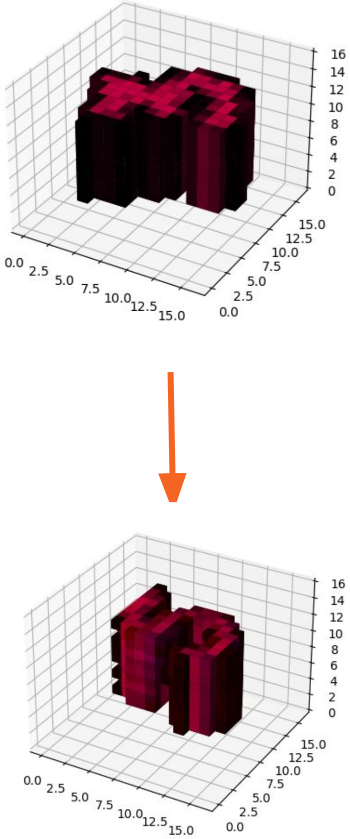
Unpaired learning with Optimal Transport on 3D MNIST images

Artem Chemodanov, Denis Kukushkin, Mikhail Shvetsov, Nikita Sushko, Roman Khalikov

Motivation

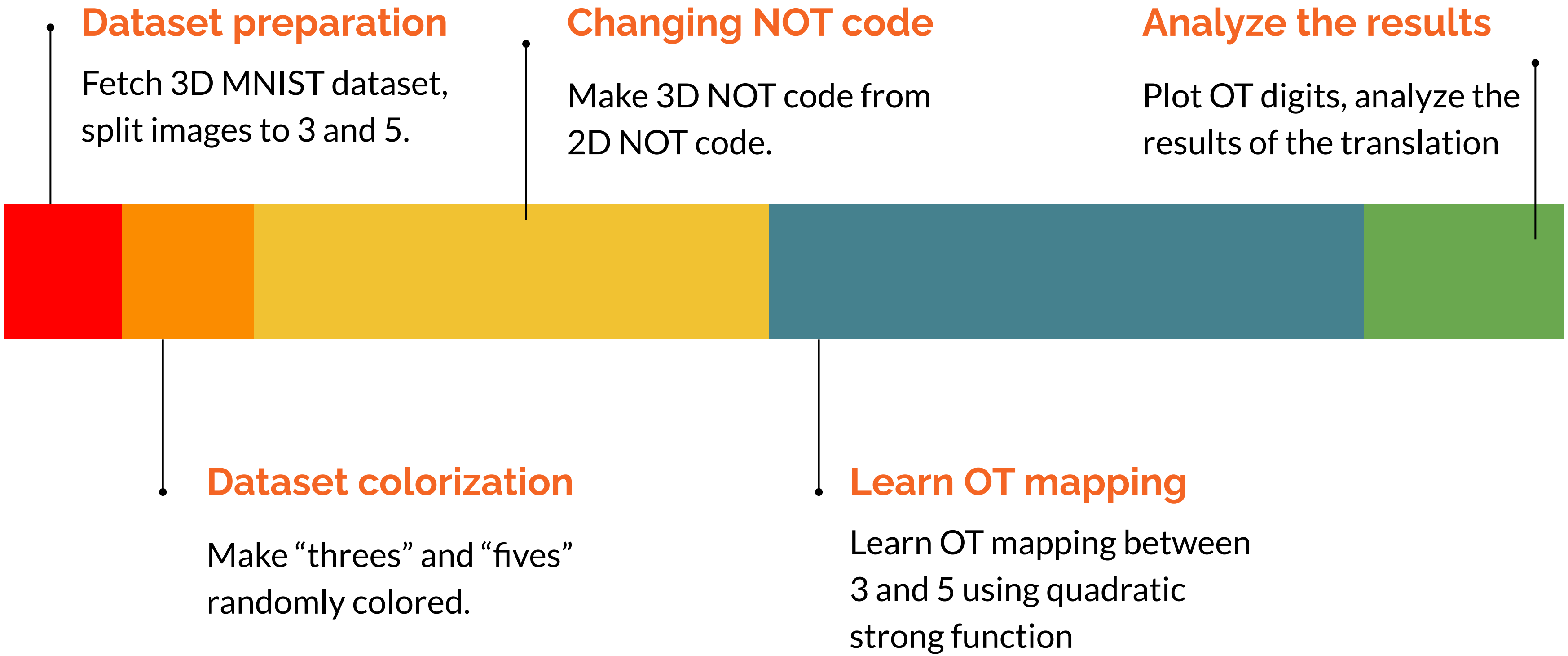
Computational Optimal Transport (OT) is a set of tools, which provide a way of transforming one distribution into another with the minimal effort.

Recent papers[1][2] show capability of OT to correctly translate 2D images. In this project, we are to implement and train OT approach on 3D MNIST digits dataset and analyze the applicability of OT method to 3D image-to-image translation.



[1] Korotin, Alexander et al. “Kernel Neural Optimal Transport.” *ArXiv* abs/2205.15269 (2022): n. Pag.
[2] Gazdieva, Milena & Rout, Litu & Korotin, Alexander & Filippov, Alexander & Burnaev, Evgeny. (2022). An Optimal Transport Perspective on Unpaired Image Super-Resolution.

Problem statement



Dataset preparation

Fetch 3D MNIST dataset, split images to 3 and 5.

Dataset colorization

Make “threes” and “fives” randomly colored.

Changing NOT code

Make 3D NOT code from 2D NOT code.

Learn OT mapping

Learn OT mapping between 3 and 5 using quadratic strong function

Analyze the results

Plot OT digits, analyze the results of the translation

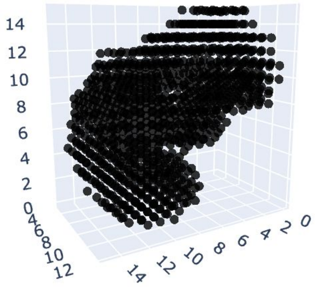
Dataset preparation

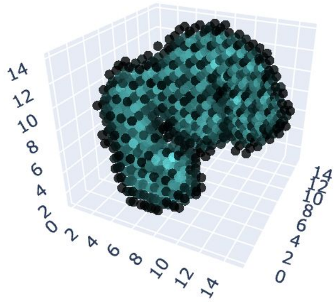
➔ **3D MNIST dataset from kaggle**
On Kaggle there is a dataset of handwritten MNIST digits converted into voxels with random rotation on X, Y and Z axis and noize.

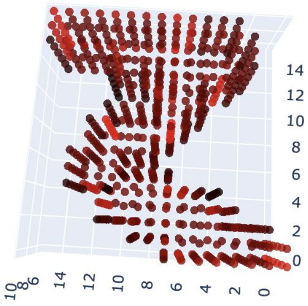
This was the recommended dataset for this task.

Pros: Easy to get, X, Y and Z rotation

Cons: Small size, hard to plot, hard to work with, bad binarization strategy selected by authors







Dataset preparation

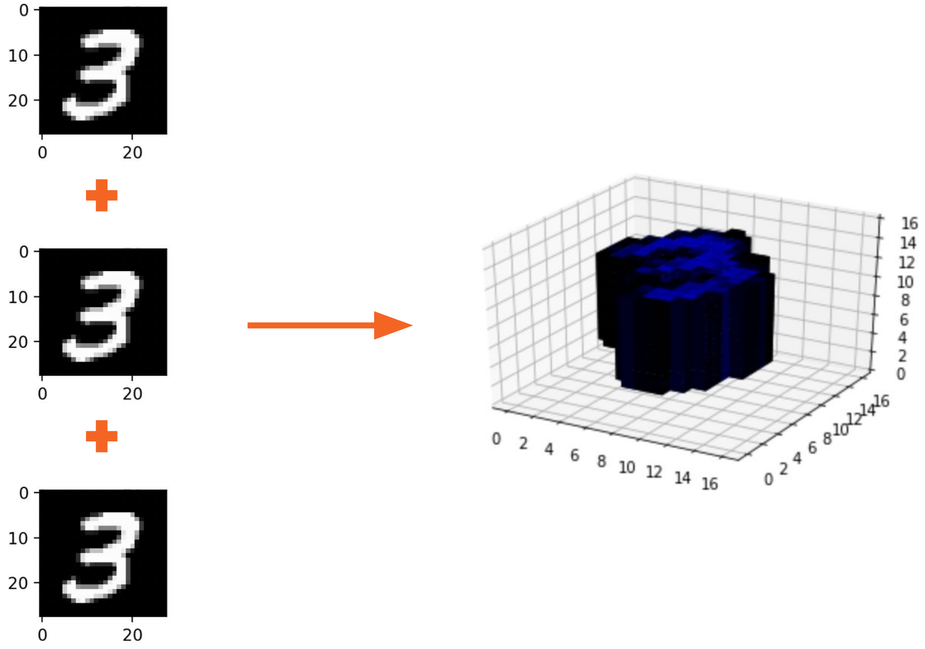
➔ **Converting 2D MNIST to voxels**

We can convert 2D MNIST to 3D by repeating the 2D images N times by Z axis and zeroing some layers.

This makes it possible to build our own pipeline of augmentations and increase the dataset size.

Pros: Huge dataset size, flexible pipeline building, easily plottable

Cons: No obvious way to implement Y and Z axis translation

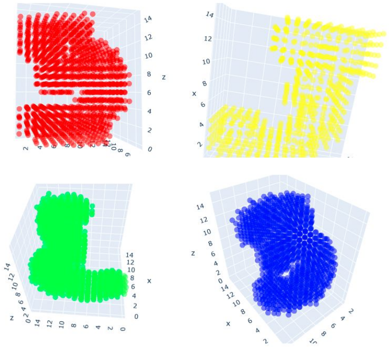


Dataset colorization

➔ 3D MNIST (Kaggle)

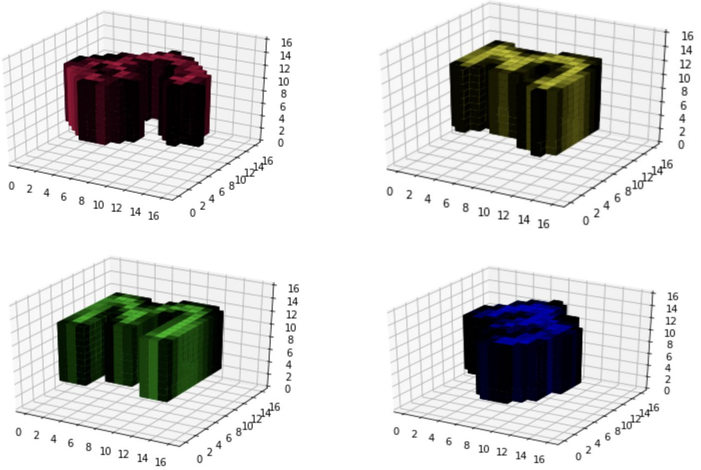
Custom colorizing function.

We assign to each non-zero voxel an RGB component, which correspond to image color, chosen randomly.



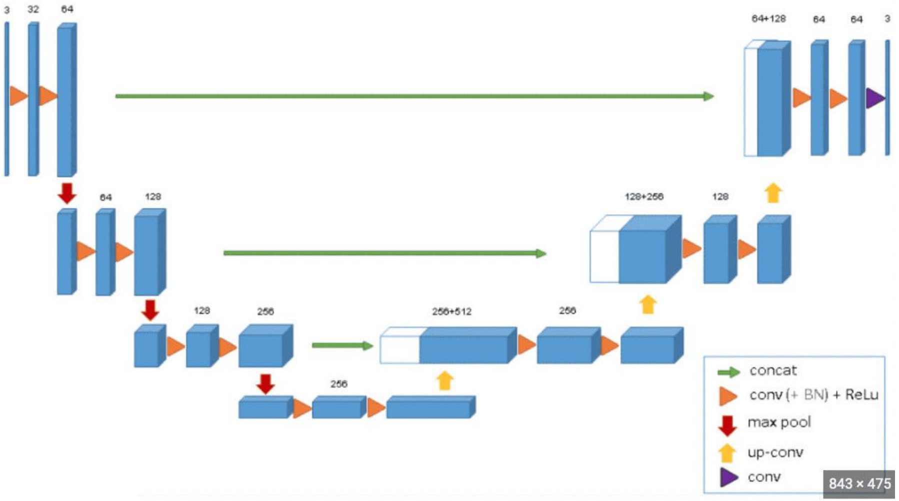
➔ 2D MNIST (Ours)

Alex Korotin's colorization function inserted into Torchvision Transform pipeline

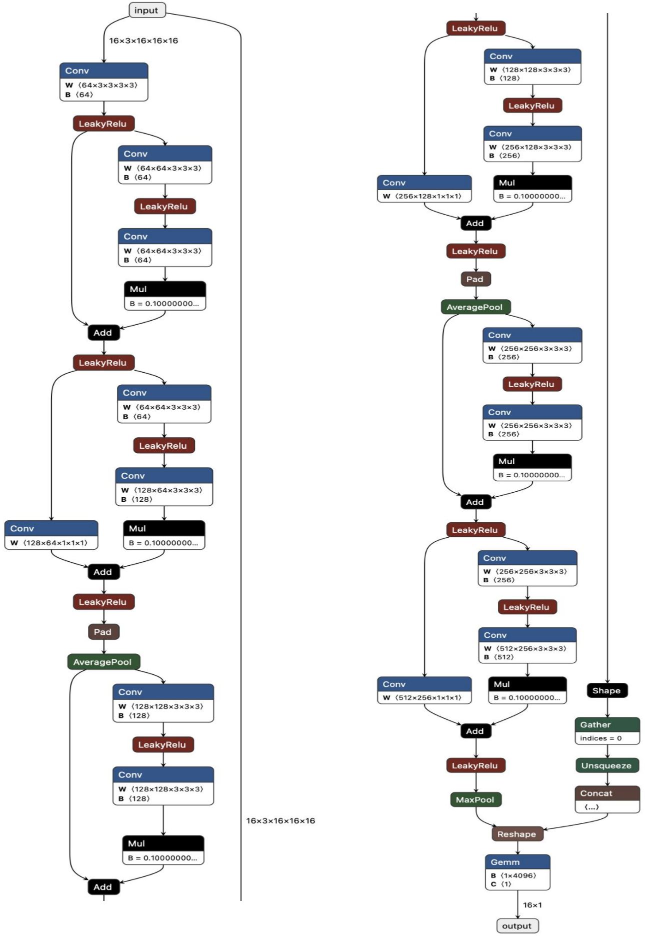


Models

UNet



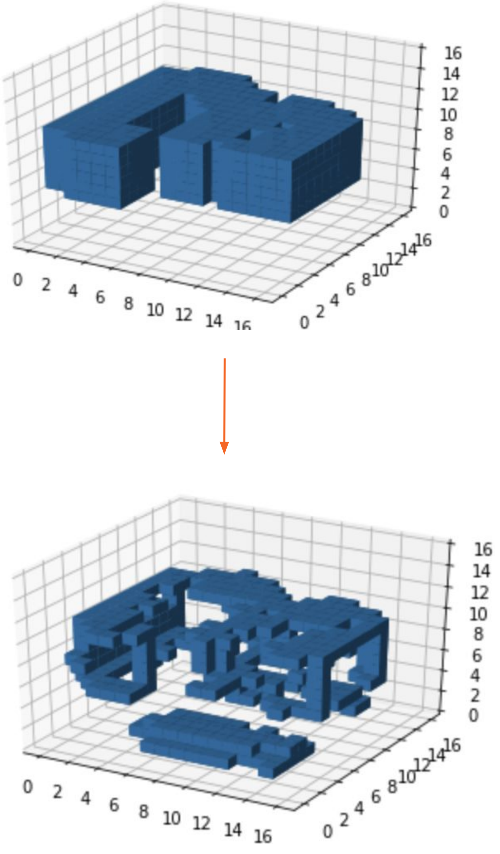
ResNet



Experiments

➔ **Uncolored Kaggle 3D MNIST dataset translation**

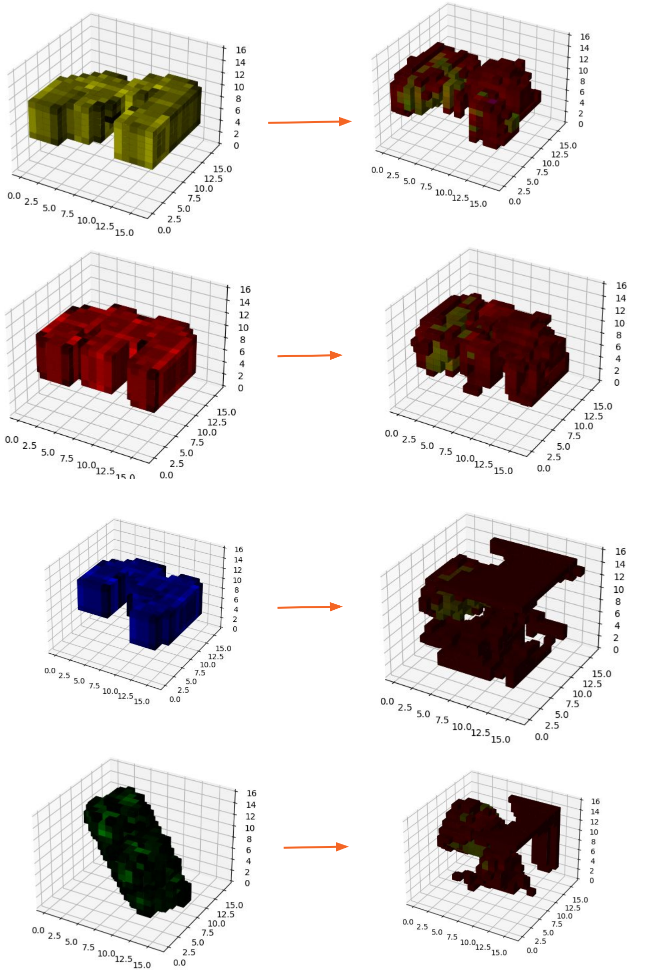
The model could not learn the transport map in any reasonable amount of iterations.



Experiments

➔ **Colored Kaggle 3D MNIST dataset translation**

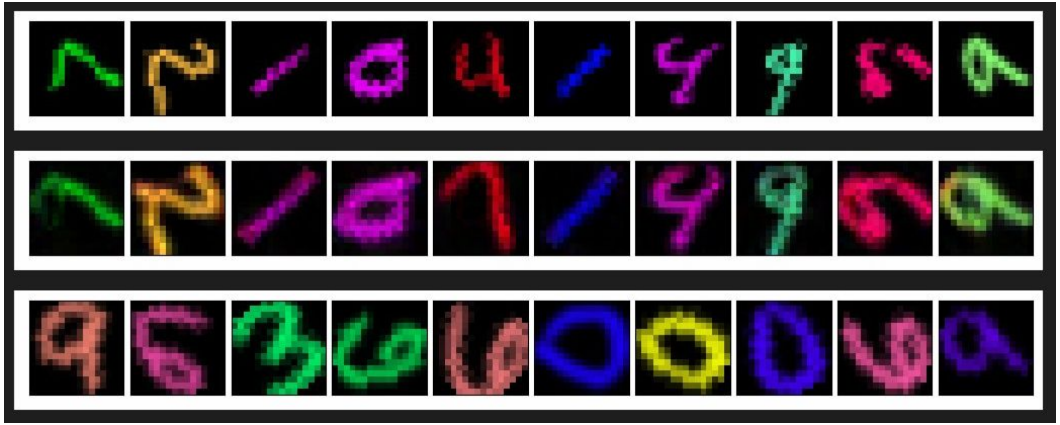
The model could not learn the transport map in any reasonable amount of iterations due to small training set size, random rotation and poor binarization of the dataset.



Experiments

➔ **2D MNIST with random rotations**

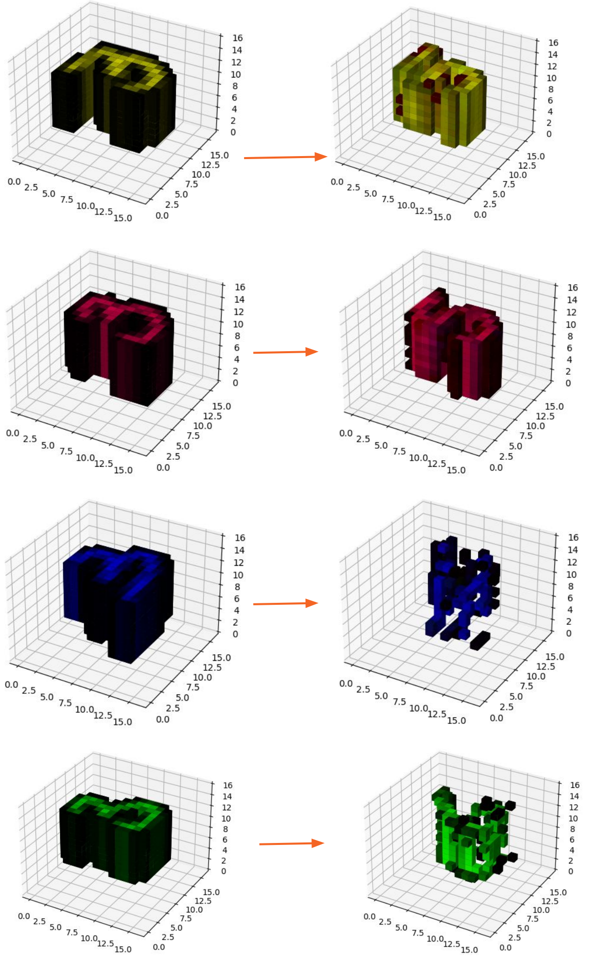
We've taken the code from Alex Korotin's seminar on Neural Optimal Transport and added random translations in range of 45 degrees. The model couldn't learn the mapping in any reasonable amount of iterations.



Experiments

➔ Our 3D MNIST without rotation

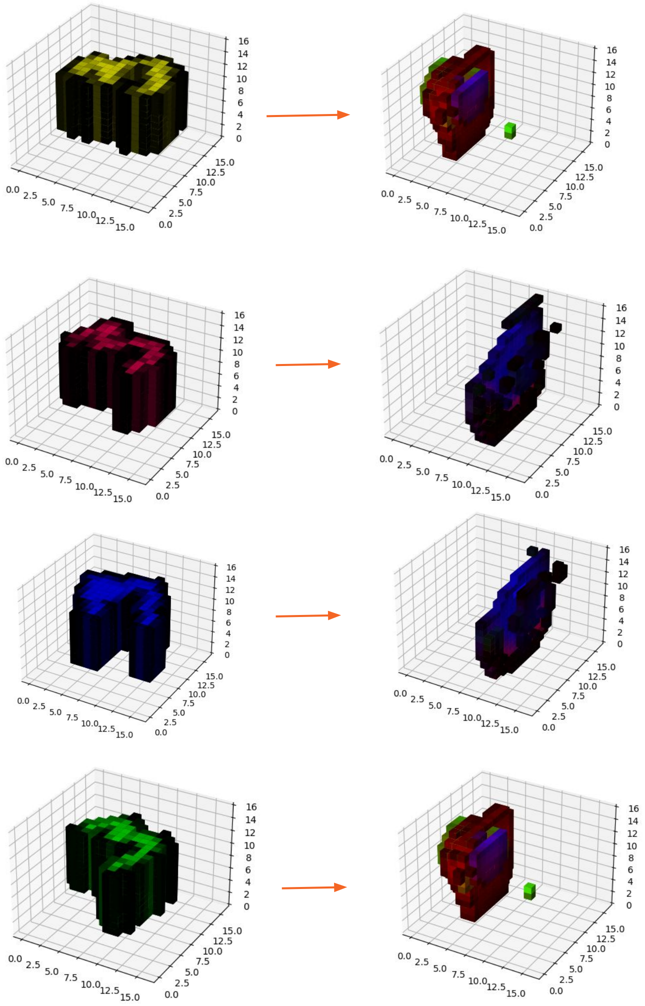
The model semi-successfully learned the mapping between threes and fives in red and yellow colors. In blue and green the mapping could not be learned, but the colors still remain.



Experiments

➔ **Our 3D MNIST with rotation**

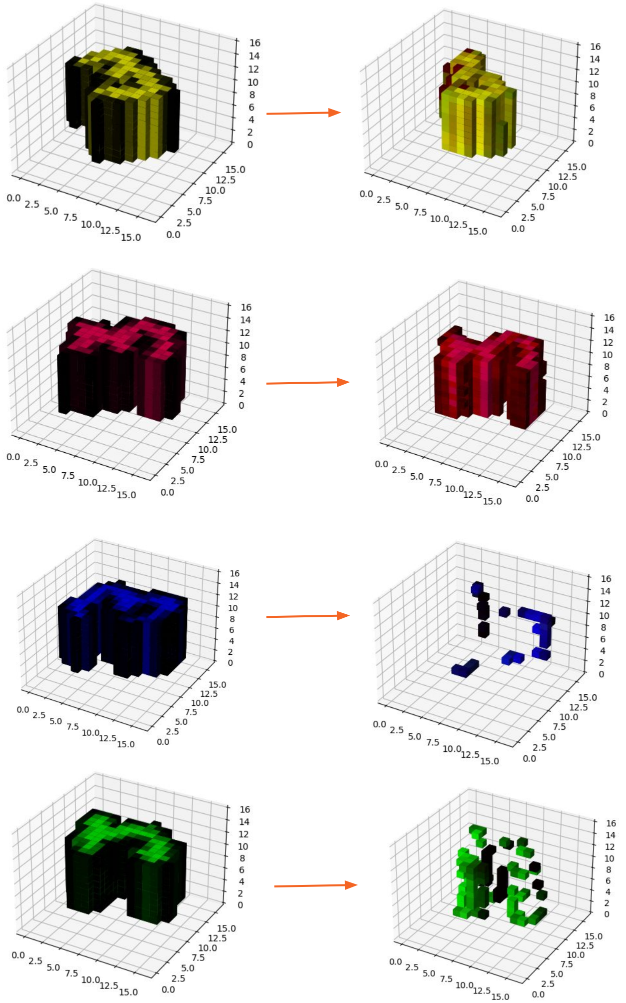
The model failed to learn the transport map in any reasonable amount of iterations.



Experiments

➔ **Transfer learning on our 3D MNIST with rotation**

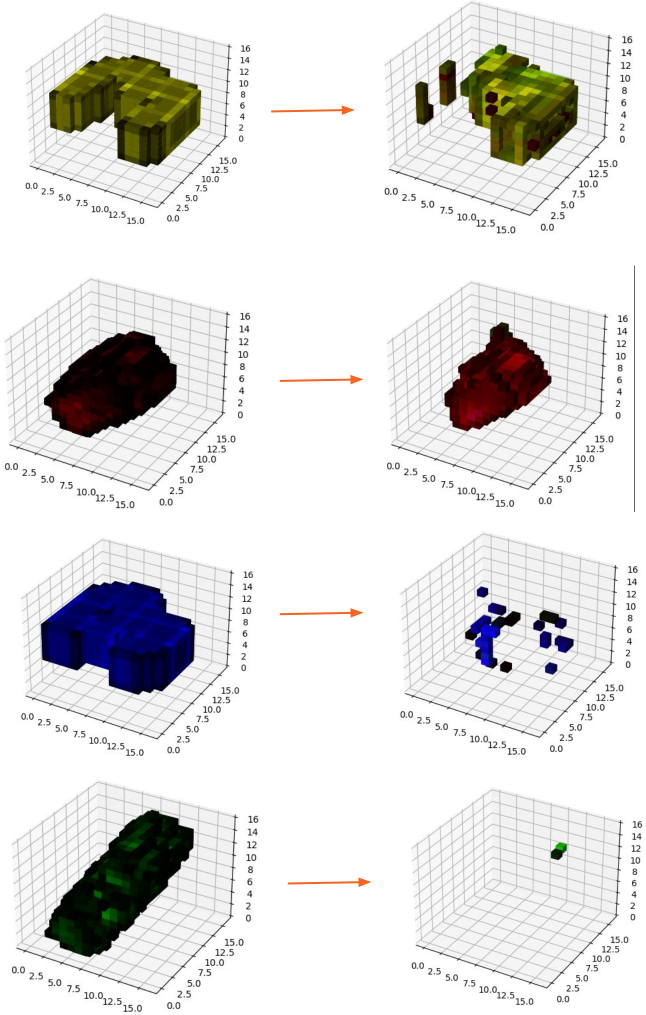
We've ran the training for 30000 iterations on unrotated dataset and used the resulting weights as a starting point to train a new model on a rotated dataset. Again, the model semi-correctly learned the mapping between yellow threes and yellow fives and red threes and fives. Other colors still look bad.



Experiments

➔ **Transfer learning from unrotated data on Kaggle 3D MNIST**

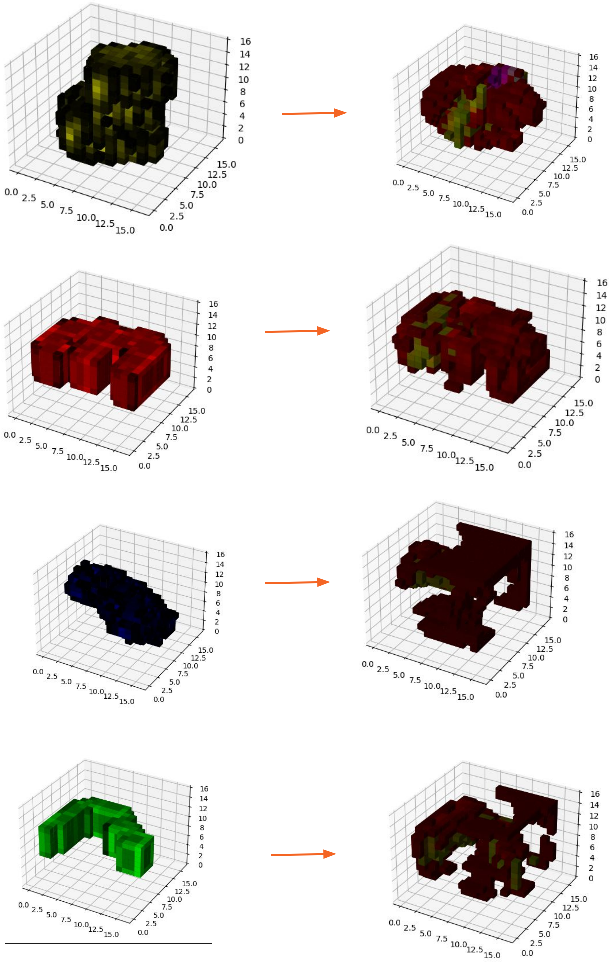
We've used the same weights from the 30000 iteration run to fit the model on Kaggle 3D MNIST data. The model diverged in the first ten iterations.



Experiments

➔ **Transfer learning from rotated data on Kaggle 3D MNIST**

We've used the weights from the 30000 iteration run to fit the model on our rotated 3D MNIST data and then used these weights to fit the model on Kaggle 3D mnist data. The model diverges in the first couple of steps.



Final thoughts

➔ Models are hard to train and are prone to **mode collapse**

➔ Models are sensitive to the **quality of the data**

➔ Adding extra features to the data will make training **a lot** harder

Thx!