnold Networks (KANConv)

Convolutional KANs extend this concept to image data by replacing standard convolutional layers with spline-based activations. These networks incorporate the advantages of KAN while handling grid-structured data efficiently. KANConv have demonstrated that using splines can capture more intricate patterns with fewer parameters compared to traditional CNNs, making them highly efficient for real-time applications. These networks also benefit from the flexibility of spline-based learning, which adapts to specific regions of the input, enhancing their performance on structured data.

In computer vision, convolution is a key operation where a small filter (or kernel) is applied across an input image, computing results at each position. In traditional CNNs, these filters consist of fixed weights. However, in KAN Convolutions, the kernel is more advanced. Each element of the kernel, denoted by ϕ, is a learnable non-linear function represented by B-Splines. Formally, each element is defined as:

$$\phi = w_1 \cdot \text{spline}(x) + w_2 \cdot \text{silu}(x) \tag{1}$$

where w_1 and w_2 are weights, and $\text{spline}(x)$ and $\text{silu}(x)$ are functions applied to the input.

In a KAN Convolution, the kernel slides over the image, applying the corresponding activation function ϕ_{ij} to each pixel a_{kl}, and calculates the output pixel as the sum of $\phi_{ij}(a_{kl})$. Let K be a KAN kernel of size $N \times M$, and let the image be represented as:

$$\text{Image} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mp} \end{bmatrix} \tag{2}$$

Then, the KAN Convolution is defined as:

$$(\text{Image} * K)_{i,j} = \sum_{k=1}^{N} \sum_{l=1}^{M} \phi_{kl}(a_{i+k,j+l}) \tag{3}$$

For example, consider a KAN kernel:

$$\text{KAN Kernel} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} \tag{4}$$

When this kernel is applied to an image, the output is computed as:

$$\begin{bmatrix} \phi_{11}(a_{11}) + \phi_{12}(a_{12}) + \phi_{21}(a_{21}) + \phi_{22}(a_{22}) & \cdots & r_{1(p-1)} \\ \phi_{11}(a_{21}) + \phi_{12}(a_{22}) + \phi_{21}(a_{31}) + \phi_{22}(a_{32}) & \cdots & r_{2(p-1)} \\ \vdots & \ddots & \vdots \\ \phi_{11}(a_{m1}) + \phi_{12}(a_{m2}) + \phi_{21}(a_{(m+1)1}) + \phi_{22}(a_{(m+1)2}) & \cdots & r_{m(p-1)} \end{bmatrix} \tag{5}$$

This structure shows how KAN Convolutions can adaptively model complex patterns in an image, offering a compact and efficient way to process data.

2.3 YOLO: Unified, Real-Time Object Detection

The You Only Look Once (YOLO) framework, proposed by Redmon et al. in "You Only Look Once: Unified, Real-Time Object Detection" [1], revolutionized the field of object detection by framing the problem as a single regression task. Instead of using a region proposal network or sliding windows like earlier methods, YOLO treats object detection as a single, unified model that directly predicts bounding boxes and class probabilities from an image. YOLO's real-time performance, achieved through its fully convolutional network architecture, has made it a popular choice for practical applications where both speed and accuracy are critical. The model's architecture has evolved through multiple versions (YOLOv1, YOLOv2, and YOLOv3), with improvements in accuracy and efficiency at each stage. The architecture can be seen in more detail in Figure 1.

3 Methodology

In this study, we enhance the YOLO object detection architecture by integrating Kolmogorov-Arnold Convolutional (KANConv) layers into it. After this we trained the normal Yolov5 and Yolov8 models along with the KAN versions, on multiple datasets for a set number of epoch, and recorded their performance. Then we ran the fine-grained feature imitation knowledge distillation, with the Yolov5x model acting as the teacher and the Yolov5n and Yolov5n-kan acting as the student, and recorded their performance their as well. Finally we used these records to compare between the normal YOLO models and their KAN counterparts. We forked and kept the original YOLO repository as similar as possible in order to use their builtin training and validation functionality. These helped tremendously in simplifying our code. The only changes we made were to make KAN versions of some blocks and then update the various files of the YOLO library in order for them to recognize these blocks.

3.1 YOLO-KAN Architecture Overview

We made two version of the YOLO-KAN models. The first version replaces one of the C2f block in the backbone of the YOLO architecture (yolov5-kan and yolov8-kan). The second version replaces all the detect blocks in the head of the YOLO architecture (yolov8-kan-d). This approach combines the power of KANConv layers while keeping the training speed relatively fast. The first version is testing how including KAN in the backbone of YOLO affects the features gathered and the performance overall. While the second version is testing how including KAN in the head affects how to model uses the gathered features to make prediction and make bounding boxes.

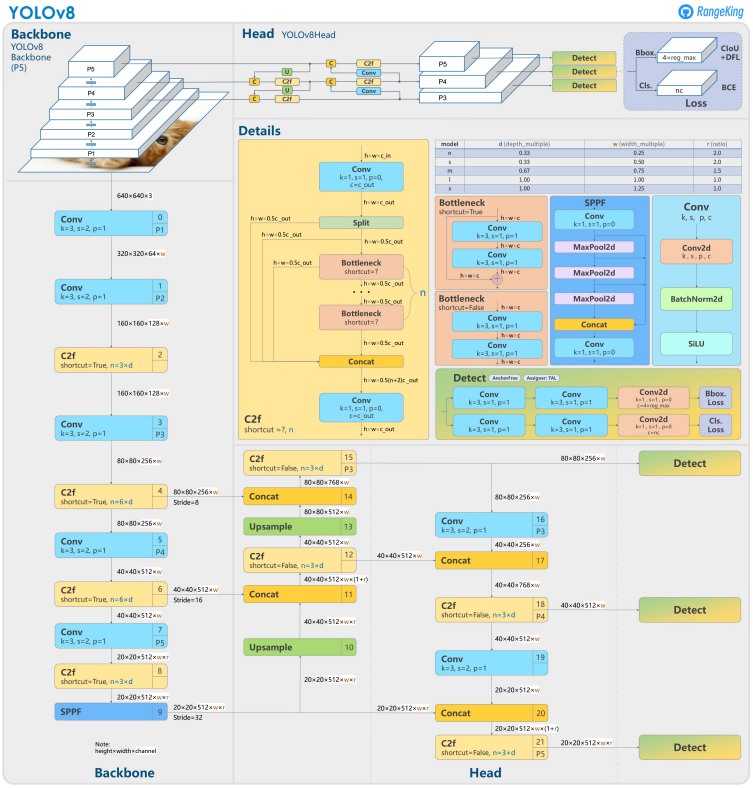

Figure 1: Brief summary of YOLOv8 model structure

3.2 Knowledge Distillation Overview

We copied the code from the fine-grained feature imitation knowledge distillation github repository in order to use it with our KAN models as the student. However, since it is meant to work on anchor based object detection model, this method only works on Yolov5 and below. So we tested this method on our yolov5-kan model acting as the student with the official Yolov5x acting as the teacher.

3.3 Datasets and metrics

We evaluated all of the models on 5 different datasets. First we tested it on the COCO dataset which was by far the largest, so we ran it for 15 epoch on the normal training for the normal and KAN-based models, and 10 epoch for the KD on the normal and KAN-based models. It is a widely-used dataset for object detection, segmentation, and captioning tasks. The second dataset is the COCO128 which is a subset of the COCO dataset with 128 images. We trained on it for 1000 epoch. We mainly did this in order to gauge the models after training for a large number of epoch. The third dataset is another object detection dataset, brain-tumor. We trained on it for 300 epoch. It is a dataset used for detecting brain tumors in MRI scans. The fourth dataset is another object detection dataset, african-wildlife. We trained on it for 300 epoch. It is a dataset aimed at detecting wildlife in outdoor environments. The fifth dataset is an instance segmentation dataset, crack-seg. We trained on it for 100 epoch. It is a dataset focused on crack segmentation.

The models were evaluated using standard object detection metrics such as mAP@.5 (it is the mAP calculated at IOU threshold 0.5.), mAP@.5:.95 (it is the average mAP over different IoU thresholds, from 0.5 to 0.95, step 0.05), Precision and Recall. These metrics were gathered easily, as they are provided automatically by the YOLO library.

4 Results

The results in the table demonstrate the performance of various YOLO architectures across multiple datasets, highlighting both their strengths and trade-offs. It can be seen that the official Yolo8x model dominates over the other on the COCO dataset, but that is due to its huge size and extensive training. It is the same reason why the Yolov5x is also dominating, amongst all the models trained normally without KD. However, things start to change when we see how the Yolov8n trained by us outperforms the official Yolov5n. This can be attributed to the growth of the YOLO model as a whole.

Yolov8 is seen to be performing better than the Yolov8-kan and Yolov8-kan-d models, only on the COCO dataset. However, the KAN models did have greater precision. The Yolov8-kan-d model performed significantly worse on the COCO128 dataset, perhaps indicating how the detect head needs more images to train properly, instead of high epoch. The Yolo8n models also out perform on the brain-tumor dataset slightly, with the KAN models having slightly greater precision. However, the KAN models take the win on the rest of the datasets with Yolov8-kan performing slightly better than the Yolov8-kan-d model.

However, once we implemented KD on both Yolov5n and Yolov5n-kan, the results shifted drastically as the KAN models dominates over the normal Yolov5n model acting as the student, over all datasets. The KD allowed the student Yolov5n model to outperform the official Yolov5n model. And the KAN model one-ups that as they had results almost comparable to Yolov5x.

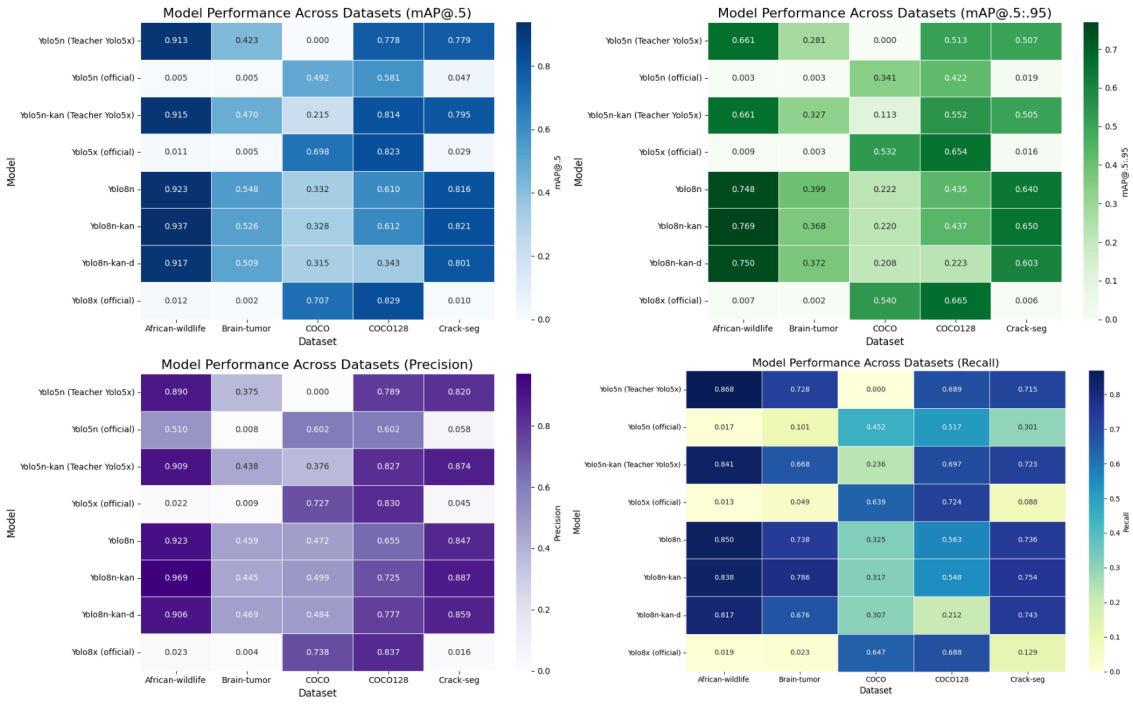

Figure 2: Heatmap of Model Performance Across Datasets.

It is to be noted that the official KAN models were only trained on the COCO dataset, so them acting as teacher on the other dataset as well is debatable.

5 Conclusion

The study presents strong evidence for utilizing knowledge distillation techniques in combination with Kolmogorov-Arnold Networks (KAN) to improve object detection applications. As a case in point, transfer-

ring the knowledge from a larger teacher model such as Yolo5x onto a considerably smaller student model such as Yolo5n-kan, Yolo5n, translated into some very large improvements in the performance metrics such as precision and recall. Most importantly, the models that had been improved with KAN performed best on specific tasks, scoring superior on datasets like Crack-seg and African-wildlife, while doing otherwise efficiently.

Hence, KAN-combining with knowledge distillation works for the ability of smaller models to give good results and save resources for specialized situations. Potentially, these approaches can really benefit the field for situations where computational consumption is valuable or when fine-tuning is needed for particular types of tasks. Future work could expand these results by implementing further architectures, different datasets, and hybrid optimization techniques to generalize the applicability of KAN with knowledge distillation in real-world situations. Knowledge distillation on non-anchor based models like Yolov8 also needs to be tested in the future. Interpreting a model implemented fully by KAN is also very desirable, as it may lead to various insights into object detection as a whole.

6 References

1. Vladimir I. Arnold and Andrey N. Kolmogorov. On functions of three variables. *Doklady Akademii Nauk SSSR*, 114:679–681, 1957.

2. Matthias Fey and Jan Eric Lenssen. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. *arXiv preprint arXiv:2406.13155*, 2017.

3. Ziming Liu. KAN: Kolmogorov–Arnold Networks. *arXiv preprint arXiv:2404.19756*, 2024.

4. Ziming Liu. Convolutional Kolmogorov-Arnold Networks (KAN) for Improved Object Detection. *arXiv preprint arXiv:2406.13156*, 2024.

5. Joseph Redmon, Santosh Divvala, Ross B. Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *arXiv preprint arXiv:1506.02640*, 2015.

6. Mingxing Tan and Ruoming Pang and Quoc V. Le. EfficientDet: Scalable and Efficient Object Detection. *arXiv preprint arXiv:1906.03609*, 2019.

7. Tao Wang, Li Yuan, Xiaopeng Zhang, and Jiashi Feng. Distilling Object Detectors with Fine-grained Feature Imitation. *arXiv preprint arXiv:1906.03609*, 2019.