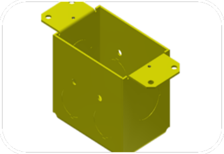
Midcurves & Machines: AI for Smarter Design

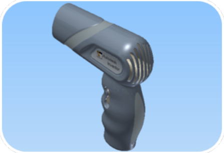
Yogesh Haribhau Kulkarni, Prashanth Sreenivasan

Introduction To Midcurve

Aerospace

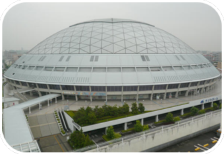


Machinery

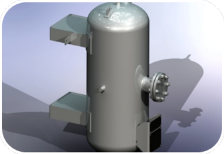


Consumer Products

Energy



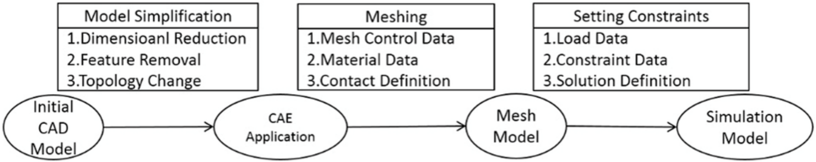
Construction



Industrial equipment

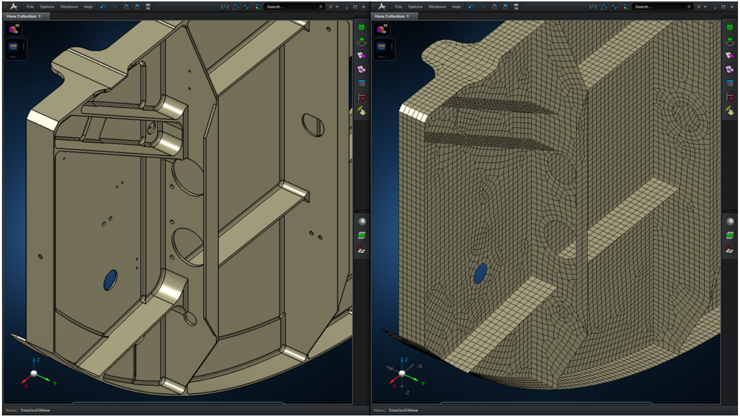
Can we use shapes directly?

- CAD : Designing Shapes
- CAE : Engineering Analysis
- CAD→CAE: Simplification for quicker results.



YHK

CAD-CAE



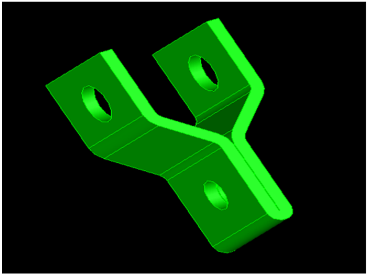
For Shapes like Sheet Metal . . .

	Solid mesh	Shell+Solid mesh	Difference (%)
Element number	344,330	143,063	-58%
Node Number	694,516	75,941	-89%
Total Degrees of freedom	2,083,548	455,646	-78%
Maximum Von. Mises Stress	**418.4 MPa**	**430 MPa**	+3%
Meshing + Solving time	Out of memory	22 mins	N/A (4G RAM)
Meshing + Solving time	**30 mins**	**17 mins**	-43% (12G RAM)

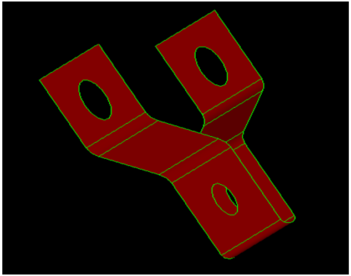
Half the computation time, but similar accuracy

YHK

Midsurface is?



Input: Solid



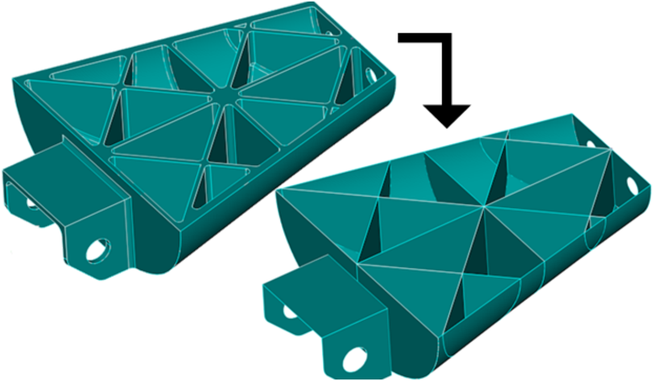
Output: Midsurface

- Widely used for CAE of Thin-Walled parts
- Computation is challenging and still unsolved

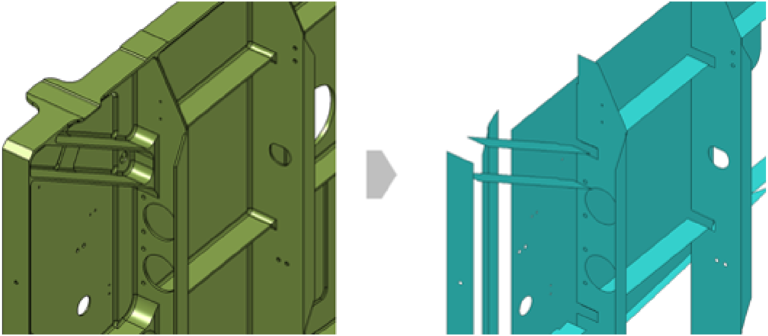
YHK

Getting Midsurface

- Going on for decades . . .
- Manually by offsetting and stitching, initially
- Many CAD-CAE packages give automatic option, but . . .



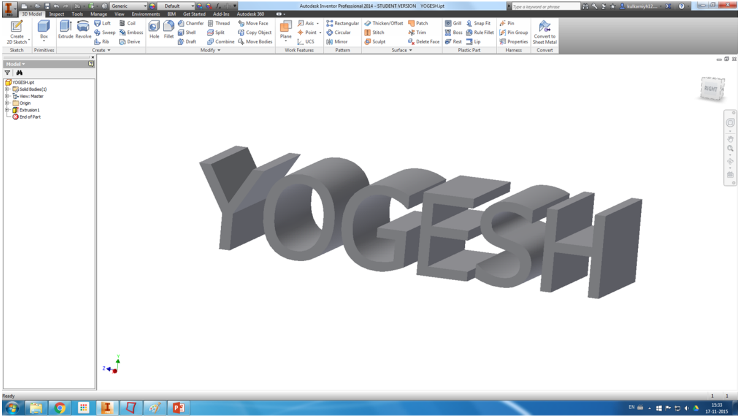
Look at the output



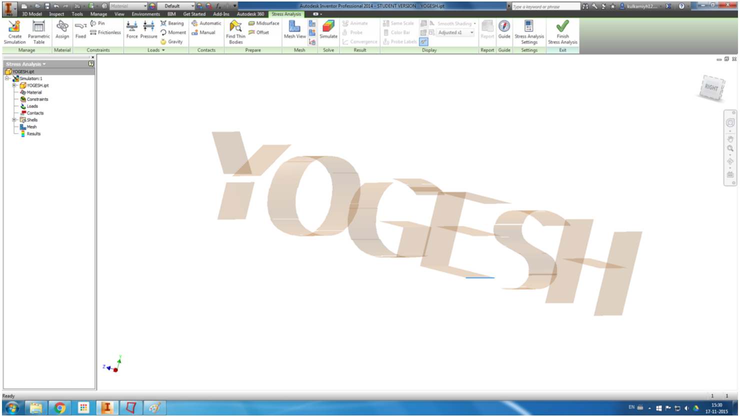
Can't tolerate gaps

- We have thickness sampling,
- To recreate-represent the original shape
- Input and output difference not desirable

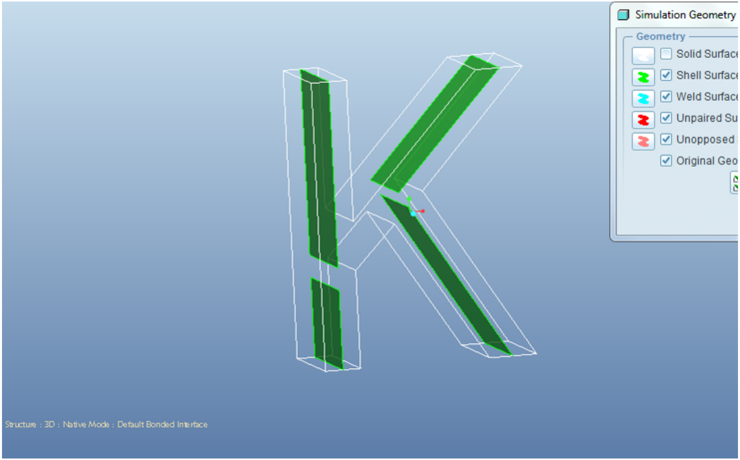
For a simple model like



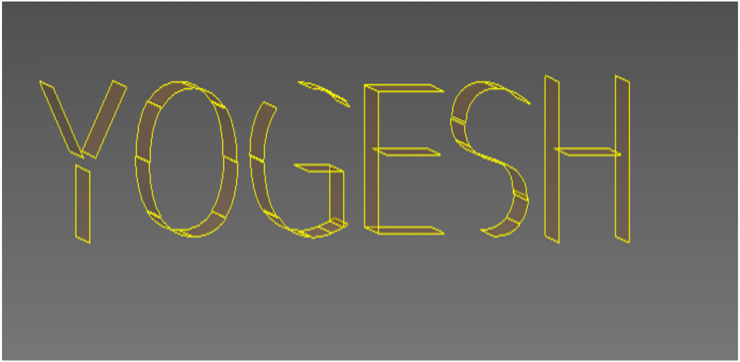
You get



For a far simpler shape



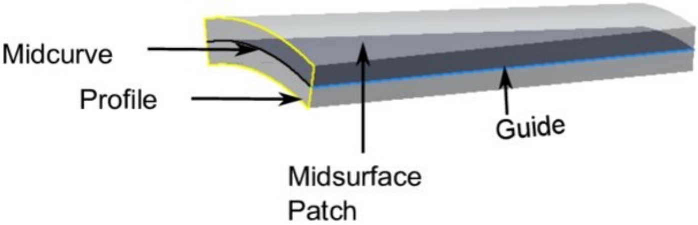
Current Quality



- Errors take weeks to correct for complex parts.
- But still preferred, due to vast savings time
- From Days to hours . . .

Midsurface Computation

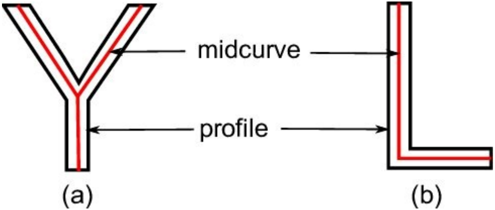
- Midsurface of a Patch is Midcurve of its profile extruded.
- So, it boils down to computing 1D midcurve of a 2D profile



YHK

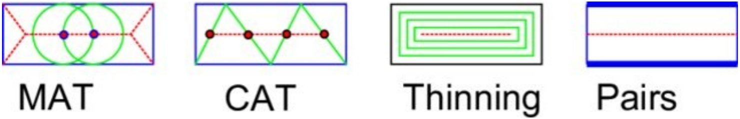
What is a Midcurve?

- Midsurface : From 3D thin Solid to 2D Surface
- Midcurve : From 2D Profile to 1D Curve



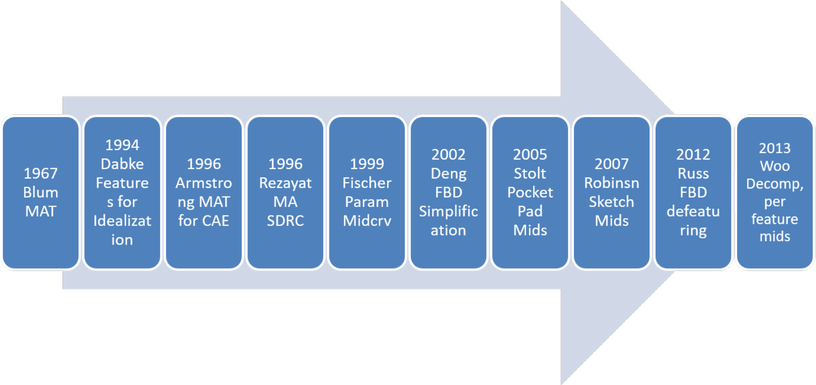
Many Approaches

- More than 6 decades of research...
- Most CAD-CAE packages...
- Rule-based!! Heuristic!! Case-by-case basis!!



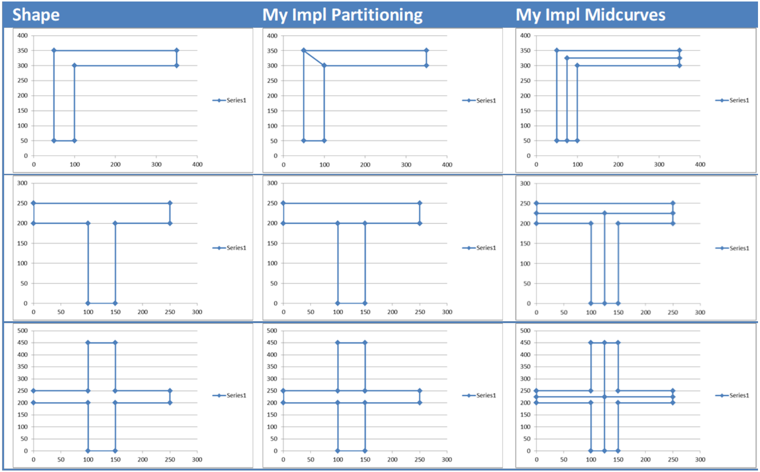
YHK

When-What?



YHK

2017: My PhD Work: Rule-based



YHK

Limitations

- Fully rule-based
- Need to adjust for new shapes
- So, not scalable

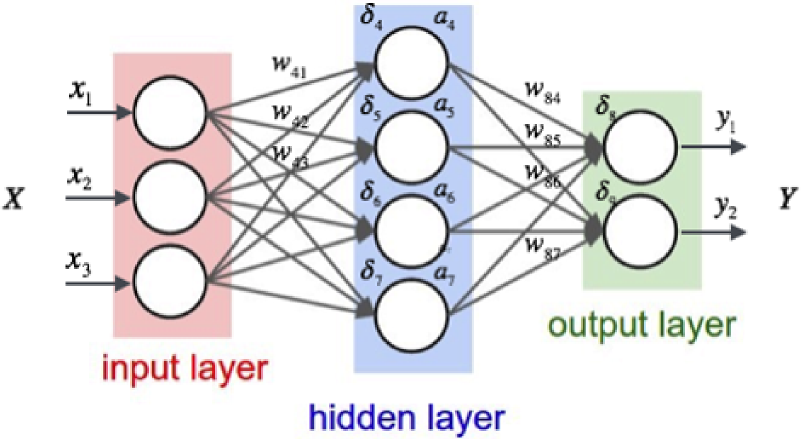
Idea

Can Neural Networks "learn" the dimension reduction transformation?

How?

- Supply lots of training data of profiles and their corresponding midcurves and train.
- Then given an unseen profile, can Neural Network compute a midcurve, mimicking the original profile shape?

Midcurve by Neural network



Midcurve : The Problem

- **Goal**: Given a 2D closed shape (closed polygon) find its midcurve (polyline, closed or open)
- **Input**: set of points or set of connected lines, non-intersecting, simple, convex, closed polygon
- **Output**: another set of points or set of connected lines, open/branched polygons possible

Midcurve : Graph 2 Graph

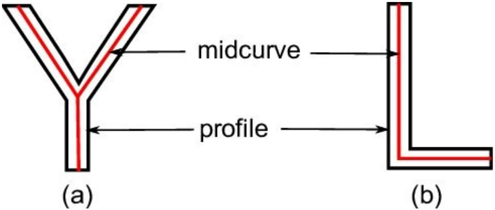
- **Input**: Graph of Input profile with vertices at nodes and lines/curves as edges
- **Output**: another Graph of Output profile with vertices at nodes and lines/curves as edges, open/branched polygons possible
- Both, input and output shapes have different topologies (number of nodes and edges are different) but geometry also, nodes and edges have different positions and shapes. So its network 2 network problem.
- Exiting Graph algorithms like node prediction and link prediction are not useful here as, there, topology of input and output is more or less similar.
- Graph to Graph translation does not seem to evolved enough to do the expected transformation.

Any ideas?

YHK

Midcurve == Dimension Reduction

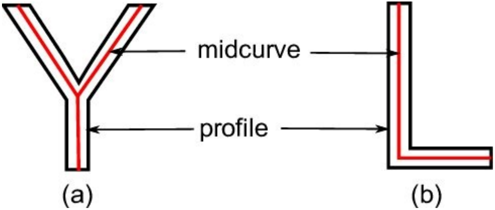
- Like PCA (Principal Component Analysis), wish to find Principal curve
- That 'represents' the original profile shape



YHK

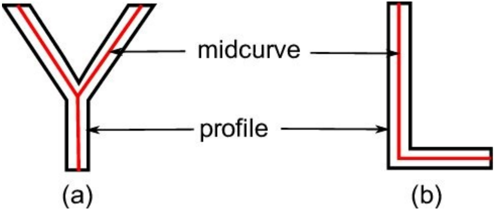
Midcurve == Translation

- Left side (input): 2D Sketch Profile
- Right Side (output): 1D Midcurve
- Sequence 2 Sequence problem



Midcurve ! = Auto-Encoder Decoder

- Its not Auto-Encoder as Input and Output are different
- Its not fixed size i/o as Input and Output sizes are different

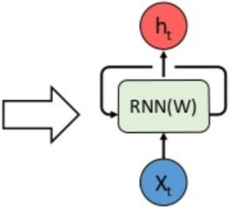


YHK

Variable Size Encoder Decoder

- Batches need fixed lengths
- Made fixed size by Padding.

Friendly	against	Scotland	at	Murray	.
Nadim	Ladki	<PAD>	<PAD>	<PAD>	<PAD>
AL-AIN	United	Arab	Emirates	<PAD>	<PAD>
ROME	1996-12	<PAD>	<PAD>	<PAD>	<PAD>
Two	goals	in	the	last	minutes



Variable Size Encoder Decoder

- OK for NLP, say Machine Translations, where padding values like "-1" can be added along with other words (vectors or indices)
- But in Geometry, its not OK.
- Because any value can represent a Valid Input, even though we don't want it to be the input.

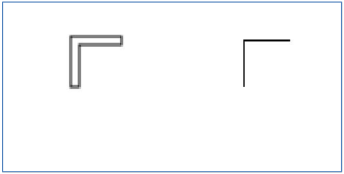
YHK

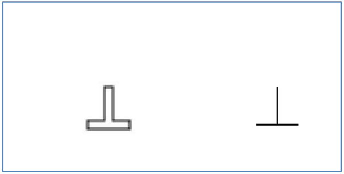
A Twist to the problem

- Till we get good variable size encoder decoder network for geometry. . .
- Decided to convert this Sequence 2 Sequence problem as Image 2 Image problem.

A Twist to the problem

- Input: Black & White Image of 2D profile
- Output: Black & White Image of 1D midcurve





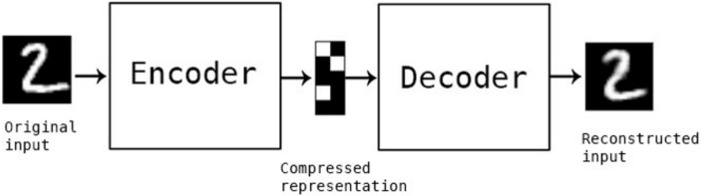
YHK

Solves . . .

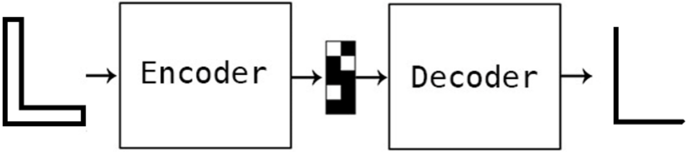
Problems of Geometric sequences

- Variable input/output sizes
- Loops need to be crossed
- Branches

Reuse Image Encoder Decoder



For Dimension Reduction



For Deep Learning

- Need lots of data
- Had just few input output image pairs
- How to augment/populate large variations ...

Phase I

Image to Image Transformation Learning:

- Img2Img: This phase focuses on learning image-to-image transformation using fixed-size 100×100 bitmaps.
- Data Augmentation: The training data will be augmented by scaling, rotating, and translating both input and output shapes within the fixed size.
- Network Architecture: An Encoder-Decoder network, specifically Semantic Segmentation or Pix2Pix, will be employed for image-based dimension reduction.

Data Preparation

Data

Original input and output are in the form of polylines, meaning a list of points, each having x,y coordinates

Profile Data		Profile Picture	Midcurve Data		Midcurve Picture
5.0	5.0		7.5	5.0	
10.0	5.0		7.5	32.5	
10.0	30.0		35.0	32.5	
35.0	30.0		7.5	32.5	
35.0	35.0				
5.0	35.0				

YHK

Data

Profile Data		Profile Picture	Midcurve Data		Midcurve Picture
0	25.0		12.5	0	
25.0	25.0		12.5	22.5	
25.0	20.0		25.0	22.5	
15.0	20.0		0	22.5	
15.0	0				
10.0	0				
10.0	20.0				
0	20.0				

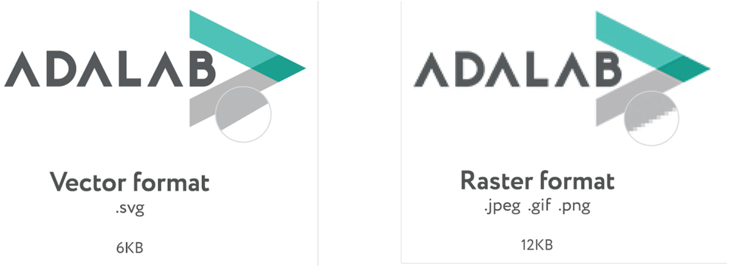
- For each shape, we have this pair of input and output. That's it.
- We need to start with these few samples only

Augmentation

- Such few profile shapes, are just not enough for Neural Networks to train.
- Need more with as much diversity as possible.
- Will need to artificially augment data with transformations, like pan, rotate, mirror, etc.
- All needs to be automatically, programmatically

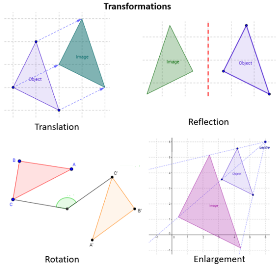
Geometry to Image

- Raw input data is in the Vector format
- Converted it to fixed size (100x100) image by rasterization of drawSVG library.



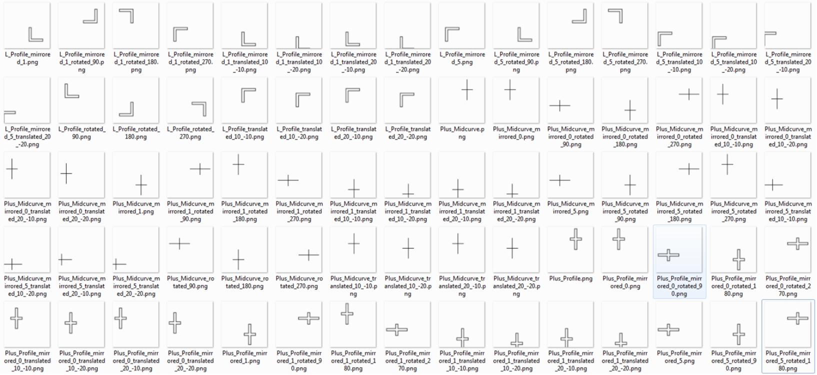
YHK

Variations



- Inputs: I, L, Plus, T
- Operations:
  - Translated
  - Rotated
  - Mirrored
  - Mirrored Translated
  - Mirrored Rotated
- Total: 896 images (still less, but not bad)

Training Data Samples



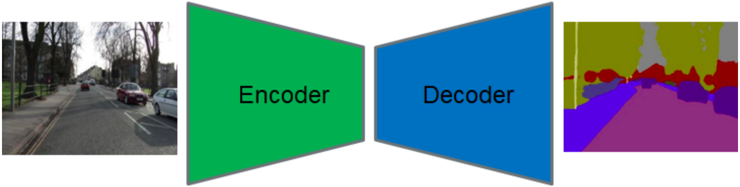
Midcurve By Neural Network

Options For Architectures

- Simple Encoder Decoder (one layer each)
- Dense Encoder Decoder
- Convolutional Encoder Decoder
- Pix2Pix
- ...

Simple Encoder Decoder

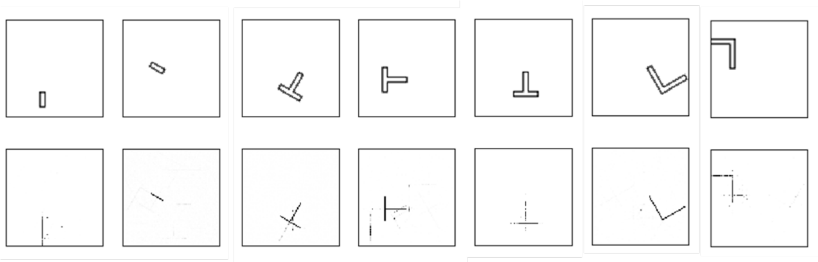
Simple Encoder Decoder



Keras Implementation

```
input_img = Input(shape=(input_dim,))

encoded = Dense(encoding_dim,
      activation='relu',activity_regularizer=regularizers.l1(10e-5))(input_img)
decoded = Dense(input_dim, activation='sigmoid')(encoded)

autoencoder = Model(input_img, decoded)

encoder = Model(input_img, encoded)
encoded_input = Input(shape=(encoding_dim,))
decoder_layer = autoencoder.layers[-1]
decoder = Model(encoded_input, decoder_layer(encoded_input))

autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')
```

Results



Results

- Not very perfect but encouraging
- NN is correct with
  - The location (bounding box)
  - Dimension Reduction is seen
- But, still some stray points and misses

What can be done?

- For the noise, use bounding boxes
- Feedback into error term: differencing with the known output expected
- Classify single pixel image as the skeleton, and rest as noise.

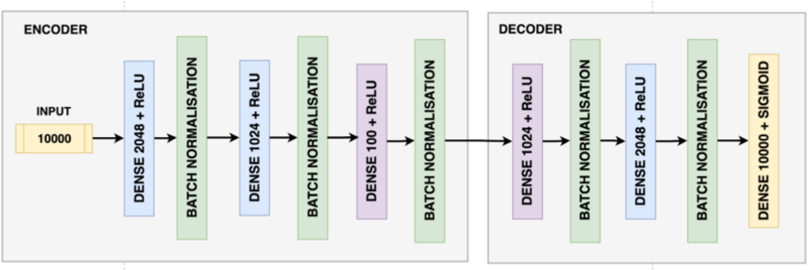
What Next?

- Add denoiser network after the current one
- More Network Architectures
- Sequence-to-Sequence based approaches, taking closed thin polygon as input and polyline as output
- Extending to 3D, ie Midsurface

Micurve using DNN-CNN
by Prashanth Sreenivasan

Dense Network Architecture

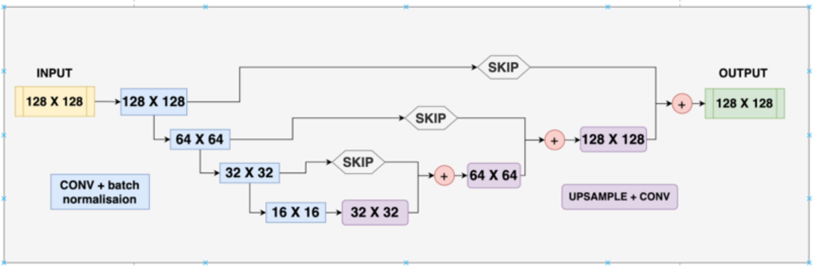
- Gradual dimension reduction
- Multiple dense layers
- ReLU activation
- Symmetric encoder-decoder



YHK

Convolutional Network Architecture

- 4 convolutional blocks
- Skip connections
- Batch normalization
- Dynamic learning rate



YHK

Results

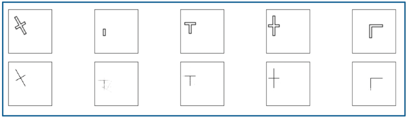
- Performance Metrics
- Comparative Analysis

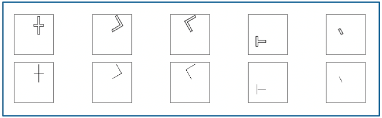
Metric	Simple	Dense	CNN
Best Epoch	100	62	93
Training Loss	0.0034	0.0049	**0.0003**
Training MAE	0.0023	0.0032	**0.0003**
Validation Loss	0.0080	0.0121	**0.0005**

YHK

Visual Results

- Input shapes
- Generated Midcurves
- Quality comparison





Conclusions

- CNN architecture performs best
- 10x reduction in loss
- Improved geometric accuracy

Summary

- Traditional methods of computing midcurves are predominantly rules-based and thus, have limitation of not developing a generic model which will accept any input shape.
- A novel Large Language Model based approach attempts to build such a generic model.
- One such model, GPT-4, seems to be very effective. Although other proprietary and open-source models need to catch-up with GPT-4, even GPT-4 needs to be developed further to understand not just sequential lines but graphs/networks with different shapes, essentially, the geometry.

Summary

- This research significantly advances midcurve computation by exploring the interplay between established methodologies and cutting-edge approaches, particularly integrating Large Language Models (LLMs).
- Emphasizing the nuanced nature of geometric dimension reduction, it identifies challenges in handling variable-length input data, representing intricate shapes, and addressing limitations in existing models.

MidcurveLLM Architecture

- Utilizes an Encoder-Decoder architecture and B-rep structures.
- Showcases promise, despite discrepancies with ground truths.

Implications and Future Directions

- Points to the need for more extensive datasets and refined training parameters.
- Serves as a crucial catalyst for advancing midcurve computation methodologies.
- Invites further scrutiny and advancements in the transformative intersection of geometry and advanced machine learning.

References

- Kulkarni, Y. H.; Deshpande, S. Medial Object Extraction - A State of the Art In International Conference on Advances in Mechanical Engineering, SVNIT, Surat, 2010.
- Kulkarni, Y. H.; Sahasrabudhe, A.D.; Kale, M.S Dimension-reduction technique for polygons In International Journal of Computer Aided Engineering and Technology, Vol. 9, No. 1, 2017.
- Chollet, F. Building Autoencoders in Keras In https://blog.keras.io/building-autoencoders-in-keras.html , 2019.
- Video: https://www.youtube.com/embed/ZY0nuykqgoE?featureōembed
- Presentation: https://drive.google.com/file/d/1Tx5JJK1_LUflMTW-B43HNN2GDMKJMOxR/preview
- Short paper: https://vixra.org/abs/1904.0429
- Github repo, source code: https://github.com/yogeshhk/MidcurveNN

Thanks . . .

- Search **"Yogesh Haribhau Kulkarni"** on Google and follow me on LinkedIn and Medium
- Office Hours: Saturdays, 2 to 5pm (IST); Free-Open to all; email for appointment.
- Email: yogeshkulkarni at yahoo dot com

(https://www.linkedin.com/in/yogeshkulkarni/, QR by Hugging Face QR-code-AI-art-generator, with prompt as "Follow me")