KM-UNet: Combining UNet and KANs for Image Segmentation

Varun Kumar

IIT Dharwad

March 26, 2025

Outline

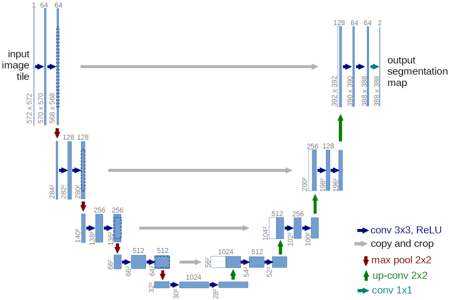
1. UNet: The Foundation
2. Kolmogorov-Arnold Networks (KANs)
3. KM-UNet: The Synthesis
4. Differences, Advantages, and Limitations
5. Potential Applications
6. Future Improvements

UNet: Introduction

- Convolutional neural network architecture for semantic segmentation.
- Classifies each pixel in an image into a specific category.
- Provides a pixel-wise mask, not just object detection.
- Example: Distinguishing healthy tissue, tumor, and background in medical images.

UNet: Architecture

- Characteristic "U" shape with two main parts:
  - **Contracting Path (Encoder):** Downsamples the input, extracting features at different scales.
  - **Expanding Path (Decoder):** Upsamples, recovering spatial information and refining the segmentation mask.
- **Skip Connections:** Crucial element; concatenates encoder feature maps with decoder feature maps at the same resolution. Provides both high-level and low-level information.



UNet: Contracting Path (Encoder)

- Similar to a typical CNN for classification.
- **Convolutional Layers:** Repeated application of convolutional layers (often 3x3 kernels).
- **Activation Functions:** Non-linear activation (traditionally ReLU: $f(x) = \max(0, x)$).
- **Pooling Layers:** Max-pooling for downsampling (usually 2x2). Reduces spatial dimensions, increases receptive field.
- **Feature Map Depth:** Increases as spatial dimensions decrease (more abstract features).

UNet: Expanding Path (Decoder)

- Key to UNet's segmentation capability. Upsamples low-resolution feature maps.
- **Upsampling Layers:**
  - Transposed Convolutions (Deconvolutions): Learnable upsampling (most common).
  - Bilinear/Nearest-Neighbor Interpolation: Simpler, non-learnable methods.
- **Concatenation (Skip Connections):** Feature maps from encoder concatenated with upsampled decoder feature maps.
- **Convolutional Layers (Decoder):** Similar to encoder, refine the segmentation mask.

UNet: Final Layer and Softmax

- **Final Layer:** 1x1 convolution to map feature maps to the number of classes.
- **Softmax Activation:** Produces a probability distribution for each pixel across classes:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where z_i is the output for class i and K is the number of classes.

UNet: Loss Functions

- **Cross-Entropy Loss:** Standard for multi-class classification/segmentation.

$$L = -\sum_{i=1}^{K} y_i \log(p_i)$$

where y_i is the ground truth (1 or 0) and p_i is the predicted probability for class i.

- **Weighted Cross-Entropy:** Used for class imbalance; assigns weights to each class.
- **Dice Loss:** Measures overlap between predicted and ground truth segmentation.

$$\text{Dice} = \frac{2|A \cap B|}{|A| + |B|}$$

Dice loss is typically $1 - \text{Dice}$.

UNet: Training

- **Optimization:** Stochastic Gradient Descent (SGD) or variants (Adam, RMSprop).
- **Backpropagation:** Calculates gradients of the loss function w.r.t. weights.
- **Data Augmentation:** Random rotations, flips, scaling, elastic deformations.

UNet: Advantages and Limitations

Advantages:

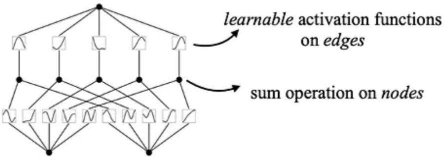
- Excellent segmentation performance.
- Efficient use of data (works well with small datasets).
- End-to-end trainable.
- Handles variable input sizes.

Limitations:

- Computational cost can be high.
- Fixed receptive field.
- ReLU non-linearity has limitations (dying ReLU).
- Fixed Activation Functions

KANs: Introduction

- Significant departure from traditional Multi-Layer Perceptrons (MLPs).
- Based on the Kolmogorov-Arnold Representation Theorem.
- **Key Idea:** Learn the *activation functions themselves*, placed on the *edges* of the network.



Algorithm 1 UNet Training (with Equations)

```
Input: 𝒟 = {(X_i, Y_i)}_{i=1}^N, E, B, α.
Initialize: θ (weights W, biases b).
for epoch = 1 to E do
    for batch = 1 to ⌈N/B⌉ do
        Sample {(X_b, Y_b)}_{b=1}^B.
        Forward Pass:
            Ŷ_b = UNet(X_b; θ)                    ▷ Convolution, ReLU, Pooling, etc.
            ℒ(θ) = LossFunction(Ŷ_b, Y_b)         ▷ e.g., −∑ y_i log(ŷ_i)
        Backward Pass:
            ∇_θ ℒ(θ) = ∂ℒ/∂θ
        Update Parameters:
            θ ← θ − α ∇_θ ℒ(θ)
    end for
end for
Output: Trained θ.
```

- **Input:** $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^{N}$, E, B, α.
- **Initialize:** θ (weights W, biases b).
- **for** *epoch* $= 1$ to E **do**
  - **for** *batch* $= 1$ to $\lceil N/B \rceil$ **do**
    - Sample $\{(X_b, Y_b)\}_{b=1}^{B}$.
    - **Forward Pass:**
      - $\hat{Y}_b = \text{UNet}(X_b; \theta)$ ▷ Convolution, ReLU, Pooling, etc.
      - $\mathcal{L}(\theta) = \text{LossFunction}(\hat{Y}_b, Y_b)$ ▷ e.g., $-\sum y_i \log(\hat{y}_i)$
    - **Backward Pass:**
      - $\nabla_\theta \mathcal{L}(\theta) = \frac{\partial \mathcal{L}}{\partial \theta}$
    - **Update Parameters:**
      - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$
  - **end for**
- **end for**
- **Output:** Trained θ.

Kolmogorov-Arnold Representation Theorem

States that any continuous multivariate function can be represented as a composition of univariate functions and additions.

$$f(x_1, x_2, ..., x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} g_{q,p}(x_p) \right)$$

where $g_{q,p}$ and Φ_q are continuous univariate functions.

KAN Architecture

- **Layers:** Similar to an MLP.
- **Edges (Connections):** Each edge has a *learnable univariate function* (typically B-splines).
- **Nodes:** Nodes simply sum the outputs of the functions on their incoming edges. *No* activation functions on the nodes.
- **No Biases**

B-Splines: Representing Univariate Functions

- Powerful way to represent smooth, continuous functions.
- Defined by:
  - **Knots:** Non-decreasing real numbers defining intervals.
  - **Order (Degree):** Degree of polynomial pieces.
  - **Control Points:** Learnable parameters determining the shape.

B-Spline Basis Functions (Cox-de Boor Recursion)

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } t_i \leq x < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,k}(x) = \frac{x - t_i}{t_{i+k} - t_i} B_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1}(x)$$

Where $B_{i,k}(x)$ is the i-th B-spline basis function of order k, and t_i are the knot values.

B-Spline Function

The actual spline function is a weighted sum of basis functions:

$$S(x) = \sum_{i=0}^{n} c_i B_{i,k}(x)$$

where c_i are the control points (learnable parameters).

B-Splines: Advantages

- **Local Control:** Changing a control point affects the spline locally.
- **Smoothness:** Inherently smooth (differentiable).
- **Flexibility:** Can represent a wide variety of functions.
- **Interpretability**

Training KANs

- **Backpropagation:** Gradients calculated w.r.t. control points of B-splines.
- **Optimization Algorithms:** SGD, Adam, etc.
- **Regularization:** L1 or L2 regularization on control points.
- **Grid Extension and Pruning**
  - **Grid Extension:** Increase B-Spline grid
  - **Pruning:** Remove connections

KANs: Advantages and Limitations

Advantages:

- Higher accuracy than MLPs.
- Interpretability (visualizable functions).
- Better extrapolation in some cases.
- No Nodal Non-Linearity
- Adaptive Complexity

Limitations:

- Computational cost of B-splines.
- Curse of dimensionality (original theorem requires many functions).
- Training stability can be sensitive.
- Limited Scope

Algorithm 2 KAN Training (with Equations)

Input: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, E, B, α, KAN architecture, initial grid.
Initialize: θ (control points c), grids.
for $epoch = 1$ to E **do**
    for $batch = 1$ to $\lceil N/B \rceil$ **do**
        Sample $\{(x_b, y_b)\}_{b=1}^{B}$.
        Forward Pass:
            $\hat{y}_b = \text{KAN}(x_b; \theta)$ ▷ $S(x) = \sum_{i=0}^{n} c_i B_{i,k}(x)$
            $\mathcal{L}(\theta) = \text{LossFunction}(\hat{y}_b, y_b)$
        Backward Pass:
            $\nabla_\theta \mathcal{L}(\theta) = \frac{\partial \mathcal{L}}{\partial \theta}$ ▷ Gradients w.r.t. c
        Update Control Points:
            $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$
        Optional: Grid Extension/Pruning.
    end for
end for
Output: Trained θ (control points c, grid).

KM-UNet: Introduction

- Combines UNet's spatial processing with KANs' non-linear function learning.
- **Key Idea:** Replace convolutional layers in UNet with *KAN-based convolutional layers*.
- Convolutional filters are represented by B-spline functions.

KM-UNet: Architecture

- Retains the overall U-shaped architecture of UNet.
- **KAN-Based Convolutional Layers:** Core difference; "weights" are B-spline functions.
- **No Nodal Activations:** Only B-spline functions on the edges.

KAN Convolutional Layer: Details

1. **Input:** A feature map.
2. **Filters:** Each filter is a set of B-spline functions (one for each element in a traditional filter).
3. **Convolution Operation:**
   - Take patches of the input feature map.
   - Evaluate the corresponding B-spline function for each element in the patch.
   - Sum the transformed values.
   - Slide the patch across the input.
4. **Output:** Feature map

KAN Convolution: Mathematical Formulation

Let:

- X: Input feature map $(H \times W \times C_{in})$
- F: KAN-based filter $(K \times K \times C_{in} \times C_{out})$, each element $F_{i,j,c_{in},c_{out}}$ is a B-spline function.
- Y: Output feature map $(H' \times W' \times C_{out})$

The convolution operation for a single output channel c_{out} at position (x, y) is:

$$Y_{x,y,c_{out}} = \sum_{c_{in}=1}^{C_{in}} \sum_{i=-(K-1)/2}^{(K-1)/2} \sum_{j=-(K-1)/2}^{(K-1)/2} F_{i+(K-1)/2,j+(K-1)/2,c_{in},c_{out}}(X_{x+i,y+j,c_{in}})$$

Training KM-UNet

- **Backpropagation:** Gradients calculated w.r.t. control points of B-splines in KAN layers.
- **Loss Function:** Same as UNet (cross-entropy, Dice loss, etc.).
- **Optimization:** SGD, Adam, etc.
- **Grid Extension and Pruning:**

Comparison: UNet vs. KM-UNet

Feature	UNet	KM-UNet
Convolution	Fixed weights, ReLU	KAN-based filters (B-splines
Non-linearity	ReLU (nodes)	B-spline functions (edges)
Learnable Params	Weights, biases	B-spline control points
Interpretability	Low	Higher (visualizable B-spline
Adaptability	Limited by fixed ReLU	More adaptable (learnable a
Computational Cost	Moderate	Higher

KM-UNet: Advantages

- **Improved Accuracy:** Learnable activations capture complex non-linearities.
- **Enhanced Interpretability:** B-splines can be visualized.
- **Greater Flexibility:** Adapts to a wider range of image characteristics.
- **Potential for Better Generalization:** Adaptive nature of KANs.

KM-UNet: Limitations

- **Increased Computational Cost:** Evaluating B-splines is expensive.
- **Training Complexity:** Tuning hyperparameters, grid extension, pruning.
- **Implementation Complexity:** Requires working with B-splines.

Potential Applications of KM-UNet

- Where high accuracy is crucial (e.g., medical imaging).
- Where interpretability is important (e.g., medical diagnosis).
- Where data is complex and heterogeneous.

Specific Examples:

- Medical image segmentation (tumors, organs, etc.).
- Satellite image analysis.
- Autonomous driving.
- Industrial inspection.

Algorithm 3 KM-UNet Training (with Equations)

Input: $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^{N}$, E, B, α, KM-UNet architecture, initial grids.
Initialize: θ (control points c for all KAN layers), grids.
for *epoch* $= 1$ to E **do**
    for *batch* $= 1$ to $\lceil N/B \rceil$ **do**
        Sample $\{(X_b, Y_b)\}_{b=1}^{B}$.
        Forward Pass:
            $\hat{Y}_b = \text{KM-UNet}(X_b; \theta)$ ▷ KAN convolutions, upsampling, etc.
            $\mathcal{L}(\theta) = \text{LossFunction}(\hat{Y}_b, Y_b)$
        Backward Pass:
            $\nabla_\theta \mathcal{L}(\theta) = \frac{\partial \mathcal{L}}{\partial \theta}$ ▷ Gradients w.r.t. all c
        Update Control Points:
            $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$
        Optional: Grid Extension/Pruning (for each KAN layer).
    end for
end for
Output: Trained θ (control points c, grids for all layers).

Future Improvements: Computational Efficiency

- **Faster B-Spline Evaluation:**
  - Hardware acceleration (GPUs, TPUs, ASICs).
  - Approximation techniques (LUTs, piecewise linear, reduced-order splines).
  - Optimized libraries.
  - Adaptive Grid Refinement
- **Network Pruning and Quantization:**
  - KAN-specific pruning (control point/basis function pruning).
  - Adaptive kernel size.
  - Quantization of control points.

Future Improvements: Interpretability and Explainability

- **Visualization Tools:**
  - Interactive visualization of B-splines.
  - Sensitivity analysis.
  - Feature importance.
- **Regularization for Interpretability:**
  - Smoothness regularization.
  - Sparsity regularization.
- **Relationship to Traditional Features:** Connect B-splines to features like edges and textures.

Future Improvements: Training and Generalization

- **Initialization Strategies:** Better initialization for B-spline control points.
- **Regularization Techniques:** Shape constraints (monotonicity, convexity).
- **Normalization Techniques:** Adapted batch/layer normalization.
- **Adaptive Learning Rates**

Future Improvements: Architectural Innovations

- **Hybrid Architectures:** Combine KAN layers with other layer types (convolutional, transformer).
  - KANs for feature extraction or refinement.
- **Attention Mechanisms with KANs:** KANs to modulate attention weights.
- **Recurrent KANs:** For sequential data (video segmentation).
- **KANs in Different Network Architectures:**
  - DeepLab, Mask R-CNN, Vision Transformers (ViTs).

Future Improvements: Theoretical Advances

- **Approximation Power of KANs:** Further research on the theoretical approximation power.
- **Generalization Bounds:** Develop theoretical bounds on generalization error.
- **Relationship to Other Function Approximation Techniques:** Connections with RBF networks, wavelet networks, Fourier neural operators.

Future Improvements: Beyond Image Segmentation

- **Other Computer Vision Tasks:**
  - Image super-resolution, denoising, inpainting, object detection.
- **Other Domains:**
  - NLP, time series analysis, scientific computing, reinforcement learning.

Future Improvements: Addressing the Curse of Dimensionality

- **Low-Rank KANs**: Low-rank approximations to reduce parameters.
- **Tensor Decomposition Techniques**: CP decomposition, Tucker decomposition for compact representation.

Conclusion

- KM-UNet combines the strengths of UNet and KANs for powerful image segmentation.
- Offers improved accuracy and interpretability compared to standard UNet.
- Future research will focus on computational efficiency, interpretability, and broader applicability.