yogeshhk / **MidcurveNN**

Computation of Midcurve of Thin Polygons using Neural Networks

Edit

#neural-networks #encoder-decoder #midcurve #cad #midsurface #deep-learning #geometry #computational-geometry Manage topics

50 commits **1** branch **0** releases **2** contributors

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

yogeshhk Added FAQs in renamed Notes.md Latest commit 626629f 21 days ago

core	Adjusted paths for save model	22 days ago
data	Added new shapes	22 days ago
images	Added FAQs in renamed Notes.md	21 days ago
reference	testing if not normalizing images works or not	last month
utils	Moved from keras to tf.keras, need to test thoroughly though	22 days ago
.gitignore	Adjusted paths for save model	22 days ago
README.md	Adjusted paths for save model	22 days ago
TheoryNotes.md	Added FAQs in renamed Notes.md	21 days ago
config.py	Moved from keras to tf.keras, need to test thoroughly though	22 days ago
environment_keras.yml	Moved from keras to tf.keras, need to test thoroughly though	22 days ago
environment_tf.yml	Moved from keras to tf.keras, need to test thoroughly though	22 days ago
requirements_keras.txt	Moved from keras to tf.keras, need to test thoroughly though	22 days ago
requirements_tf.txt	Moved from keras to tf.keras, need to test thoroughly though	22 days ago

README.md

MidcurveNN

Midcurve by Neural Networks

License:

Description

- 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
- Type:
  - It is a translation problem. In 2D, input is the sketch profile, whereas the output is the midcurve. Input points are ordered (mostly forming closed loop, manifold). Output points may not be ordered, and can have branches (non-manifold)

- It is a variable input and variable output problem.
- It is a network 2 network problem (not Sequence to Sequence) with variable size inputs and outputs
- For Encoder Decoder like network, libraries like Tensorflow need fixed length inputs. If input has variable lengths then padding is done with some unused value. But the padding will be a big issue here as the padding value cannot be like 0,0 as it itself would be a valid point.
- Another problem of using seq2seq is that both input polygons and output branched midcurves are not linearly connected, they may have loops, or branches. Need to think more.
- Instead of going to point list as i/o let's look at well worked format of images. Images are of constant size, say 64x64 pixels. Let's colour profile in the bitmap (b&w only) similarly midcurve in output bitmap. With this as i/o LSTM encoder decoder seq2seq can be applied. Variety in training data can be populated by shifting/rotating/scaling both i/o. Only 2D sketch profile for now. Only linear segments. Only single simple polygon with no holes.
- How to vectorise? Each point as 2 floats/ints. So total input vector is polygon of m points is 2m floats/ints. Closed polygon with repeat the first point as last. Output is vector of 2n points. In case of closed figure, repeat the last point. Prepare training data using data files used in MIDAS. To make 100s, 1000s of input profiles, one can scale both input and output with different factors, and then randomly shuffle the entries. Find max num points of a profile, make that as fixed length for both input and output. Fill with 0,0??? As origin 0,0 could be valid part of profile...any other filler? NULL? Run simple feed forward NN, later RNN, LSTM, Seq2seq
- See https://www.tensorflow.org/tutorials/seq2seq, https://www.youtube.com/watch?v=G5RY_SUJih4, A Neural Representation of Sketch Drawings, https://magenta.tensorflow.org/sketch_rnn https://github.com/tensorflow/magenta/blob/master/magenta/models/sketch_rnn/README.md
- Add plotting capability, show polygons their midcurves etc, easy to debug and test unseen figures.

Plan

- Prep data with transformations
  - Represent 2D profiles in a file, ideal is vector format like SVG. List of points/lines/curves.
  - Once one shape is available, both input as well as output midcurve should go through transformations like translation, rotation, scaling, mirror, etc.
  - This should be done programmatic-ally to generate huge number of input-output pairs.
  - Initial polygon profile data should be from PhD .data fies, having similar scheme for closed and open loops
  - Using these .dat files generate vagarious bitmap 100x100 images programmatic-ally, both for input profile as well as midcurve profile. DrawSVG library can be used to rasterize the vector images
  - Manually: Plot the points of profile and/or midcurve in Excel as scatter line plot. Remove all background grid legends, etc. Copy and paste as image in Powerpoint. Resize it.
  - DrawSVG method is preferred as it puts lots of blank space around the object, which is needed in case of transformations.

Phase I (Image based, Encoder-Decoder fixed size based)

```
- Img2Img: i/o fixed size 100x100 bitmaps
- Populate many by scaling/rotating/translating both io shapes within the fixed size
- Use Encode Decode like Semantic Segmentation or Pix2Pix to learn dimension reduction
```

Phase II (Geometric Points based, Variable size sequences i/o, once such dynamic size encoder-decoder is available)

```
- Ordered sequence of points as io
- RNN LSTM like machine translation or summarization
- Different sizes of input and output
- Closed->closed/open, Manifold->Manifold/Non-manifold
```

Next?

- Full conference/journal paper with results from dense/pix2pix
- Further this research if/once decent collaboration is possible on network to network encoder decoder NNs

Publications/Talks

- Vixra paper MidcurveNN: Encoder-Decoder Neural Network for Computing Midcurve of a Thin Polygon, viXra.org e-Print archive, viXra:1904.0429 http://vixra.org/abs/1904.0429
- ODSC proposal https://confengine.com/odsc-india-2019/proposal/10090/midcurvenn-encoder-decoder-neural-network-for-computing-midcurve-of-a-thin-polygon

Implementation Notes:

- Keras (TBD: Moving from independant Keras to Tensorflow.Keras, so wait for update here)
  - DON'T Conda install -c conda-forge keras DIRECTLY
  - Keras installation cribbed that tensorflow is not present
  - Reinstalled tensorflow by pip – U
  - Got error saying not able to load tensorflow. Looking at https://github.com/ContinuumIO/anaconda-issues/issues/10034 installing conda install vs2013_runtime
  - Go to Anaconda prompt and follow https://medium.com/@margaretmz/anaconda-jupyter-notebook-tensorflow-and-keras-b91f381405f8
  - If you get QT error for matplotlib, use "import matplotlib;matplotlib.use('TKAgg')"
  - Using https://blog.keras.io/building-autoencoders-in-keras.html to have a base code
- Pix2Pix
  - Github Keras-GAN code https://github.com/eriklindernoren/Keras-GAN/tree/master/pix2pix
  - Same Keras-GAN code in notebook format https://www.kaggle.com/vikramtiwari/pix-2-pix-model-using-tensorflow-and-keras/notebook
  - Google Colab based code (different from Keras-GAN) https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/r2/tutorials/generative/pix2pix.ipynb

Why Pix2Pix may not be good

- In midcurveNN problem, both input and output are black and white images with with pixel width.
- For given input image, exact and only one possible output is there. No other output would be correct.
- So, Loss function of such learning systems would be exact and can be specified by simple distance formulas like L1, L2, etc.
- Simple/Dense/CNN based Encoder Decoder have exact loss function.
- In case of GANs, like conditional GAN or Pix2Pix, the purpose of Discriminator is actually to learn the Cost Function itself. (Ref: Phillip pod cast video regarding Pix2Pix). There "goodness" of the output is learnt in the Cost Function, and not the Exactness as specified in the examples above.
- Thus, theoretically, GANs are not suitable for MidcurveNN problem. Comments?

ToDOs

- Get Denoiser working in .py and .ipynb
- Get cnn working in .py and .ipynb
- Get pix2pix working in .py and .ipynb
- Generate more data from more profiles

Errors and Solutions

- no library called "libcairo-2" was found https://stackoverflow.com/questions/28211418/python-oserror-cannot-load-library-libcairo-so-2 Install GTK 3 runtime https://github.com/tschoonj/GTK-for-Windows-Runtime-Environment-Installer and add to path
- Failed to create a directory: models/autoencoder_model.pkl\variables; No such file or directory use a forward slash with the checkpoint directory or os.paths.join("models","modelname")
- If using Keras pass *_constraint arguments to layers.

References

- Image-to-Image Translation with Conditional Adversarial Nets https://phillipi.github.io/pix2pix/

Disclaimer:

Author (yogeshkulkarni@yahoo.com) gives no guarantee of the results of the program. It is just a fun script. Lot of improvements are still to be made. So, don’t depend on it at all.