PREDICTING THE VOLUME OF ICE ON ANTARCTICA

Exam Project - Applied Machine Learning - 11/06-2025

Jeppe Brøndum Bach, Anna Sophia Maxen, Maria Friis Greibe

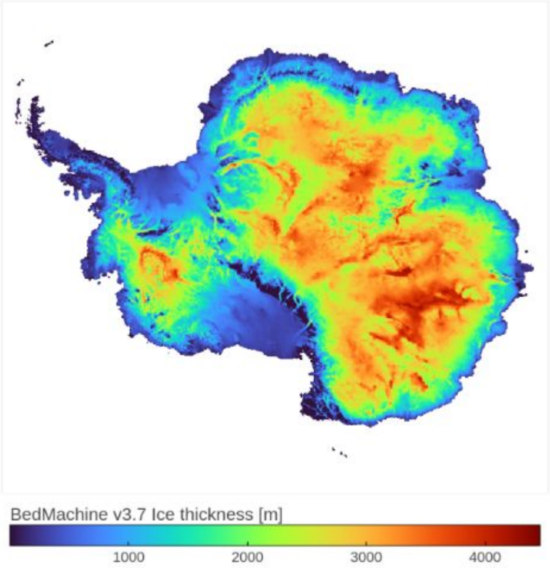
All participants contributed evenly

INTRODUCTION

Goal: Create a model that predicts ice thickness on Antarctica

Climate modelling

Benchmark: BedMachine v3.7



CONTENT

1. Data presentation
2. Preprocessing
3. Models
   a. LightGBM
   b. CNN (and CNN+LightGBM)
   c. CNN + NN
4. Results
5. Outlook

DATA

We would like to sincerely thank Niccolo for the neat data as well as the guidance!

Thanks!

Tabular

Maps

DATA

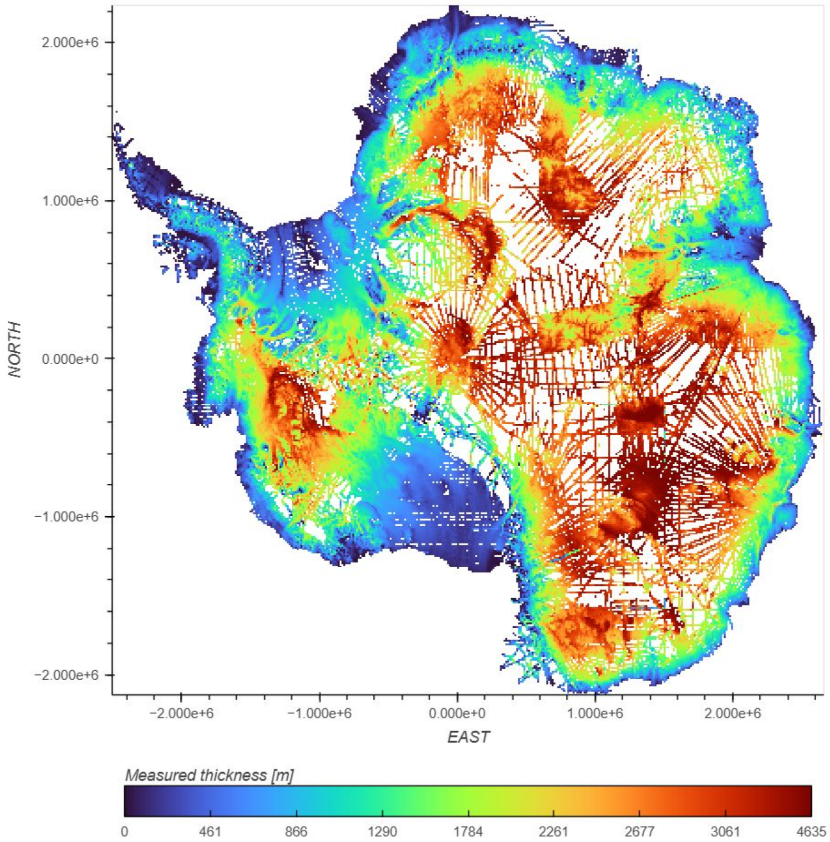
Shape: (79890423, 14)

80 million ice thickness measurements

EPSG:3031 projection

Label:
'THICK'

13 features:
'LON', 'LAT', 'geometry', '**EAST**', '**NORTH**', '**vx**', '**vy**', '**v**', 'ith_bm', '**smb**', '**z**', '**s**', '**temp**'



Thanks!

Tabular

Maps

DATA

	map1	map2	map3	map4
Variables	vx vy	smb	temp	elevation BedMachine
Resolution	450 m	2000 m	2605 m	500 m
Size	6.49 GB	24.5 MB	49.5 MB	808.8 MB

PREPROCESSING

Why:

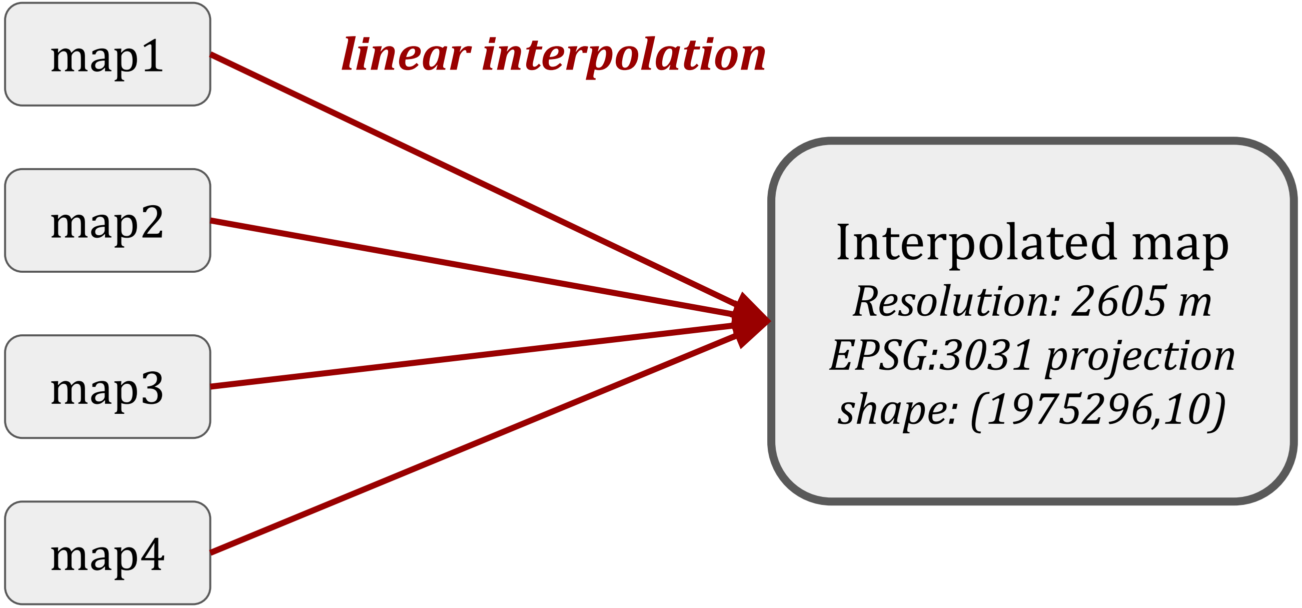
- Correlation
- Data size
- Data processing for CNNs and inference

Interpolated map

Creating images

Data for CNN +NN

PREPROCESSING



['EAST', 'NORTH', 'v', 'ith_bm', 'smb', 'z', 's', 'temp', 'vx', 'vy']

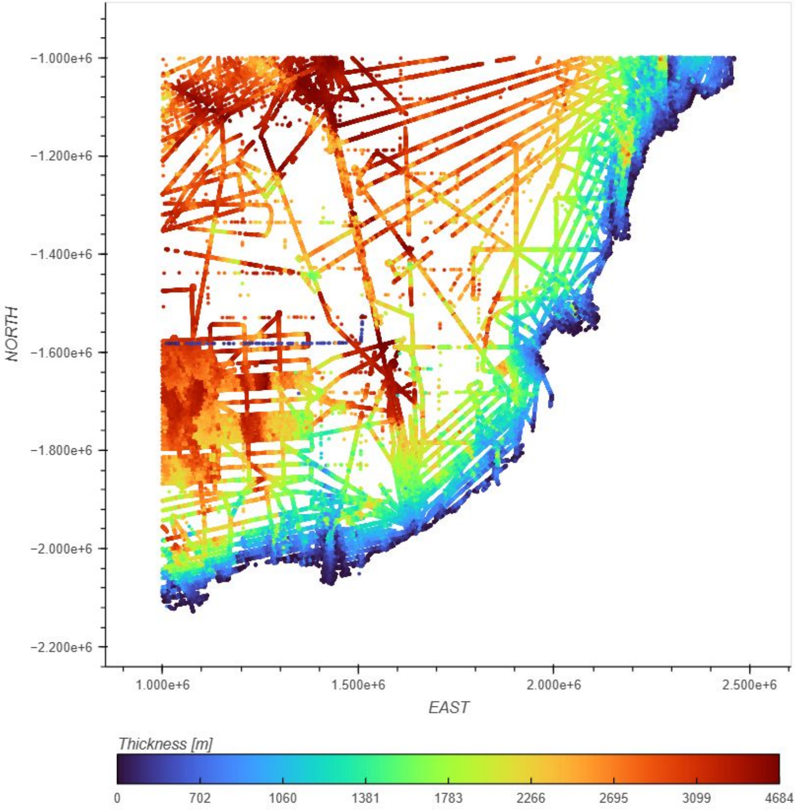
Interpolated map

Creating images

Data for CNN +NN

PREPROCESSING

The target points



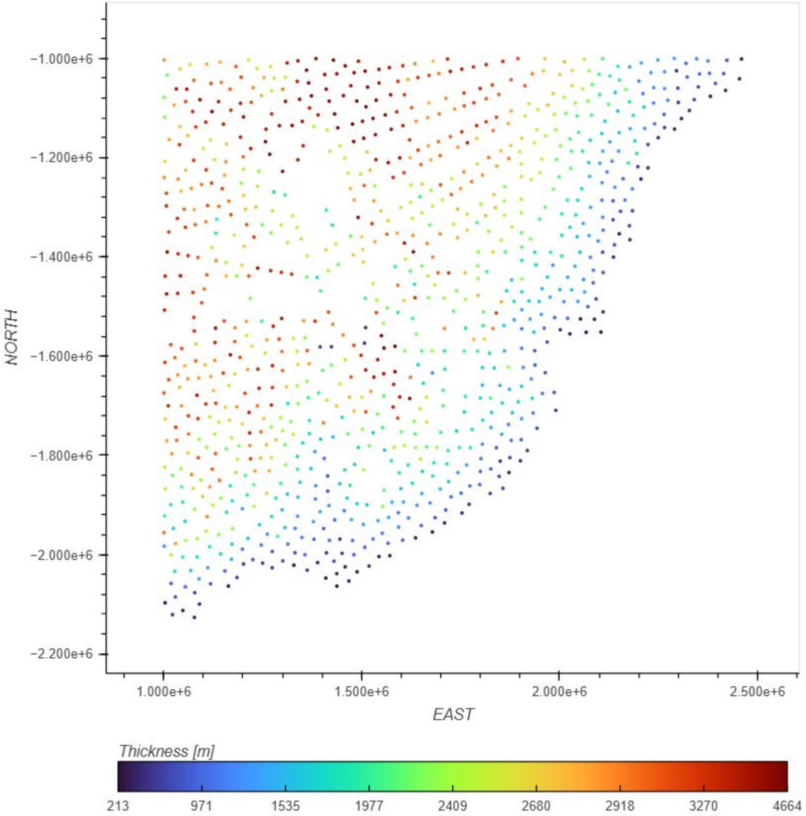
Interpolated map

Creating images

Data for CNN +NN

PREPROCESSING

Choosing random points and securing a minimum distance



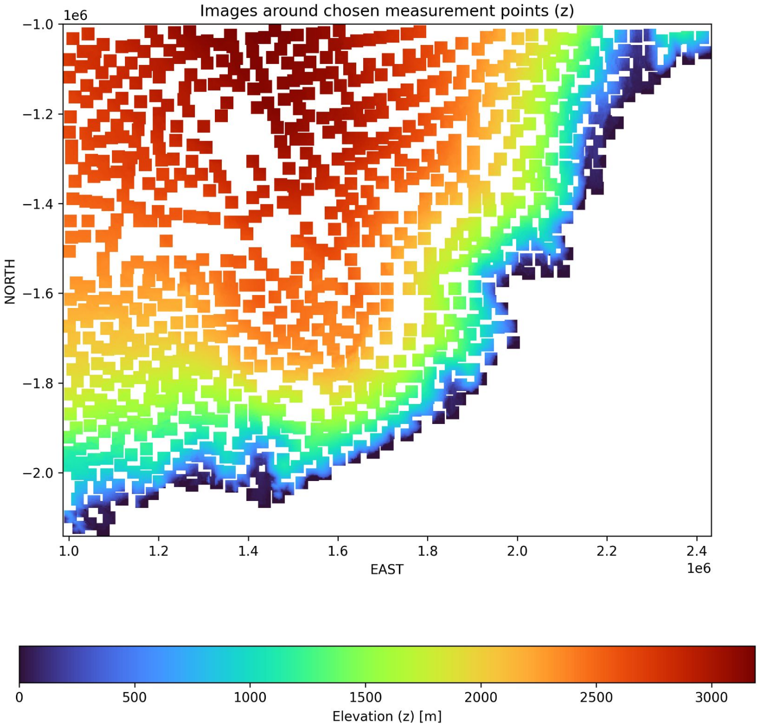
Interpolated map

Creating images

Data for CNN +NN

PREPROCESSING

Create images around targets with data from interpolated map



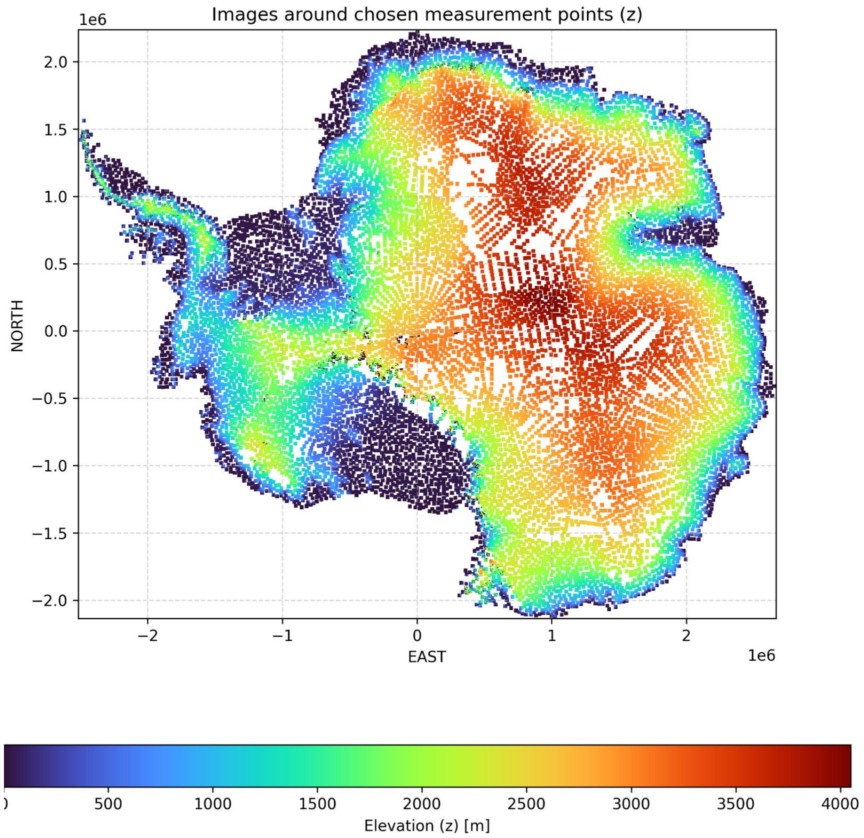
Interpolated map

Creating images

Data for CNN +NN

PREPROCESSING

Samples: 12103
Image size: 11x11
Channels: 9



Interpolated map

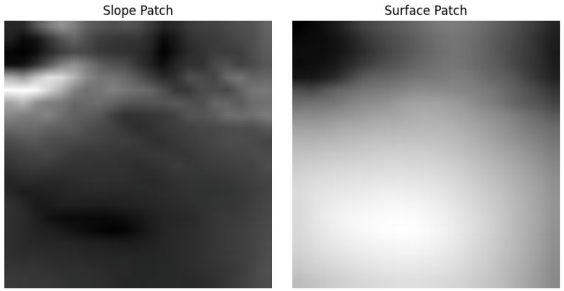
Creating images

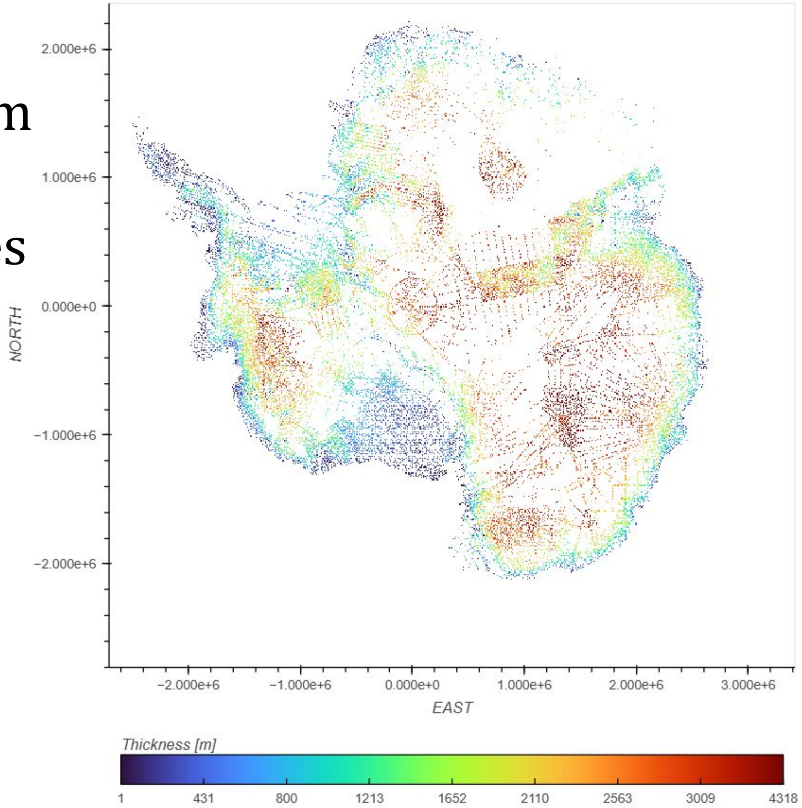
Data for CNN +NN

PREPROCESSING

20.000 random points separated by at least 10km

Five 10km x 10km patches around each point with different resolutions





Interpolated map

Creating images

Data for CNN +NN

THE MODELS

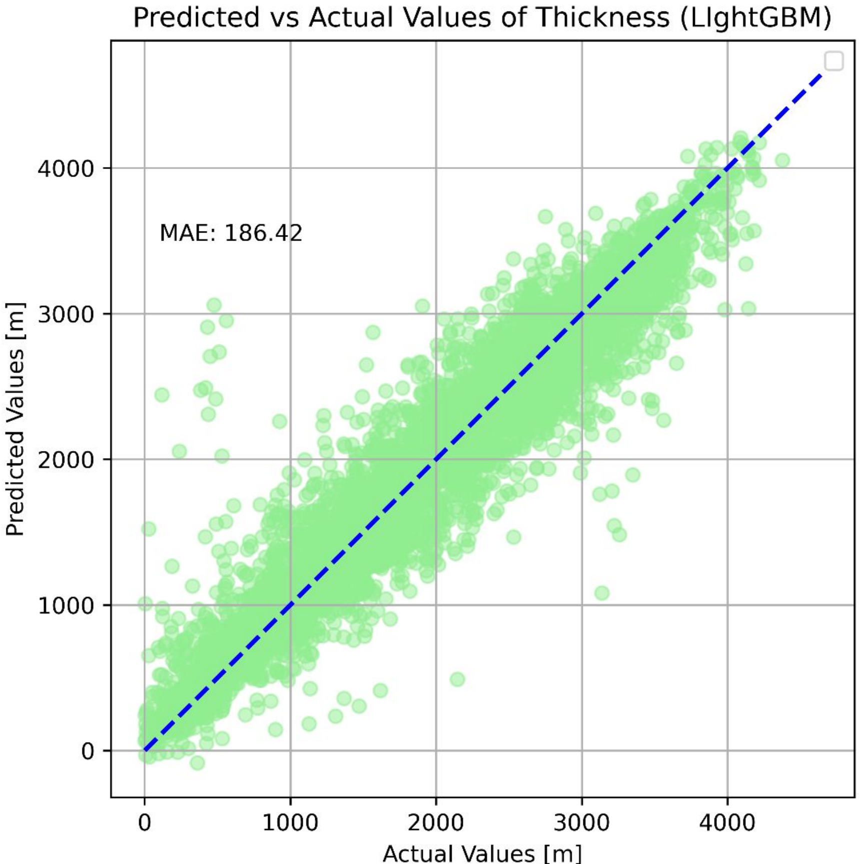
Four models of interest:

- LightGBM (tabular data)
- CNN (images)
    - CNN + LightGBM (images)
- CNN + NN (images + tabular data)

THE MODELS

Simple LGBM

Tried HP Opt:
best result default



LightGBM

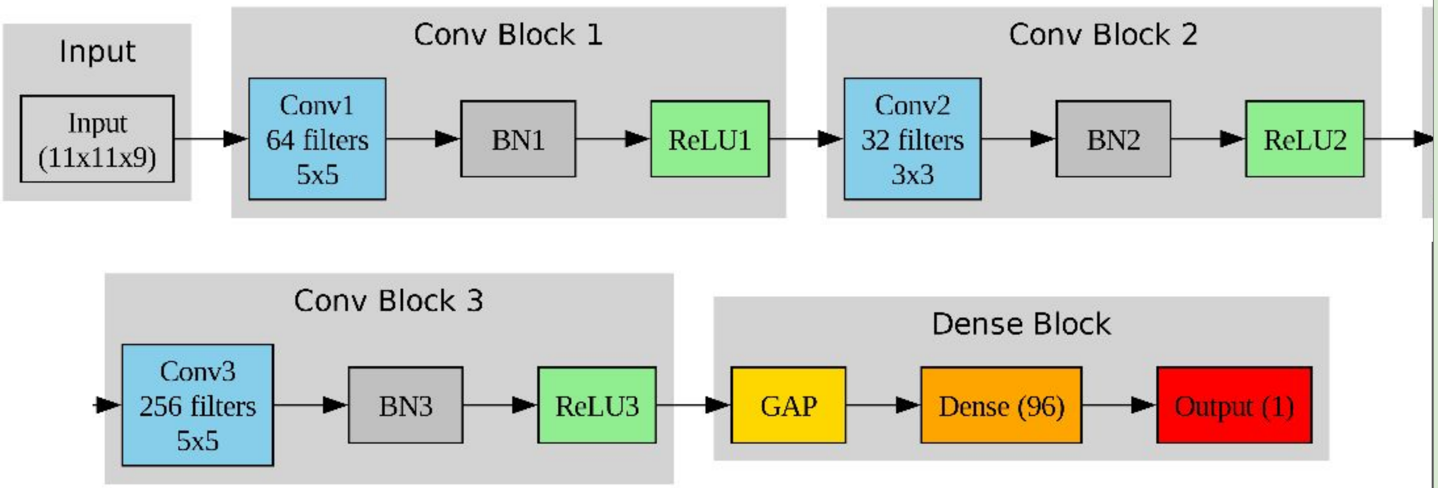
CNN with Tensorflow

+LightGBM

Pytorch CNN+NN

Summary

THE MODELS



Filters, kernel sizes and learning rate is tuned with keras-tuner

LightGBM

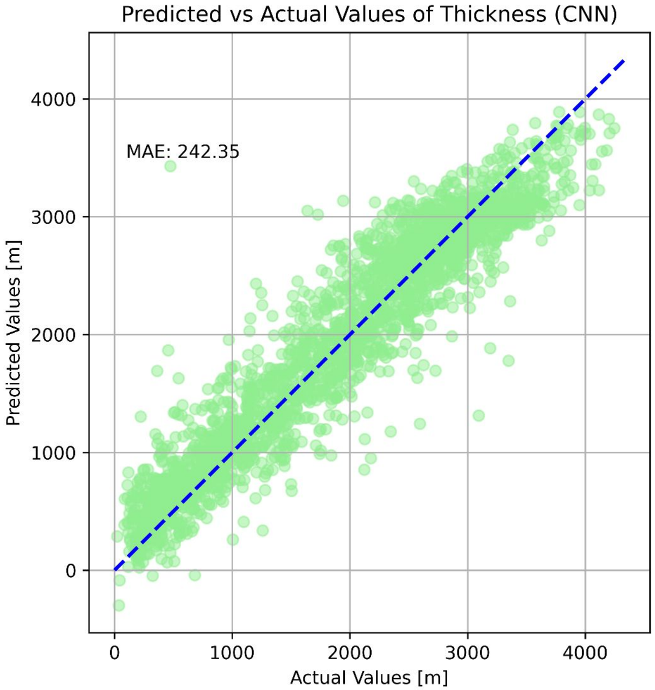
CNN with Tensorflow

+LightGBM

Pytorch CNN+NN

Summary

THE MODELS



LightGBM

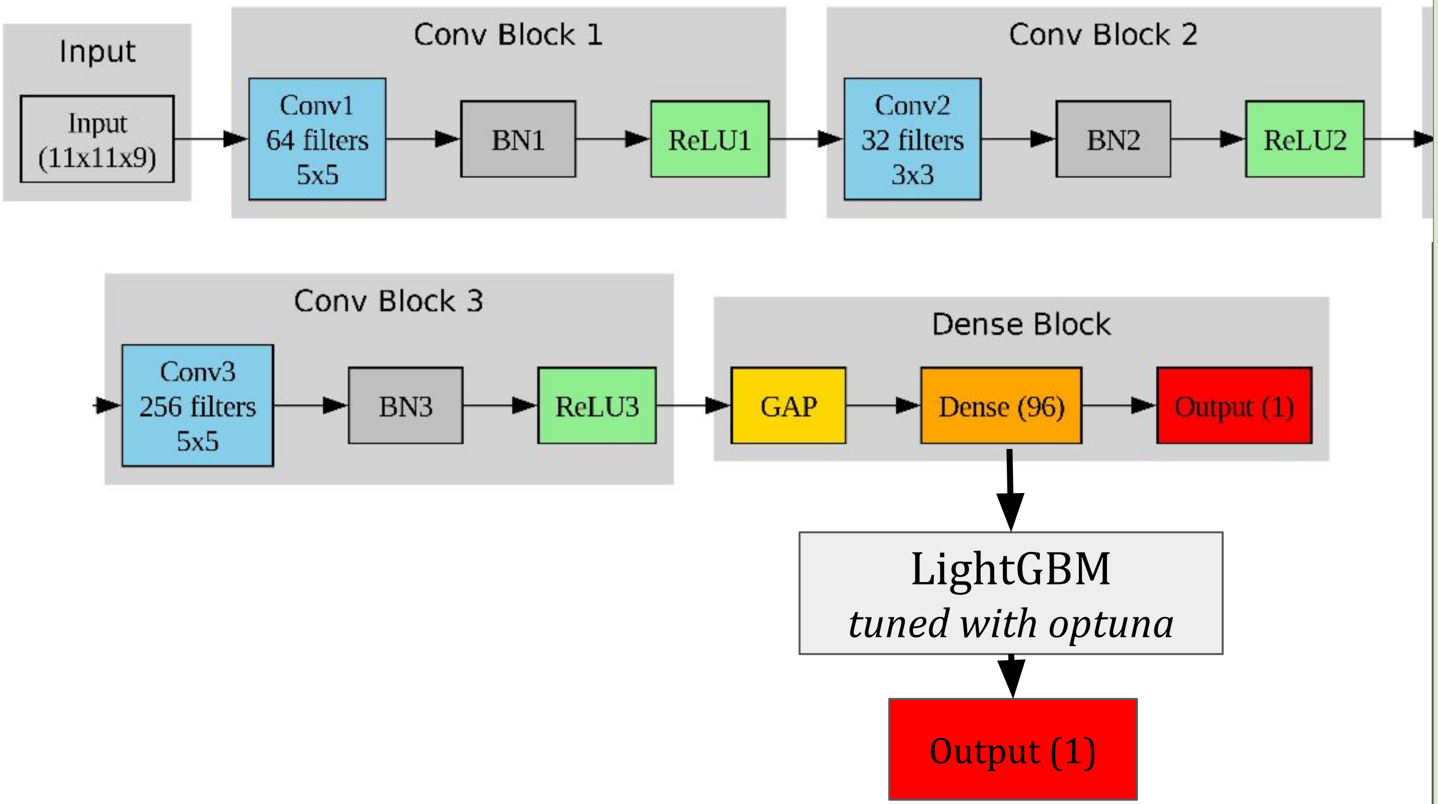
CNN with Tensorflow

+LightGBM

Pytorch CNN+NN

Summary

THE MODELS



LightGBM

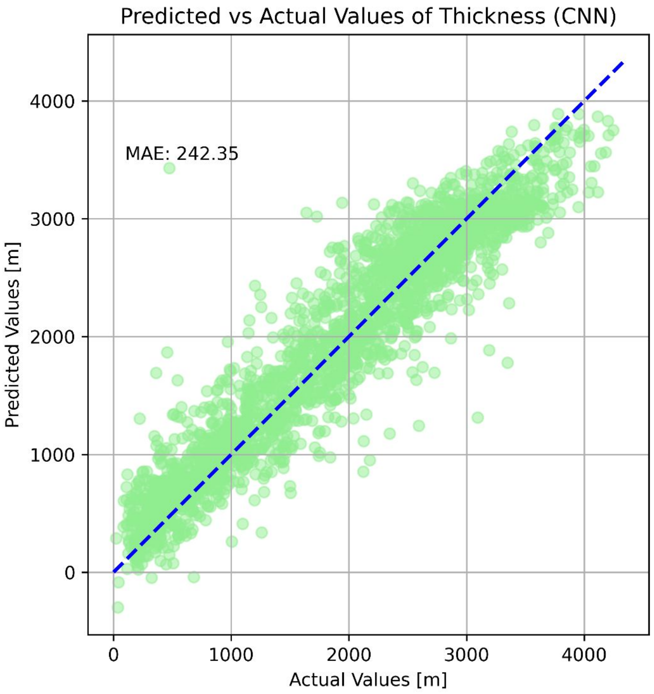
CNN with Tensorflow

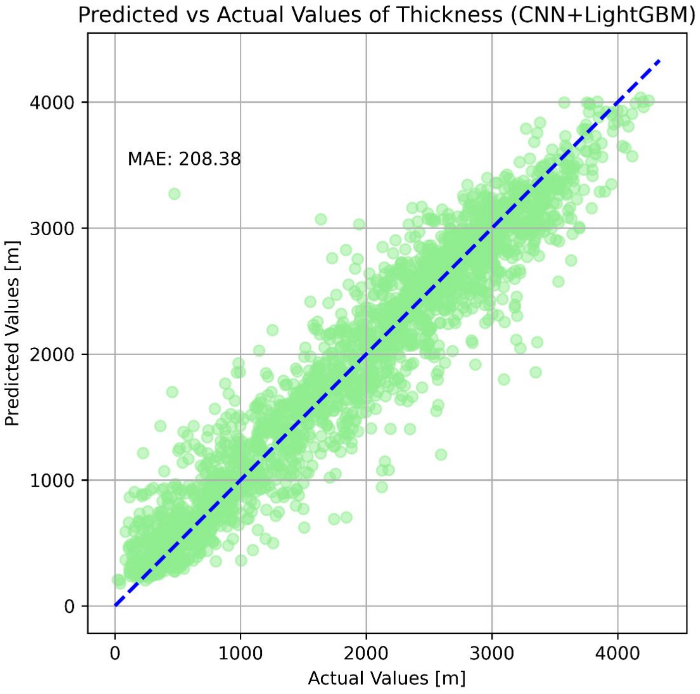
+LightGBM

Pytorch CNN+NN

Summary

THE MODELS





LightGBM

CNN with Tensorflow

+LightGBM

Pytorch CNN+NN

Summary

THE MODELS

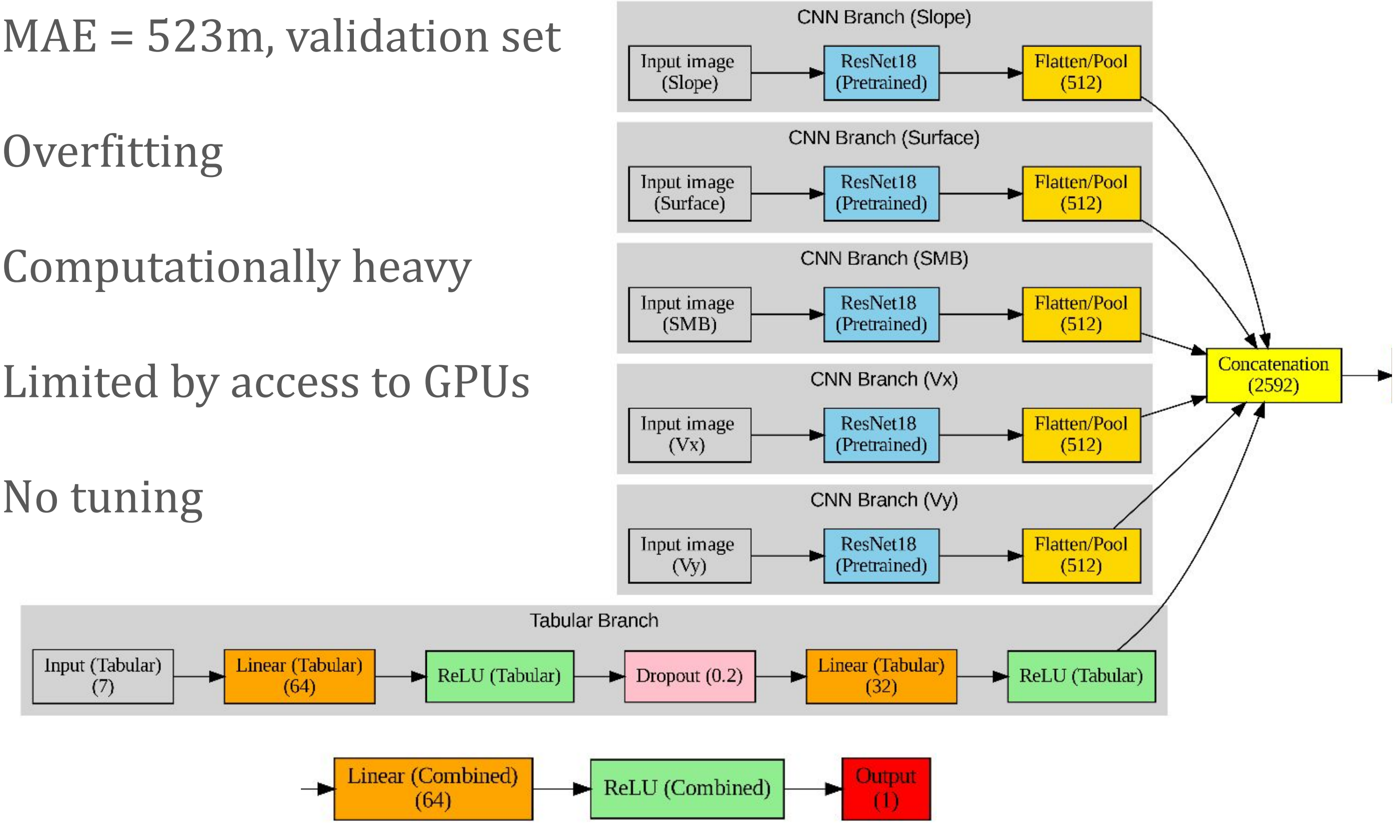
MAE = 523m, validation set

Overfitting

Computationally heavy

Limited by access to GPUs

No tuning



LightGBM

CNN with Tensorflow

+LightGBM

Pytorch CNN+NN

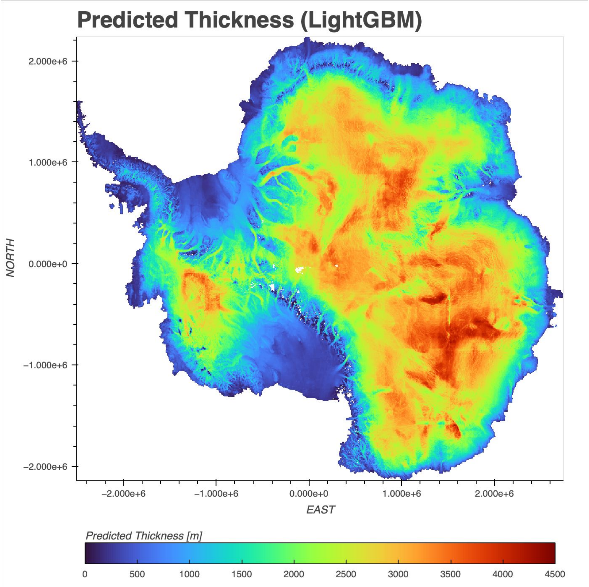
Summary

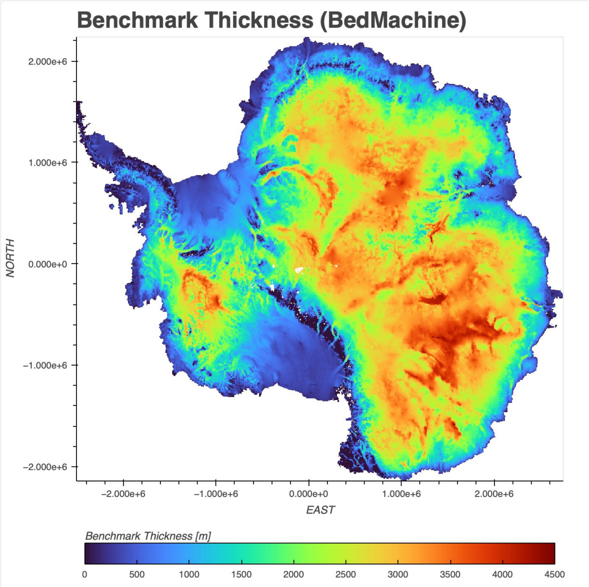
THE MODELS

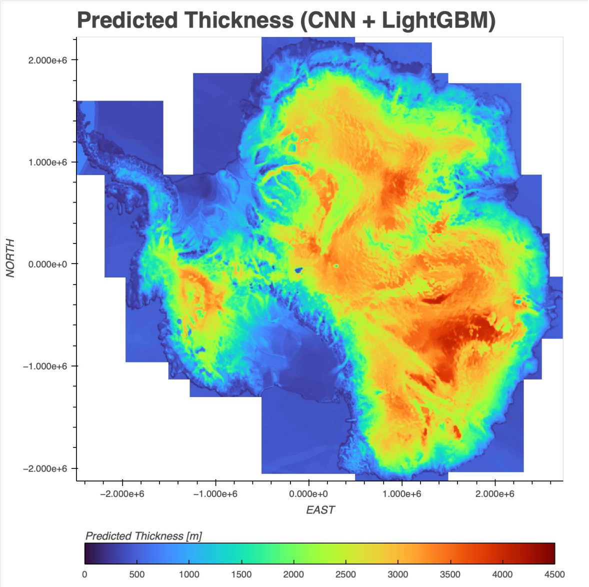
Results from training:

- **LightGBM:** MAE = 186.42
- **CNN:** MAE = 242.35
    - **CNN + LightGBM:** MAE = 208.38
- **CNN + NN:** MAE = 523

RESULTS







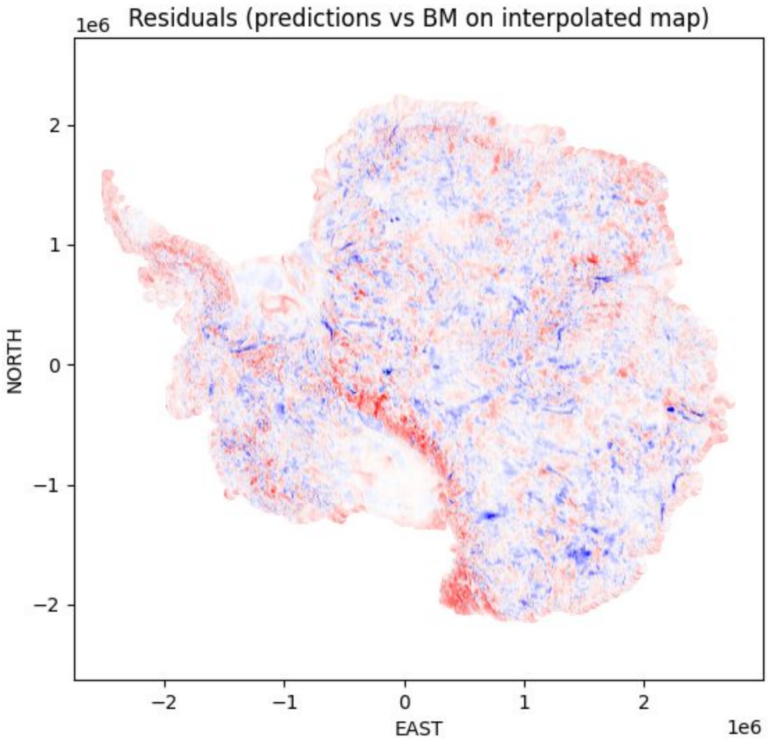
Volume est: 26.3e6 km^3

Volume est: 27e6 km^3

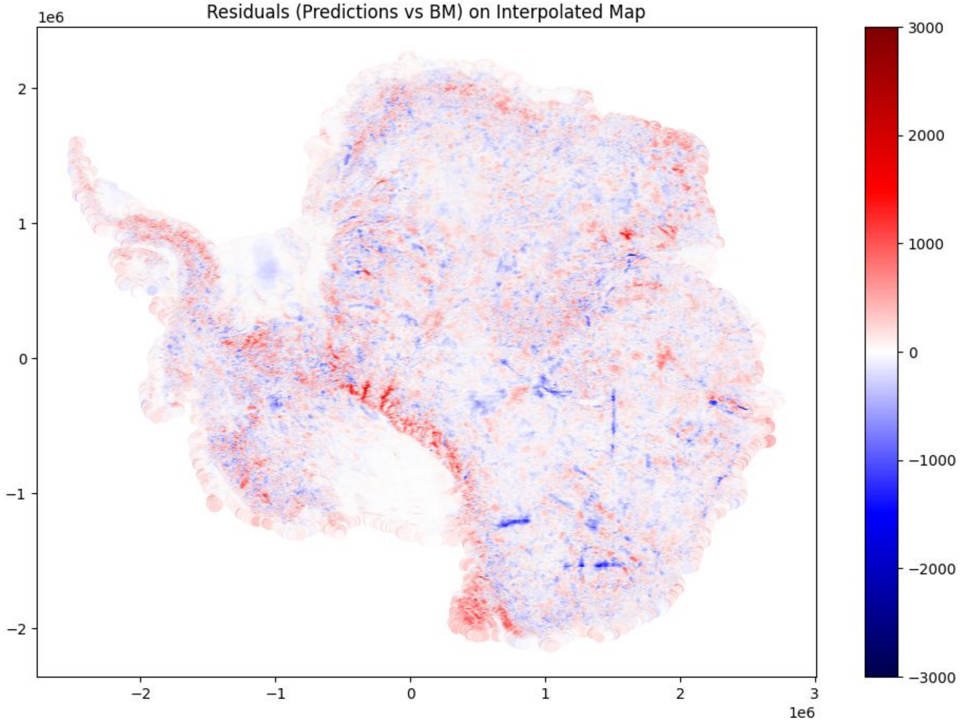
Volume est: 27.8e6 km^3

RESULTS

CNN + LGBM



LGBM



OUTLOOK

- Introducing new variables:
  - Rock or no rock?
  - Floating ice or grounded ice?
- Work to identify correlation
  - Possible overfitting?
- Computer power

CONCLUSION

- Model
  - LightGBM best, tuning?
- Results
  - Good for large parts of Antarctica
- ML + Physics (PIML)
- Data preprocessing is important!

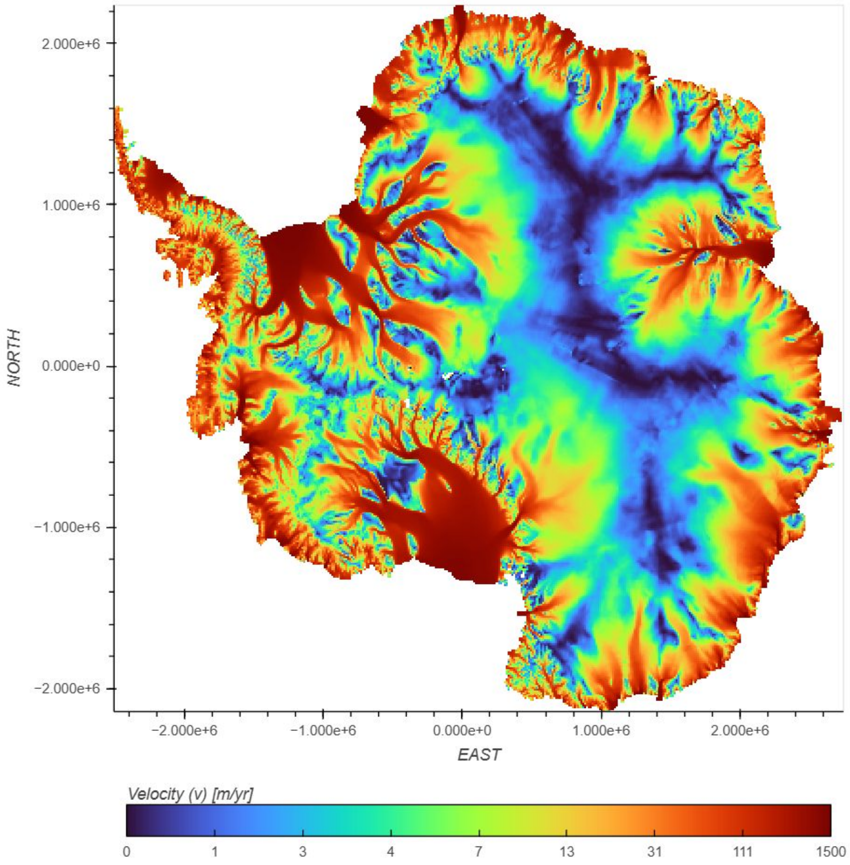
APPENDIX

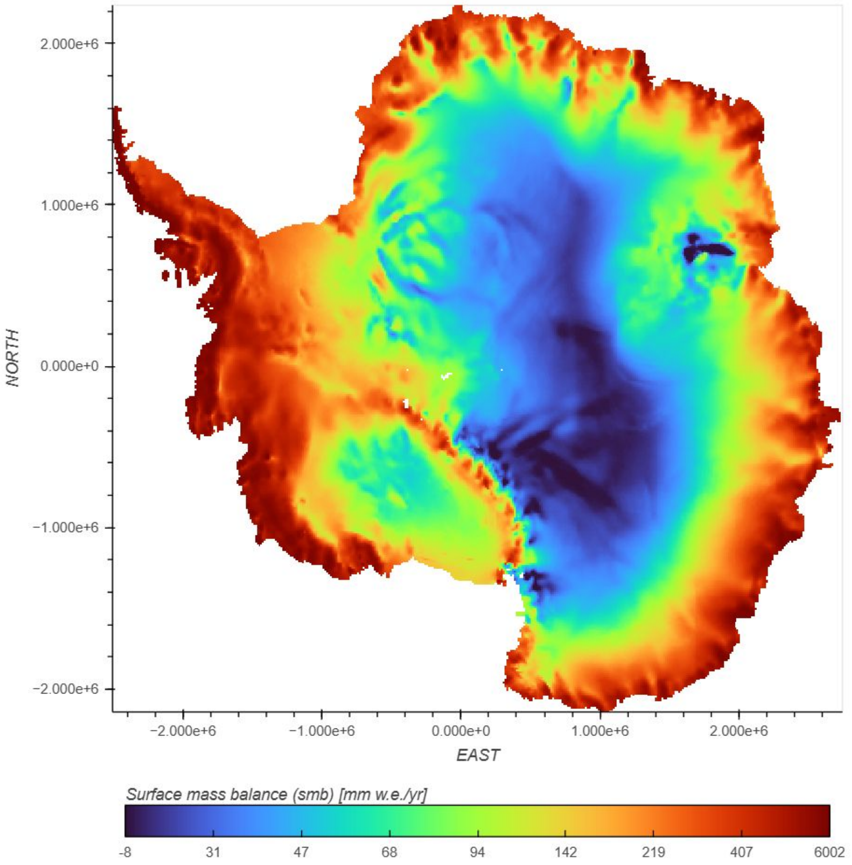
Interpolated map data structure

	EAST	NORTH	v	ith_bm	smb	z	s	temp	vx	
0	-9.648837e+03	2.237066e+06	766.738880	181.808865	425.540239	19.628939	0.003440	261.330896	-154.084036	751.096
1	-7.043673e+03	2.237066e+06	778.073047	180.071241	426.613247	19.443316	0.000901	261.328837	-162.828831	760.844
2	-4.438508e+03	2.237066e+06	782.989503	170.621189	428.149692	18.431648	0.000617	261.327784	-164.894030	765.429
3	-1.833344e+03	2.237066e+06	784.883535	156.723843	430.348457	16.943702	0.001370	261.332561	-162.327349	767.914
4	-3.570048e+04	2.234461e+06	716.191136	117.184048	410.579005	12.703958	0.011815	261.474882	-106.576447	708.216
...	...	...	...	...	...	...	...	...	...	
1975291	1.053258e+06	-2.139611e+06	2.228523	90.132909	418.930191	215.075900	0.034924	259.078931	-2.201916	-0.343
1975292	1.055863e+06	-2.139611e+06	1.143510	135.768250	410.513393	90.458594	0.064816	259.258269	-1.127426	0.191
1975293	1.048048e+06	-2.142216e+06	13.212093	82.234228	436.594883	67.931569	0.093504	258.817662	-8.664803	-9.973
1975294	1.050653e+06	-2.142216e+06	5.153131	121.287103	429.531392	157.341866	0.055818	258.992232	-1.527397	-4.921
1975295	1.053258e+06	-2.142216e+06	2.111907	151.928517	421.002829	138.470849	0.050553	259.158605	-0.442218	-2.065

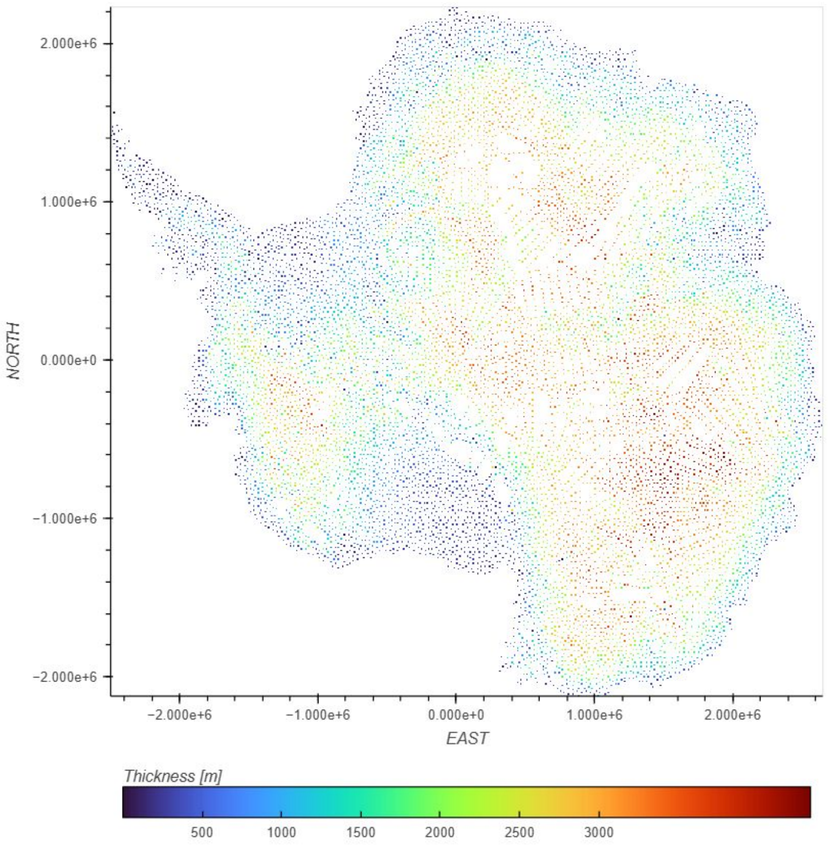
1975296 rows × 10 columns

Interpolated map (some example plots)





Chosen target points for CNN + LightGBM



Choosing minimum distance between target points

- For LightGBM the min distance was 15 km (29477 points)
  - For 25 km min distance the LightGBM performs approximately as well as the CNN + LightGBM
- For CNN (tensorflow) the min distance was 25 km (12103 points)
  - Choosing 15 km results in 11 hours computational time for just creating the training images
- For CNN (Pytorch) the min distance was 10 km, but we included an upper limit on the number of points (20.000)

Making images for CNN

- Long computation time: (shown here is for 11x11 imagesize)

```
Extracting subgrids: 100%|██████████| 12104/12104 [4:11:04<00:00,  1.24s/it]
```

- We tried different image sizes to investigate if knowing more about the surroundings decreases the MAE
    - Using imagesize 21x21 didn’t significantly decrease the MAE and the computational time was even higher

Making images for CNN

Data structure:

► Dimensions:	(**sample**: 12103, y: 11, x: 11)		
▼ Coordinates:			
x	(sample, x)	float64	...
y	(sample, y)	float64	...
sample	(sample)	int32	0 1 2 3 ... 12099 12100 12101 12102
▼ Data variables:			
v	(sample, y, x)	float64	...
ith_bm	(sample, y, x)	float64	...
smb	(sample, y, x)	float64	...
z	(sample, y, x)	float64	...
s	(sample, y, x)	float64	...
temp	(sample, y, x)	float64	...
vx	(sample, y, x)	float64	...
vy	(sample, y, x)	float64	...
THICK	(sample)	float64	...

Predicting with CNN + LightGBM

- Images were created from interpolated map with grid size 11x11
- The kernel couldn’t handle predicting on the entirety of Antarctica, so we divided it into tiles and predicted on these
    - You have to be extra careful at the borders of the tiles; we created an overlap between the tiles

```
for east_start in east_tiles:
  for north_start in north_tiles:
    tile_df = ddf[
      (ddf['EAST'] >= east_start - OVERLAP) & (ddf['EAST'] < east_start + STEP +OVERLAP) &
      (ddf['NORTH'] >= north_start - OVERLAP) & (ddf['NORTH'] < north_start + STEP + OVERLAP)
    ]
    tile = tile_df.compute()
```

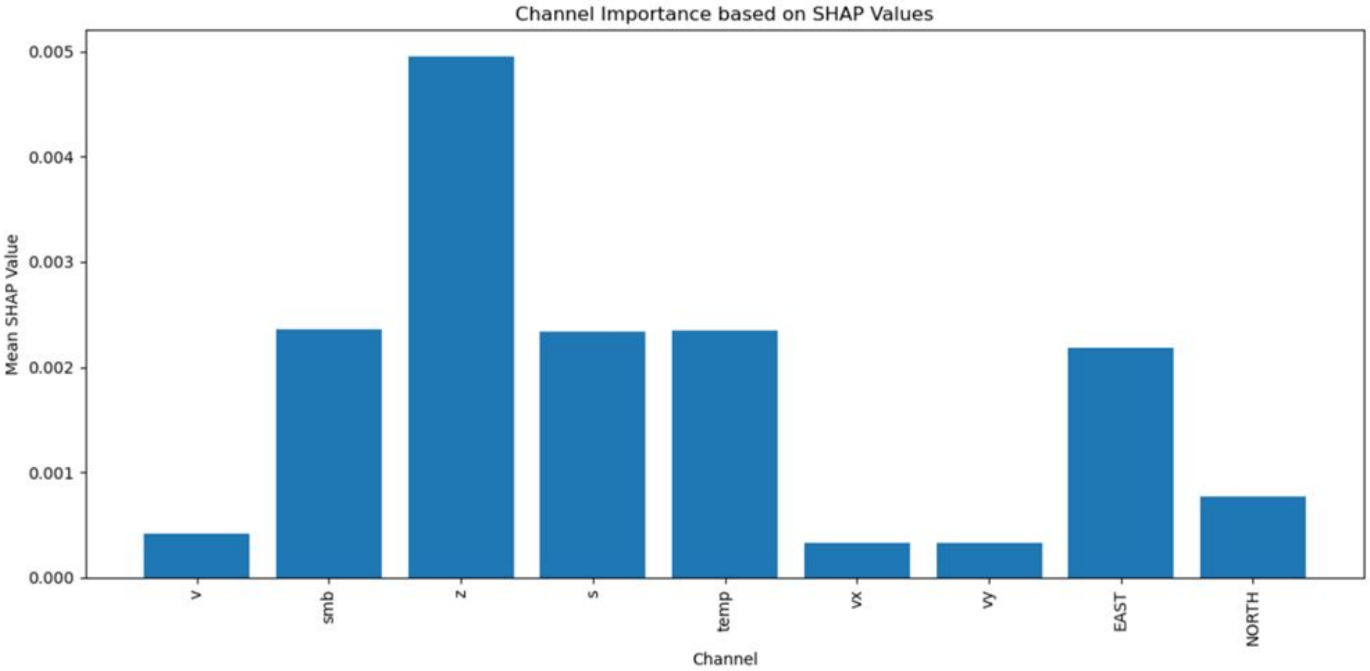
CNN architecture + LightGBM HP

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 11, 11, 9)	0
conv2d (Conv2D)	(None, 11, 11, 64)	14,464
batch_normalization (BatchNormalization)	(None, 11, 11, 64)	256
re_lu (ReLU)	(None, 11, 11, 64)	0
conv2d_1 (Conv2D)	(None, 11, 11, 32)	18,464
batch_normalization_1 (BatchNormalization)	(None, 11, 11, 32)	128
re_lu_1 (ReLU)	(None, 11, 11, 32)	0
conv2d_2 (Conv2D)	(None, 11, 11, 256)	205,056
batch_normalization_2 (BatchNormalization)	(None, 11, 11, 256)	1,024
re_lu_2 (ReLU)	(None, 11, 11, 256)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 256)	0
penultimate (Dense)	(None, 96)	24,672
dense (Dense)	(None, 1)	97

LightGBM hyperparameters:

```
Best hyperparameters:  {'num_leaves': 129, 'learning_rate': 0.02889607022096179,
'n_estimators': 704, 'max_depth': 15, 'subsample': 0.8717906147804415,
'colsample_bytree': 0.6129414641677116}
```

CNN tensorflow channel importance

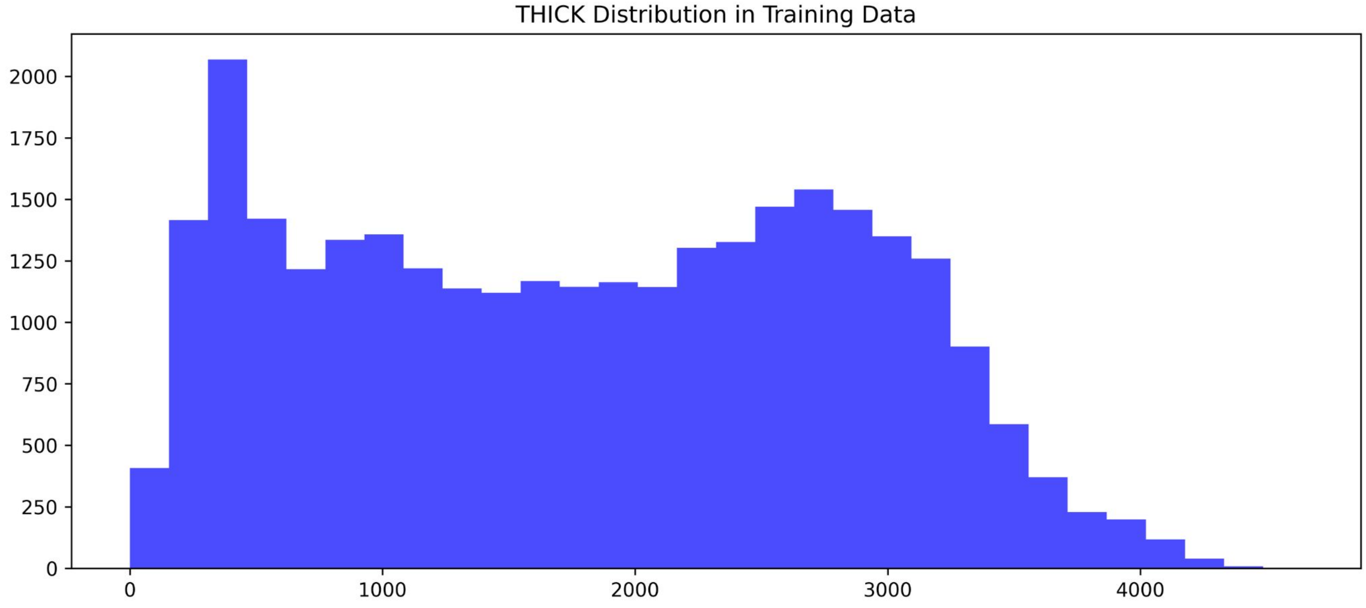


LightGBM-only model HP

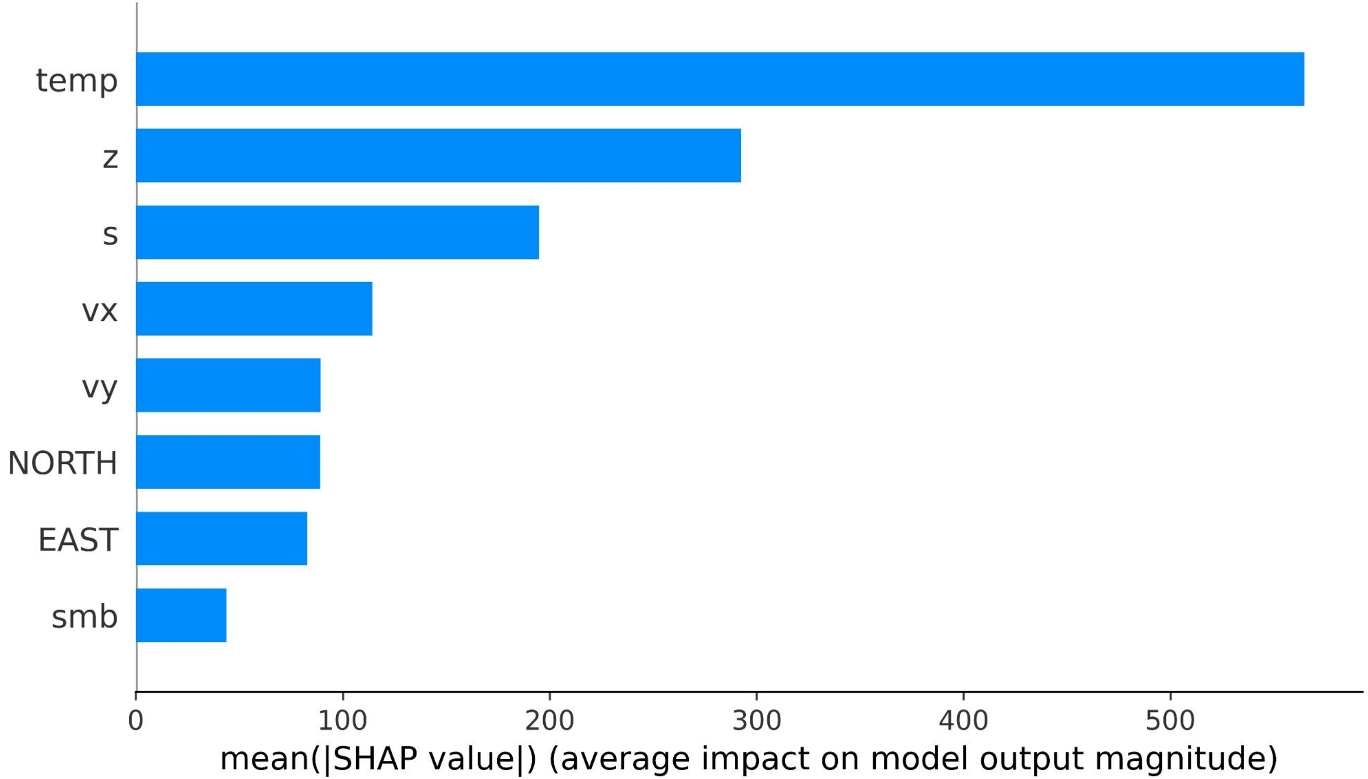
LightGBM Hyperparameters:

```
'objective': 'regression',
    'metric': 'mae',
    'boosting_type': 'gbdt',
    'num_leaves': 31,
    'learning_rate': 0.1,
    'feature_fraction': 0.9,
```

LightGBM-only training target distribution



Shap values for LightGBM-only model



Initial approach to interpolating maps

The preliminary interpolation was done to have data at all points, to easier extract values for the exact same coordinates.

These would then be filtered by distance and used as midpoints for smaller maps from our mapped data.

They also allowed for ‘map selection’, from the interpolations themselves, as we did interpolations for all variables.

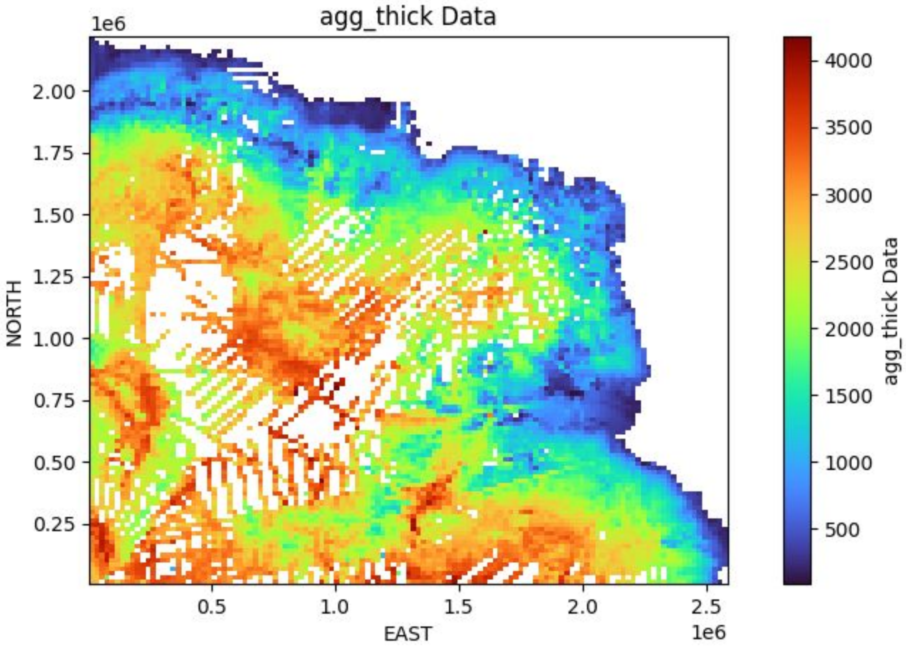
The plan was to test our algorithms on easily extracted data.

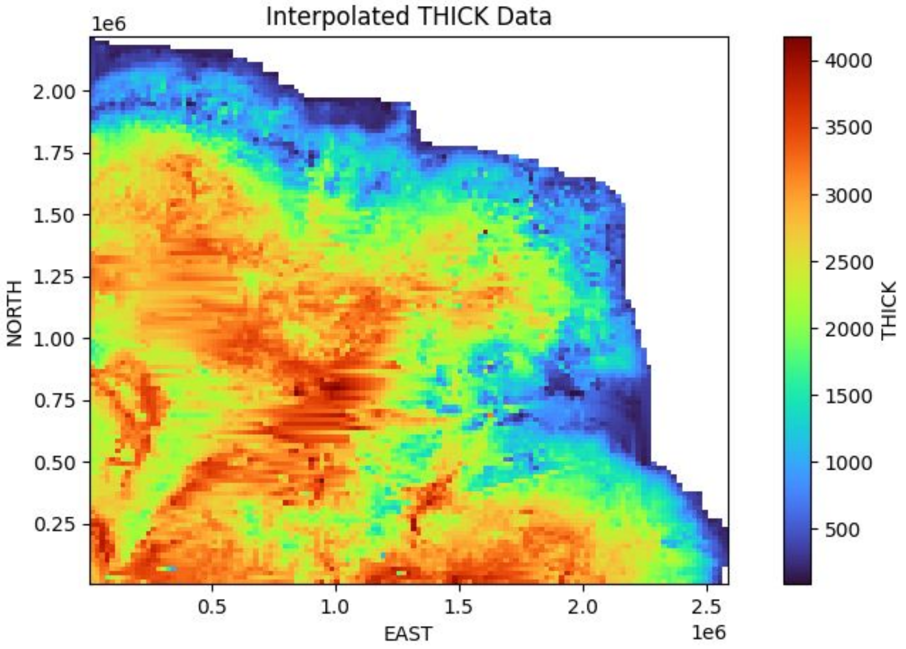
They were not used for any of the final models.

Initial approach to interpolating maps

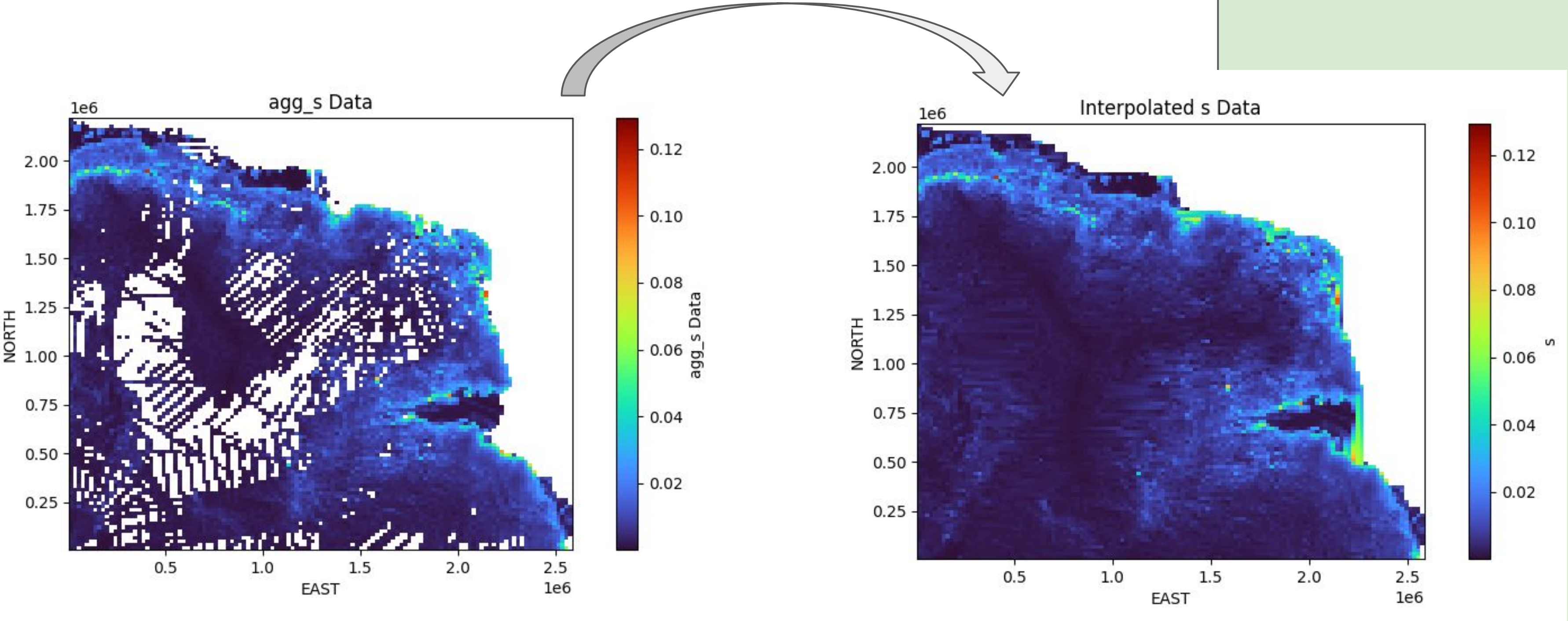
Raw data

Linear interpolation

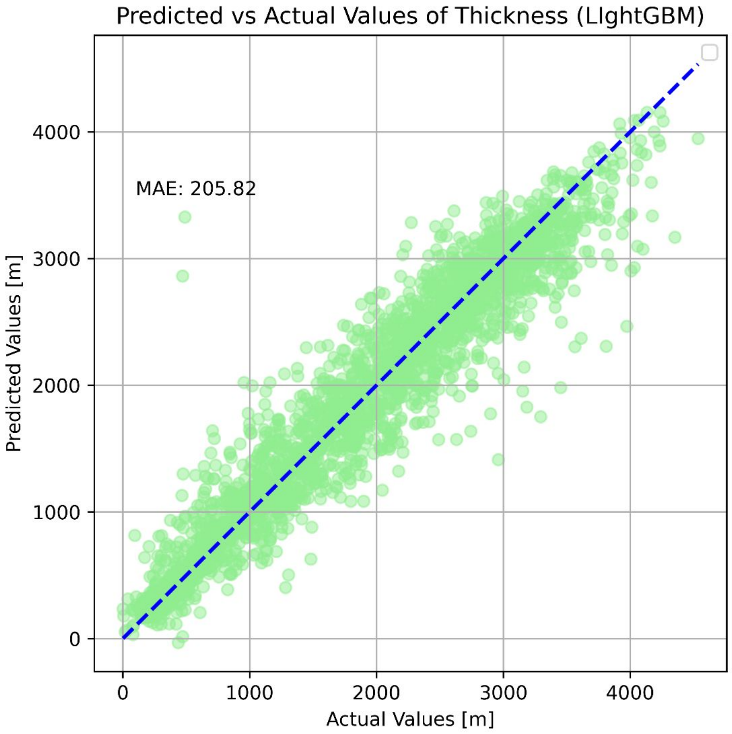


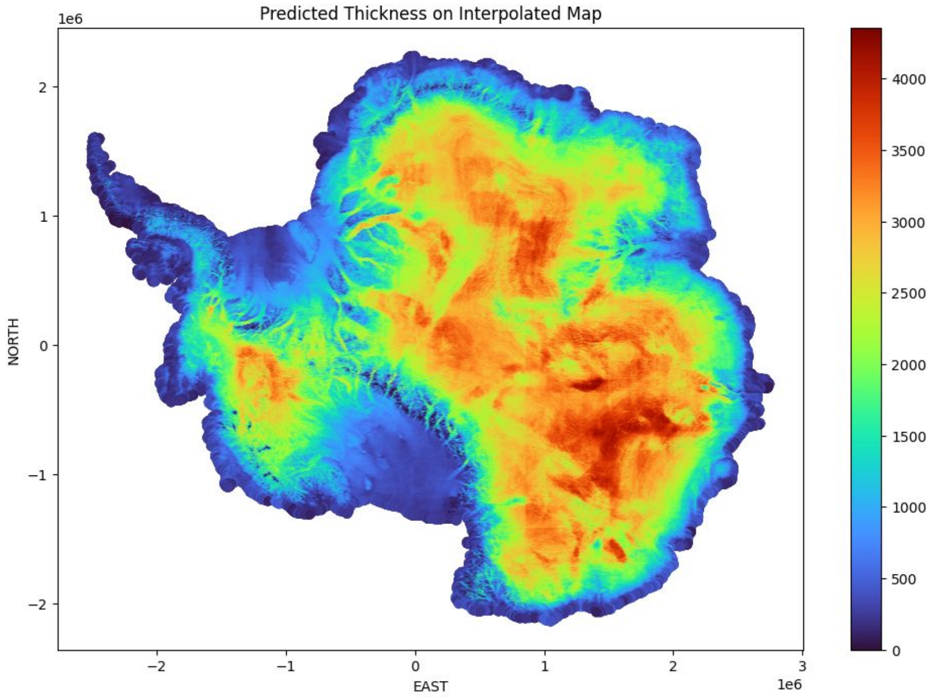


Initial approach to interpolating maps



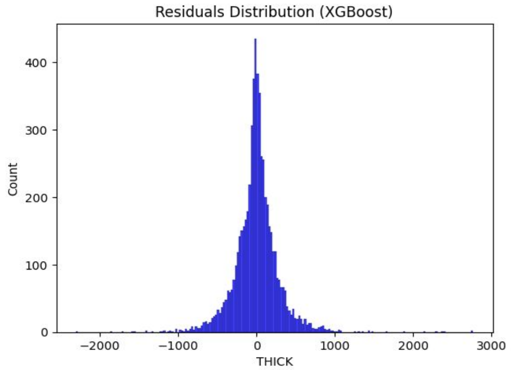
LightGBM with distance 25 km bw points

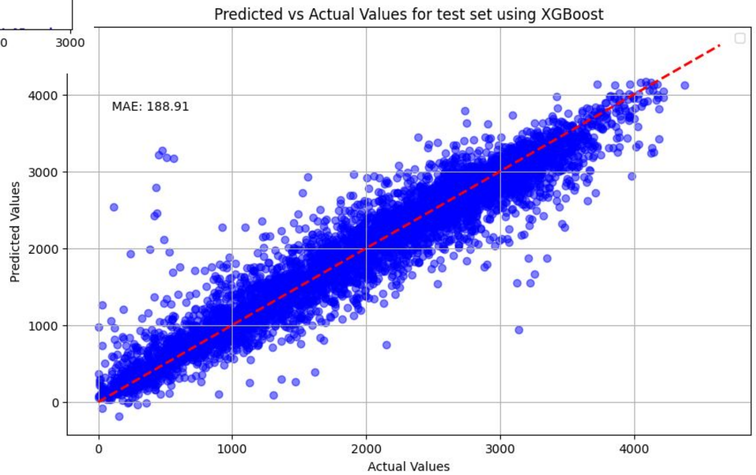


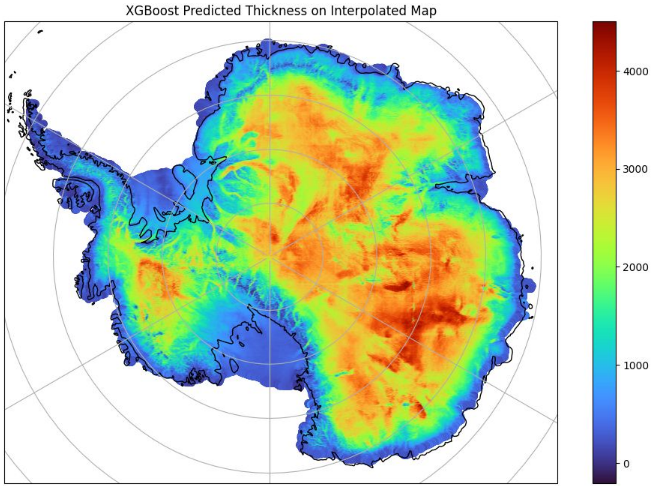


Volume est: 26.2e6 km^3

XGBoost with distance 15 km bw points



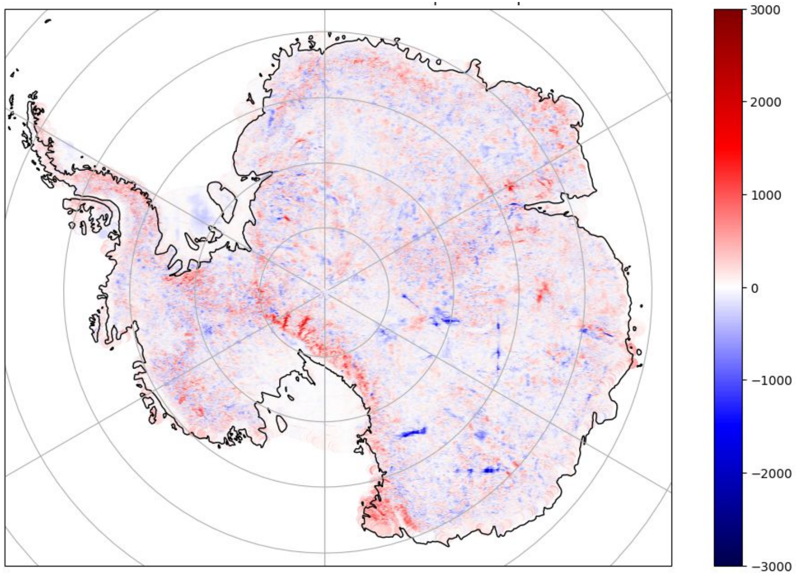


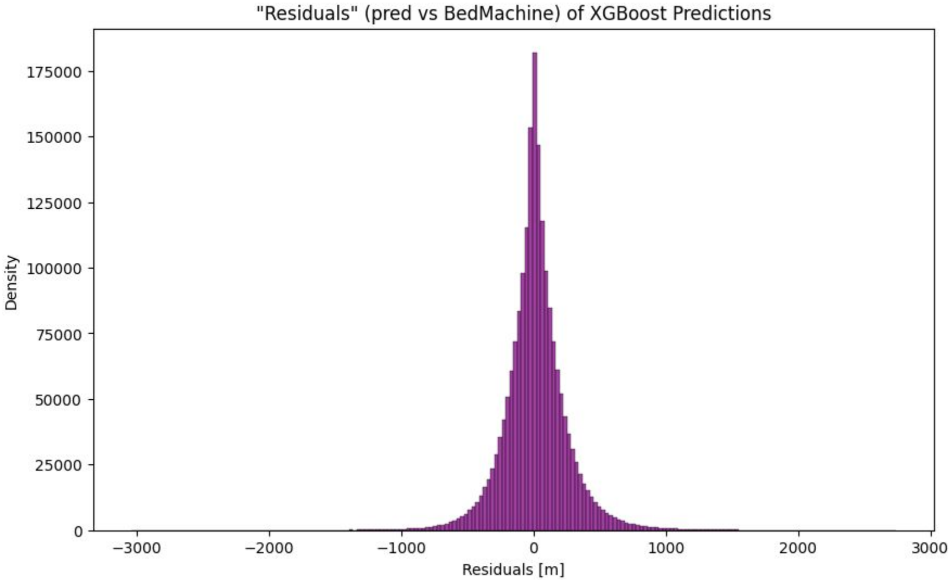


Volume est: 26.2e6 km^3

Benchmark vs full prediction residuals for XGBoost model,

Mean Absolute Error with Benchmark Set: 166.5





CNN + NN overfitting