Code (GitHub): [link](https://github.com/szczekulskij/tomography-reconstruction-CNN)

Proof of Work notebook: [link](https://github.com/szczekulskij/tomography-reconstruction-CNN/blob/master/src/pow_polygons.ipynb)

**Dataset creation**

Randomly rotated and sized polygons (n sides, where n is between 4 and 6). Centralised in the middle of the picture.

Resolution: 128x128 pixels. Binary values.

Graphical user interface, application

Description automatically generated

Angles:

* By default, one hot encoded (array of 180 zeros, with few ones signifying the correct angles)
* Transformed - Assign higher values to angles close to the correct angle and smaller values to angles further away. Values follow the sigmoid function (since the output of our CNN is sigmoid as well + it's a smooth curve)

**Model**

Text

Description automatically generated

**Output:**

(after applying k-means like function)

Text

Description automatically generated

I haven't implemented SFS or any other reconstruction algorithm to showcase how well the reconstruction would be. However, given inputs are sinograms of polygons and the predicted angles are spot on, reconstruction should be flawless as well.

**Disclaimers:**

**Main Question - Why is it working now, but didn't before?**

1. Image resolution - Previous image resolution was too small (64 x 64), there was a loss of data - sinograms weren't `crisp` enough
2. Model architecture - Updated the model based on experience & added dropout layers.
3. Sinogram - We were using a "wrong" kind of sinogram - the default scikit learn sinogram functions outputs sinogram with small values (2e-10), which obviously were too small as an input to AI
4. Angles - We're now using transformed angles. There is a smooth sigmoid-like curve from the worst to best angles - which gives AI so much-needed directions.

Generally speaking, the more I worked with AI the more I learnt about the importance of the quality of input data. Previously both AI inputs (sinogram & angles) were very poor, therefore resulting in poor AI. Once fixed by steps 1,2 & 4, AI had no trouble whatsoever. Training took 4 hours - but AI could converge further given more time (I stopped it early)

**My Feedback**

The coding, re-reading papers & training took me five days. Honestly speaking, the previously written code was poor in every aspect, but that's to be expected from a person with no industrial coding & data experience, which I had none of 2 years ago.

**Further work:**

1. Check if the current architecture generalizes to all polygons (it works for 4,5,6).  
   ETA 2 days work + 2 days training  
   Requirements:
   1. Change in dataset generation
   2. Checks on quality of sinogram data (if polygon with n=100, is sinogram good enough? When do we need to increase the resolution yet again?)
   3. Checks on quality of angles data (if polygon with n=100, then 100 out of 180 angles are correct - in that case, the sigmoid function isn't probably the best way to transform angles data)
2. Add noise  
   ETA 1 day
   1. Changes in data generation to include noise.
   2. Either re-train AI, or train on top of current pre-trained AI (transform learning)