Computing Midcurve with multi-layer and Convolutional Neural Networks

Prashanth Sreenivasan
AI Engineer and Data Scientist
Pune, India
prashanthsrn11@gmail.com

Yogesh H. Kulkarni
AI Advisor
Pune, India
yogeshkulkarni@yahoo.com

***Abstract*— Midcurve computation: the dimension reduction of 2D thin polygon shapes to 1D curves is important for several Computer-Aided Design (CAD) and Computer-Aided Engineering (CAE) applications. This paper advances the state-of-the art in neural network-based Midcurve computation by building on the previous work in MidcurveNN. This paper introduces two neural network-based architecture for computing Midcurves: a dense neural network architecture and a CNN-based architecture. We explore the effectiveness of modern deep learning techniques. While dense networks show limited improvement, CNN-based models with skip connections reduced average loss by a factor of 10.**

Keywords—Mid-curve computation, dimension reduction, convolutional neural networks, dense neural networks

I. INTRODUCTION

Computer-Aided Design (CAD) and Computer-Aided Engineering (CAE) applications often require dimensionality reduction of complex geometric shapes to make analysis and processing simpler. One important such reduction is the Midcurve - a one dimensional curve representation of 2D thin polygon shapes. A Midcurve captures the essential geometry of the shape while reducing the complexity significantly.

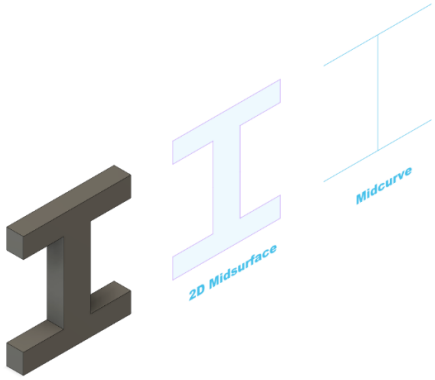

Figure 1: Dimensional reduction from 3D object to 1D midcurve [5]

Midcurve finds application in various areas like Finite Element Analysis(FEA), Character animation, Robot path planning etc.. Chang [3] proposes a solution using Midcurve based on Medial axis transform (MAT) to assist in path planning workspace, effectively dilating the robot's free space and widening the narrow passages.

II. RELATED WORK

Various approaches like Voronoi diagram based, Thinning based, Tracing based, Decomposition based, Pairing based and Mesh based have been followed for the extraction of Midcurve and Midsurface[1]

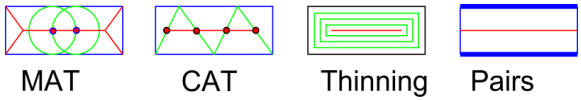

Figure 2 : Comparison of MAT, CAT, Thinning, Pairs [1]

Some of the traditional midcurve computation techniques based on these approaches are Medial Axis Transform (MAT), Chordal Axis Transform (CAT - Mesh based) etc.. For example; MAT traces the locus of the center of a circle as it slides over the object interior[4]. MAT captures sharp corners of the boundary where radius approaches zero[Fig 3]. The resultant curve is called the Medial Axis Curve.

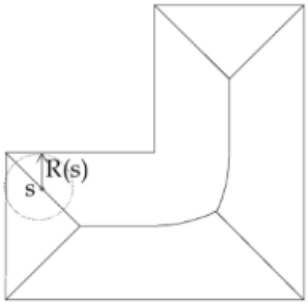

Fig 3: MAT capturing sharp corners [1]

The traditional algorithms are very sensitive to minor irregularities or variations in the shape boundary, thereby producing results which then need significant preprocessing for usage[2]. Also as the shape complexity or resolution increases, calculations like equidistant points or tracking edge movements becomes computationally expensive. Real time applications like interactive simulations and gaming require quick updates. These algorithms find it difficult to get applied in such use cases due to their inherent computational complexity and sensitivity to shape changes.

The MidcurveNN[2] paper introduced several key innovation; by converting the Midcurve computation as an image-to-image transformation problem. The shift to image-based representation was driven by limitations in handling geometric shapes: they can't be treated like simple sequences, some shapes branch in multiple direction, and the inputs and outputs can have different lengths. Graph structures focus more on connectivity than spatial accuracy.

Converting shapes to fix images offers a solution to this problem and works well with neural networks. This approach also allows easy data augmentation, by rotation, mirroring and scaling. Though this approach loses some precision by converting exact geometry to pixels, it offers a practical solution that works well with standard CNN encoder-decoder networks.

MidcurveNN[2] used a simple encoder-decoder architecture[10] for dimensional reduction and used supervised learning with training data pairs. This method though was successful in capturing midcurves of 2D closed shapes, it had certain limitations. It followed a simple architecture design with basic dense layers and had limited capability to capture spatial relations. It also lacked the modern neural network optimizations.

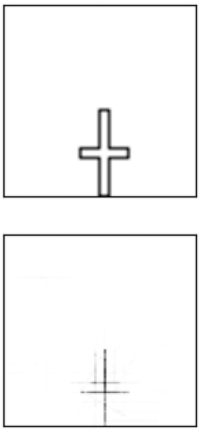

Figure 4: Noisy Midcurve results in simple encoder decoder model [5]

This paper addresses the above limitations of MidcurveNN by introducing two new architectural variants: Dense encoder-decoder architecture and Convolutional Neural Network based architecture.

III. Proposed Architectural Improvements

The original MidcurveNN framework introduced the method of tackling the problem of midcurve computation as a supervised learning problem. This implementation demonstrated feature learning and midcurve generation from input images. The MidCurveNN paper [2] demonstrates that a *Simple Single Layer Encoder and Decoder* network can learn the dimension reduction function well.

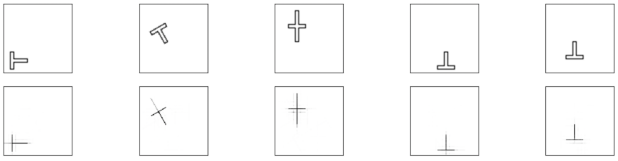

Figure 5: Original simple encoder decoder model performance on test dataset [5]

A. Updated Architectural Approaches

This paper introduces two new frameworks which use the recent advancements in deep learning to better capture the essential features for dimension reduction. This paper advances the current state-of-the-art in neural network-based Midcurve computation through these new frameworks with comparative analysis of performance.

This section goes into the architecture behind the two frameworks, the hypothesis behind why it would perform better than the original model, and the results.

Both these frameworks have a common preprocessing pipeline. This normalizes the images into 128x128 pixels and scales the pixels to a [0,1] range. This ensures the network behaves consistently across varied inputs.

B. Dense Encoder Decoder Architecture

The first proposed architecture uses a fully connected deep structure. The hypothesis was to expand the representational capacity of the original MidcurveNN model using gradual dimension reduction through multiple dense layers to enable better feature extraction.

The encoder pipeline gradually reduces the input dimension from 10,000 to 100, through intermediate representations of 2048 and 1024 neurons. Each layer has ReLU activation functions. The decoder pipeline is symmetric to the encoder structure, and progressively expands the compressed representation to its original dimensions. Finally the last layer employs a sigmoid activation function to produce the output.

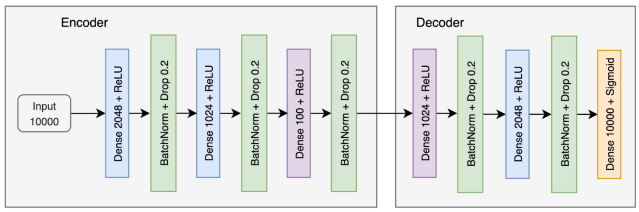

Figure 6: Dense Encoder Decoder Architecture

We incorporated several training strategies to enhance training stability and convergence. The model is trained using an Adam Optimizer with a custom fine-tuned learning rate of 0.0001. The training is monitored through validation performance, with early-stopping to prevent over-fitting[7]. We decided on a batch size of 32 to balance between computational efficiency and stability.

Although the dense encoder decoder uses more layers to gradually reduce the dimensions, the average loss and MAE are worse compared to the original single encoder decoder model. [Fig 14] compares the performance of models keeping batch size and epochs constant. This result motivated us to explore the CNN based architecture approach.

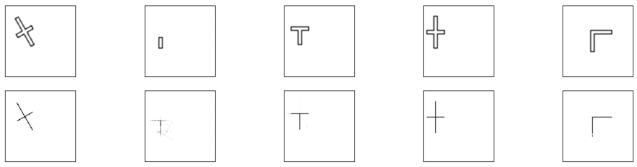

Figure 7: Dense encoder decoder outputs on test dataset [5]

C. CNN-based Architecture

The second proposed architecture uses convolutional network structure which preserves spatial relationships in the inputs. This architecture tackles a fundamental drawback in fully connected networks: the inability to capture spatial hierarchies[11].

The CNN encoder is a feature extraction pipeline having four convolutional block layers. Each layer progressively reduces spatial dimension while increasing depth (32→64→128→256 filters) [Fig 8]. This allows the network to capture both the fine details and the overall structure. Each layer is followed by batch normalization. This improves the stability and normalization of features.

One of the key features in this architecture is the use of skip connections between encoder and decoder layers[8]. These connections preserve information and gradients that might otherwise be lost or diluted by passing through multiple layers. The decoder pathway uses transposed convolutions for upsampling, with skip connections at each level to preserve spatial accuracy.

The training process incorporates dynamic learning rates using a plateau monitoring system. The learning rate is halved when loss stabilizes for five consecutive epochs. This approach, combined with early stopping ensures better results while preventing overfitting.

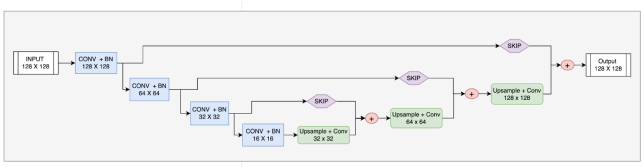

Figure 8: CNN Encoder Decoder Architecture

This model shows the best results amongst the three models. Computing midcurves with almost no noise.

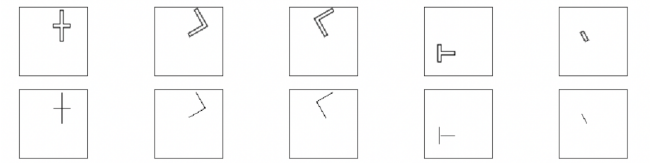

Figure 9: CNN based models output on test dataset

IV. Experimental Setup

This section summarizes the dataset preparation and the evaluation metrics.

A. Dataset Preparation

We have used the same dataset as the MidcurveNN research. The base shapes used for this research are 2D thin polygons, including english alphabets and other geometric shapes. These shapes are augmented using translations, rotations, mirroring and adding noise. This is done to improve the robustness and prevent overfitting. The input, output images are converted to 100x100 grayscale images. Input images contain thin polygon shapes and output images contain their polyline midcurves.

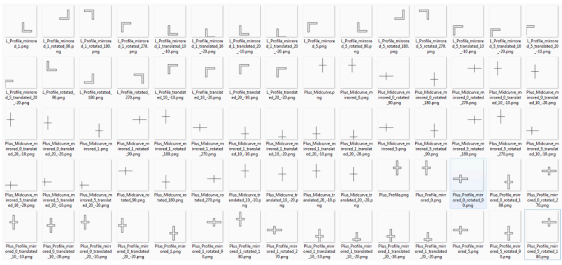

Figure 10: Input output dataset [6]

B. Evaluation Metrics

Binary cross-entropy loss[9], this measures the probabilistic distributions in the predicted and actual pixel distribution in the generated image. It is suitable for image-to-image tasks where output pixels are normalised between 0-1.

Mean absolute error or MAE measures the average absolute pixel difference between the generated and the actual image providing a direct measure of reconstruction.

V. Results

This section shows the performance of the models with respect to the epochs. The comparison of various performance metrics of these models are also discussed here. In these observations the epoch and batch size are maintained the same across training. The total Epochs are 100 and the batch size is 16.

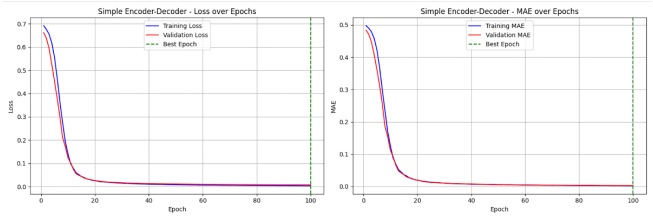

Figure 11 : Loss and MAE over Epoch in Simple model

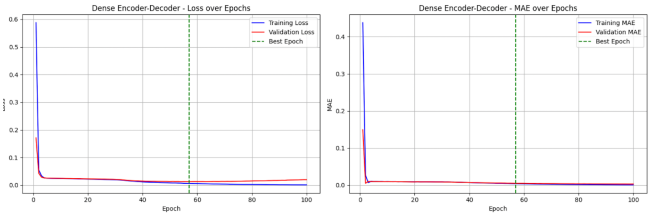

Figure 12: Loss and MAE over Epochs in Dense model

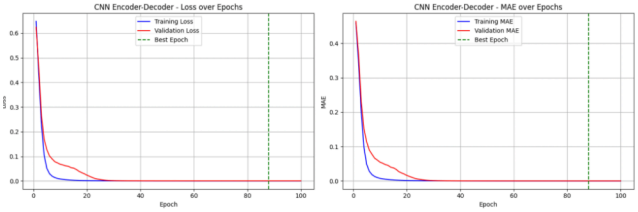

Figure 13: Loss and MAE over Epochs in CNN model

Metric	Simple Encoder-Decoder	Dense Encoder-Decoder	CNN Encoder-Decoder
Best Epoch	100	62	93
Training Loss	0.0034	0.0049	**0.0003**
Training MAE	0.0023	0.0032	**0.0003**
Validation Loss	0.0080	0.0121	**0.0005**
Validation MAE	0.0031	0.0042	**0.0003**
Training Time	48s	59s	2m 42s

Figure 14 : Training metric comparisons across models with constant epoch and batch size

VI. CONCLUSION

This paper demonstrates two deep neural network architectures for tackling the problem of midcurve computation. Building on the MidcurveNN paper [2], the first architecture is one with dense layer encoder decoder. This model fails to surpass the traditional single encoder decoder model, inspiring the exploration of the second architecture. The second architecture is convolutional neural network based architecture. This approach reduces the average loss by a factor of 10, significantly improving the results and removing the noise.

VII. REFERENCES

1. Kulkarni, Y.; Deshpande, S.: Medial object extraction - a state of the art. In International Conference on Advances in Mechanical Engineering, SVNIT, Surat, 2010.
2. Kulkarni, Yogesh. (2022). MidcurveNN: Neural Network for Computing Midcurve of a Thin Polygon. Computer-Aided Design and Applications. 19. 1154-1161. 10.14733/cadaps.2022.1154-1161.
3. Chang, Y.C. & Saha, Mitul & Prinz, F. & Latombe, J.C. & Pinilla, Miguel. (2004). Medial Axis Transform assists path planning in configuration spaces with narrow passages.
4. Kulkarni, Yogesh & Sahasrabudhe, Anil & Kale, Mukund. (2013). Strategies for using feature information in model simplification.
5. Kulkarni, Yogesh. Midcurve NN Github, "https://github.com/yogeshhk/MidcurveNN"
6. Kulkarni, Yogesh. Simple Encoder Decoder for MidcurveNN, Kaggle Dataset. "https://www.kaggle.com/code/yogeshkulkarni/simple-encode-decoder-for-midcurvenn/input"
7. Ying, Xue. (2019). An Overview of Overfitting and its Solutions. Journal of Physics: Conference Series. 1168. 022022. 10.1088/1742-6596/1168/2/022022.
8. H. Bhojwani, V. Bhavsar, R. Gajjar and M. Patel, "Image Resolution Enhancement Using Convolutional Autoencoders with Skip Connections," 2021 2nd International Conference on Range Technology (ICORT), Chandipur, Balasore, India, 2021, pp. 1-5, doi: 10.1109/ICORT52730.2021.9582015.
9. Ruby, Usha & Yendapalli, Vamsidhar. (2020). Binary cross entropy with deep learning technique for Image classification. International Journal of Advanced Trends in Computer Science and Engineering. 9. 10.30534/ijatcse/2020/175942020.
10. Chollet, F.: Building autoencoders in keras. https://blog.keras.io/building-autoencoders-in-keras.html, 2019.
11. Panichev, O.; Voloshyna, A.: U-net based convolutional neural network for skeleton extraction. In Pro-ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2019.
12. Christopher Tierney, T.R., Declan Nolan; Armstrong, C.: Using mesh-geometry relationships to transfer analysis models between cae tools. Proceedings of the 22nd International Meshing Roundtable, 2013.
13. Shen, W.; Zhao, K.; Jiang, Y.; Wang, Y.; Zhang, Z.; Bai, X.: Object skeleton extraction in natural images by fusing scale-associated deep side outputs, 2016.
14. Wang, Y.; Xu, Y.; Tsogkas, S.; Bai, X.; Dickinson, S.; Siddiqi, K.: Deepux for skeletons in the wild, 2018.
15. Zhao, K.; Shen, W.; Gao, S.; Li, D.; Cheng, M.M.: Hi-: Hierarchical feature integration for skeleton detection, 2018.