Università degli Studi di Padova

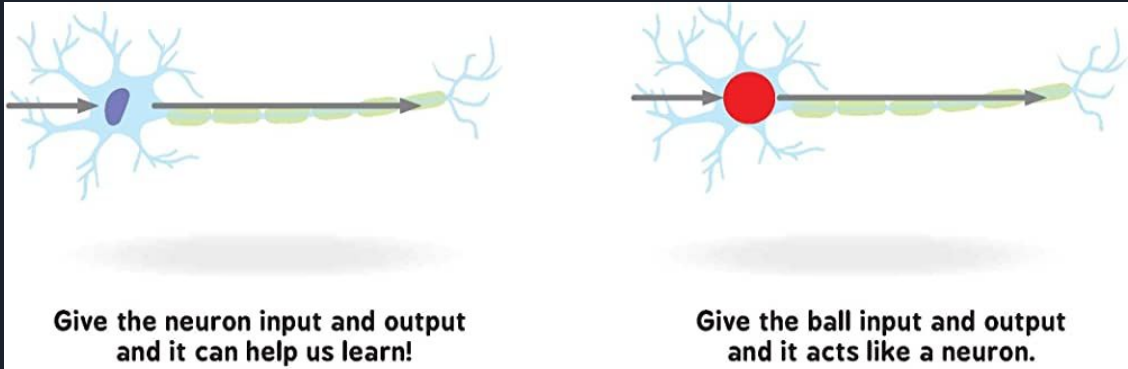
On the Information Bottleneck Theory of Deep Learning

Project for the course : Information theory
Università degli Studi di Padova A.Y 2022/2023

Ausilio Lorenzo
Höth Max Henning
Martemucci Walter
Prodan George

Presentation outline

1. Information Bottleneck theory of Deep Learning
2. Saxe IB paper
3. Case study : Convolutional Neural Networks
   a. SimpleCNN
   b. ResNet
4. Data : Mel-spectrogram representation of audio
5. Computation of Mutual Information
   a. Binning
   b. KDE
   c. Kraskov
6. Results
7. Conclusions



Information Bottleneck Theory of Deep Learning

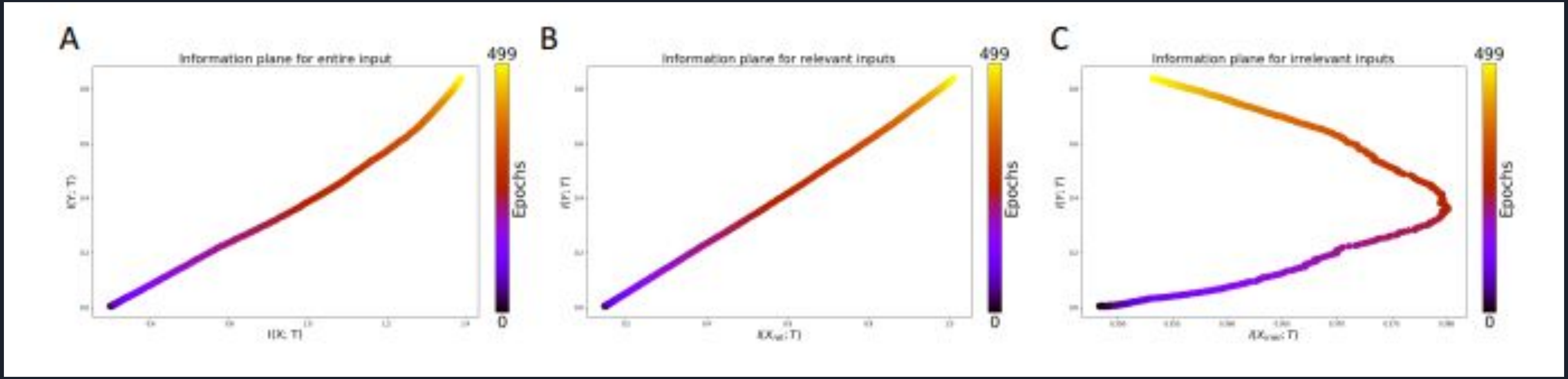
- Practical successes of deep neural networks have not been matched by theoretical progress that satisfyingly explains their behavior

- Information theoretic tools to explain generalization capabilities of Neural Networks

- Makes three specific claims
    - Deep Neural networks undergo two phases : an *initial fitting phase* and a *subsequent compression phase*
    - Compression is causally linked to the generalization capabilities of Neural Networks
    - Compression phase occurs thanks to the stochasticity of Stochastic Gradient Descent

Saxe IB Paper

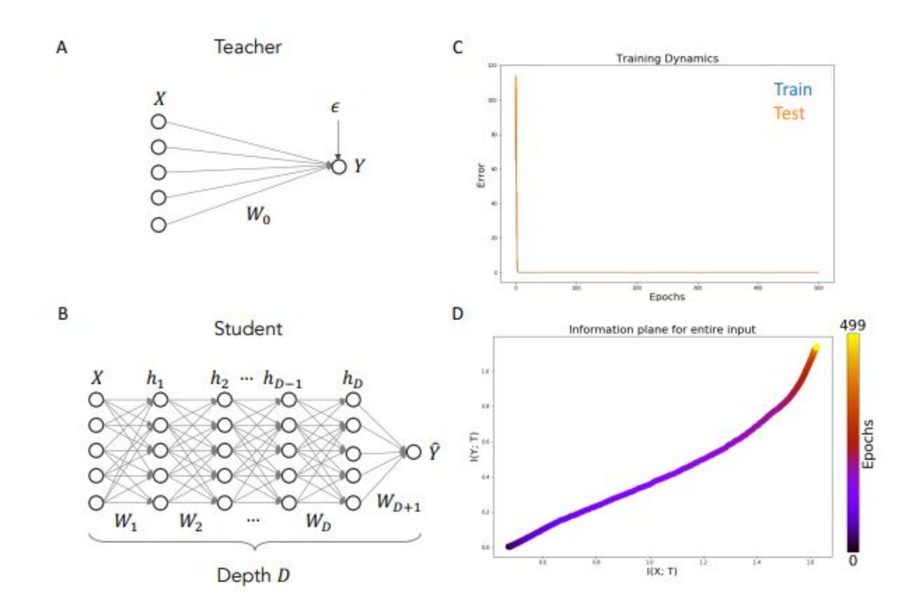
- First claim: *"Deep Neural networks undergo two phases : an initial fitting phase and a subsequent compression phase"*
- Fitting and compression happen at the same time :
    - compression of the task irrelevant
    - fitting of the task relevant
    - overall Mutual Information may still increase

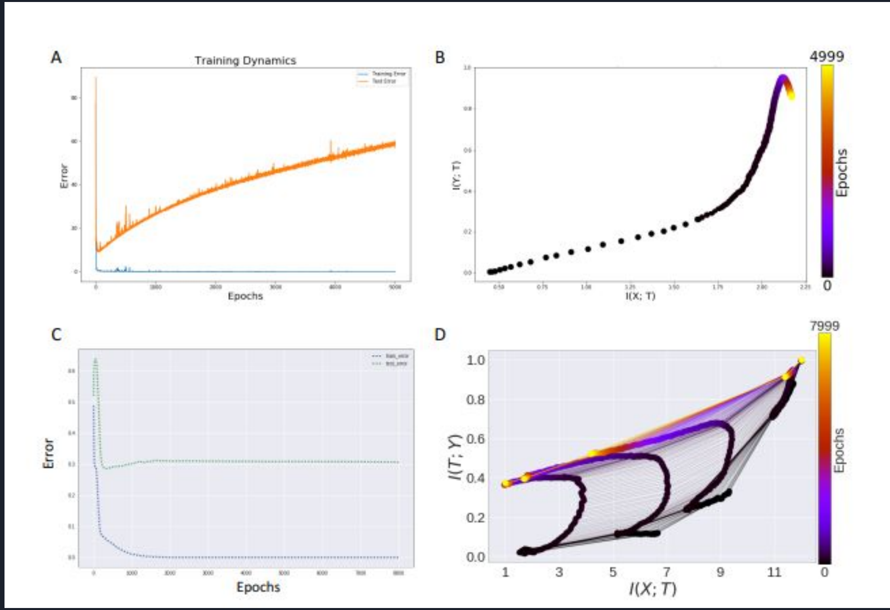


Saxe IB Paper

- Second claim : *"Compression is causally linked to the generalization capabilities of Neural Networks"*

- Networks that generalize well may/may not compress and vice versa





Saxe IB Paper

- Third Claim : *“Compression phase occurs thanks to the stochasticity of Stochastic Gradient Descent”*

- Full batch GD also shows a compression phase, likely due to the non-linearities
    - single-saturating nonlinearities like *ReLU* don’t lead to compression
    - double-saturating nonlinearities like *tanh* lead to compression

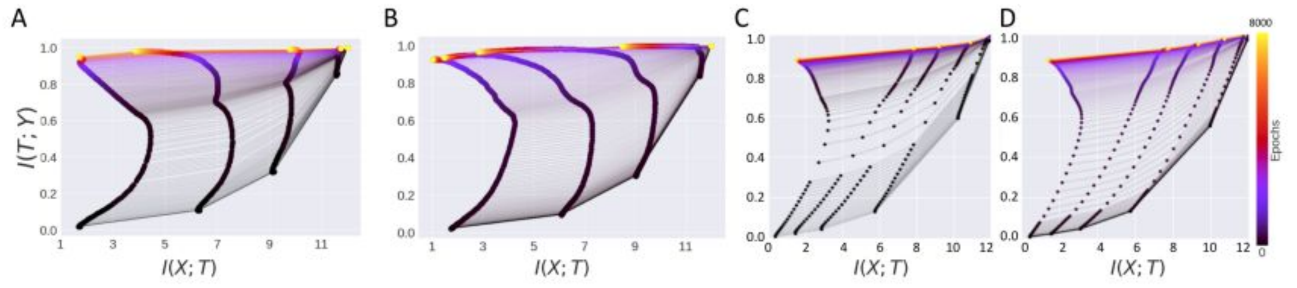


Figure 5: Stochastic training and the information plane. (A) tanh network trained with SGD. (B) tanh network trained with BGD. (C) ReLU network trained with SGD. (D) ReLU network trained with BGD. Both random and non-random training procedures show similar information plane dynamics.

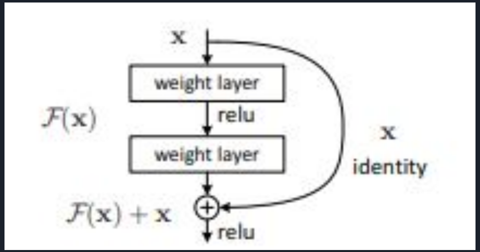
Case study : Convolutional Neural Networks

Networks used : SimpleCNN and ResNet

Inputs: 2D spectrogram

- **SimpleCNN** :
  - 4 layers (Convolution + ReLU/tanh + BN)
  - from 8 to 32 filters
  - the output block is composed by an Adaptive Pooling layer and a dense layer

```
SimpleCNN
├─Sequential: 1-1
│    └─Conv2d: 2-1
│    └─ReLU: 2-2
│    └─BatchNorm2d: 2-3
│    └─Conv2d: 2-4
│    └─ReLU: 2-5
│    └─BatchNorm2d: 2-6
│    └─Conv2d: 2-7
│    └─ReLU: 2-8
│    └─BatchNorm2d: 2-9
│    └─Conv2d: 2-10
│    └─ReLU: 2-11
│    └─BatchNorm2d: 2-12
├─AdaptiveAvgPool2d: 1-2
├─Linear: 1-3
```



- **ResNet:**
  - proposed for the first time by Zhang et al in 2015 for image recognition
  - we use a mini ResNet with 1 Conv and 2 Residual Layers, followed by a MaxPooling layer

Data : Mel-spectrogram representation of audio

Free Music Archive (FMA)

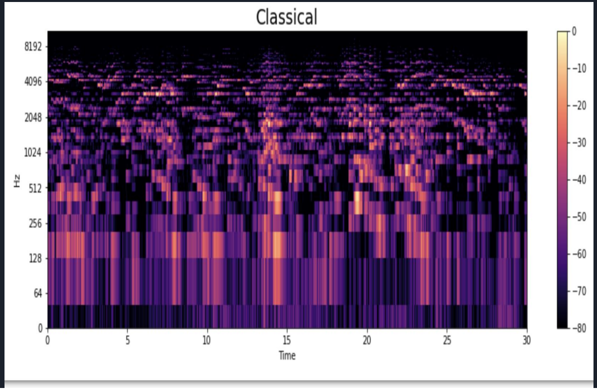
8 genres of music

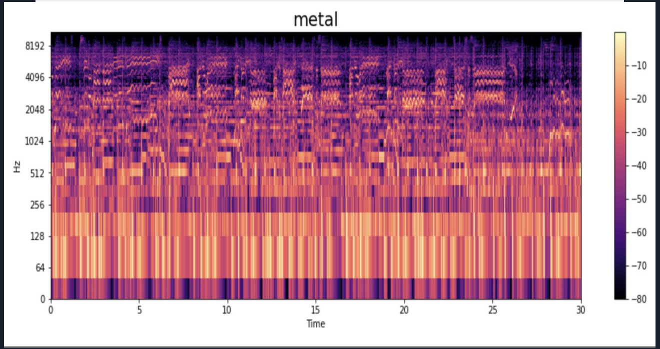
FMA-small: 8000 clips of 30 seconds

Split the dataset into 80/10/10

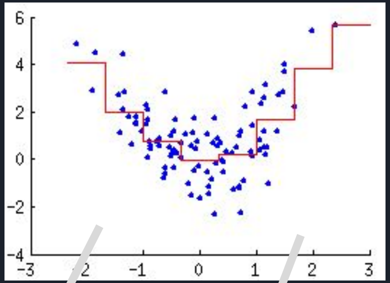
Stratified split

Class distribution in the three sets is representative of the class distribution in the whole dataset





The Binning Method



The Mutual Information between the INPUT distribution and the output T given by one LAYER:

$$I(T,X) = H(T) - \cancel{H(T|X)} = -\sum_{i=1}^{N} p_i \log p_i$$

perform binning for each class

The Mutual Information between the OUTPUT distribution and the output T given by one LAYER:

$$I(T,Y) = H(T) - H(T|Y) = H(T) + \sum_{i=1}^{n} \sum_{j=1}^{N_j} p_{ij} \log p_{ij}$$

KDE approach (Kolchinsky et al.)

Estimates the mutual information between input and the layer activity by assuming that the activity is distributed as a mixture of Gaussians.

$$T = h + \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, \sigma^2 I)$$

The noise is added solely for the purposes of analysis, and is not present during training or testing the network.

Noise variance = 0.01.

Necessity of noise assumptions

The activity of a neural network is often a continuous deterministic function of its input. That is, in response to an input X, a specific hidden layer might produce activity h = f(X) for some function f.

The mutual information between h and X is given by:

$$I(h; X) = H(h) - H(h|X)$$

h is typically continuous and continuous entropy, defined for a continuous random variable is

$$H(Z) = -\int p_Z(z) \log p_Z(z) dz$$

can be negative and possibly infinite, in particular, note that if p(z) is a delta function, then H(Z) = -inf.

To yield a finite mutual information, some noise in the mapping is required such that H(h|X) remains finite.
A common choice (and one adopted here for the linear network, the nonparametric kernel density estimator) is to analyze a new variable with additive noise, T = h + Z, where Z is a random variable independent of X. Then H(T|X) = H(Z) which allows the overall information I(T;X) = H(T) - H(Z) to remain finite.

Computation of MI : KDE method

An upper bound for the mutual information with the input is:

$$I(T;X) \leq -\frac{1}{P}\sum_i \log \frac{1}{P}\sum_j \exp\left(-\frac{1}{2}\frac{\|h_i - h_j\|_2^2}{\sigma^2}\right)$$

P is the number of training samples and hi denotes the hidden activity vector in response to input sample i.

Mutual information with respect to the output:

where L is the number of output labels,

Pl number of data samples with output label l,

$$\begin{aligned} I(T;Y) &= H(T) - H(T|Y) \\ &\leq -\frac{1}{P}\sum_i \log \frac{1}{P}\sum_j \exp\left(-\frac{1}{2}\frac{\|h_i - h_j\|_2^2}{\sigma^2}\right) \\ &\quad -\sum_l^L p_l \left[-\frac{1}{P_l}\sum_{i,Y_i=l} \log \frac{1}{P_l}\sum_{j,Y_j=l} \exp\left(-\frac{1}{2}\frac{\|h_i - h_j\|_2^2}{\sigma^2}\right)\right] \end{aligned}$$

Lower bound:

$$
\begin{aligned}
I(T;Y) \;\geq\; & -\frac{1}{P}\sum_i \log \frac{1}{P}\sum_j \exp\left(-\frac{1}{2}\frac{\|h_i - h_j\|_2^2}{4\sigma^2}\right) \\
& -\sum_l^L p_l \left[-\frac{1}{P_l}\sum_{i,Y_i=l} \log \frac{1}{P_l}\sum_{j,Y_j=l} \exp\left(-\frac{1}{2}\frac{\|h_i - h_j\|_2^2}{4\sigma^2}\right)\right]
\end{aligned}
$$

Computation of MI : Kraskov method

- I(X;T), the MI between X = inputs and T = activations, is approximated as H(T), the entropy of T. $\boxed{\textit{Compression = decreasing H(T) over time}}$

- $$\begin{aligned} I(T;X) &= H(T) - H(T|X) \\ &= H(T) - H(Z) \\ &= H(T) - c \end{aligned}$$
  where c is an unknown constant

- But H(T) has to be computed using the Kraskov estimator :

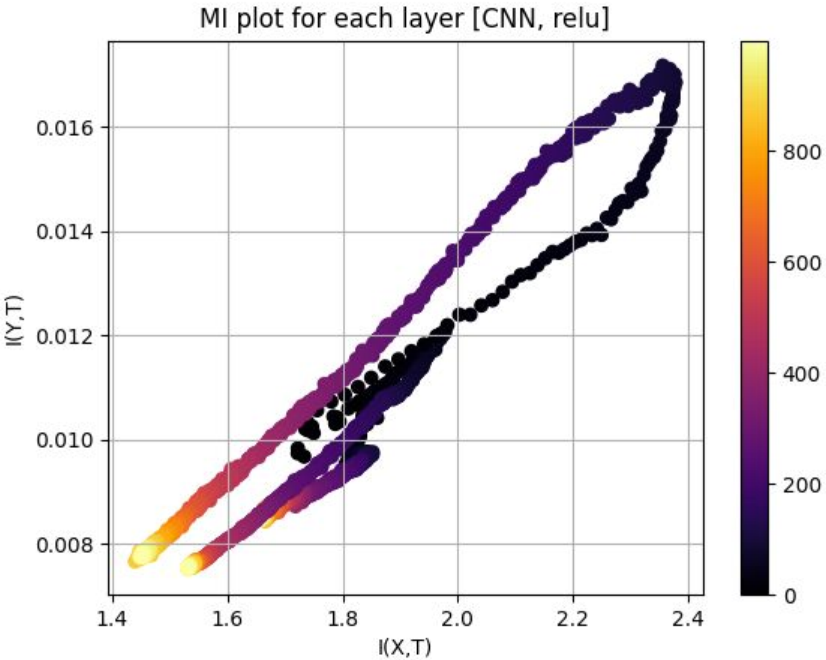
$$\frac{d}{P}\left(\sum_i \log(r_i + \epsilon)\right) + \frac{d}{2}\log(\pi) - \log\Gamma\left(\frac{d}{2} + 1\right) + \psi(P) - \psi(k)$$

where d = dimension of T, P = number of samples, r_i = distance to k-th nearest neighbor,

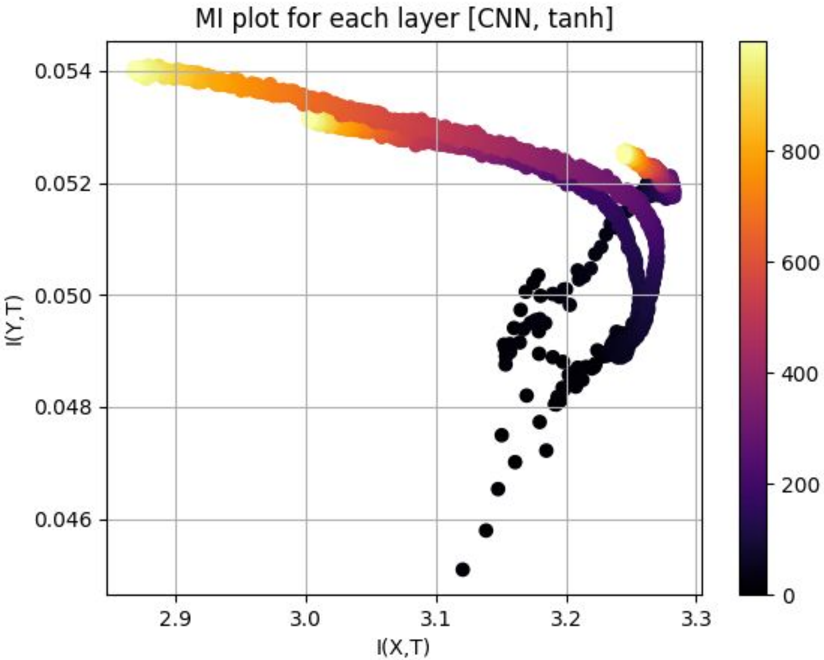
$\epsilon = 10^{-16}$ is added for numerical stability

RESULTS - Binning Method

ReLU



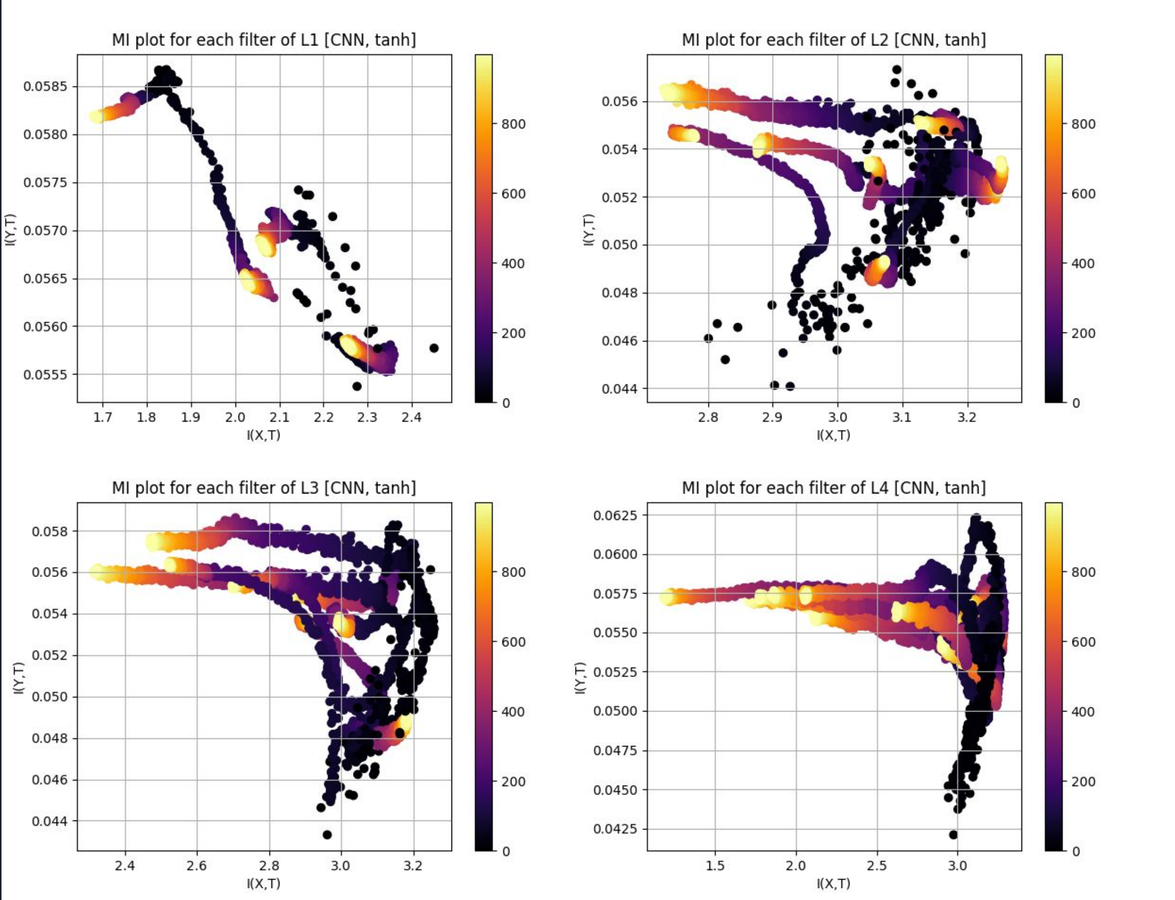
tanh



RESULTS

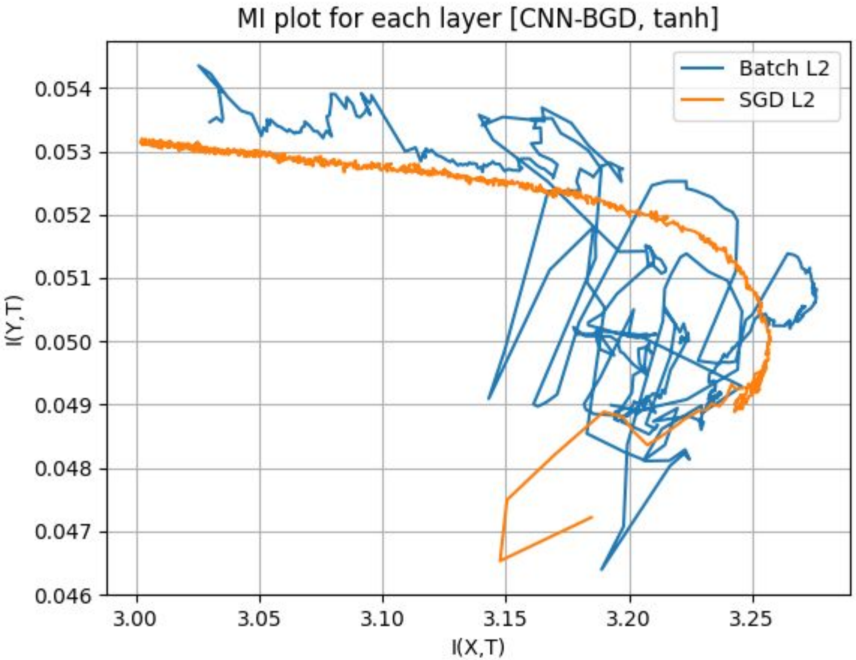
Binning Method

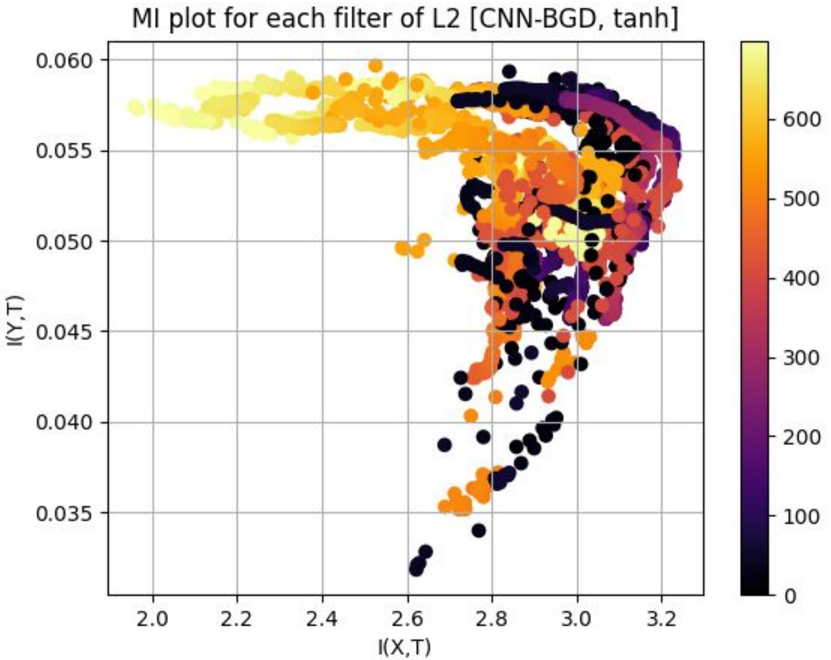
MI computed for each individual filter



RESULTS - Binning Method

Stochastic GD / Batch GD

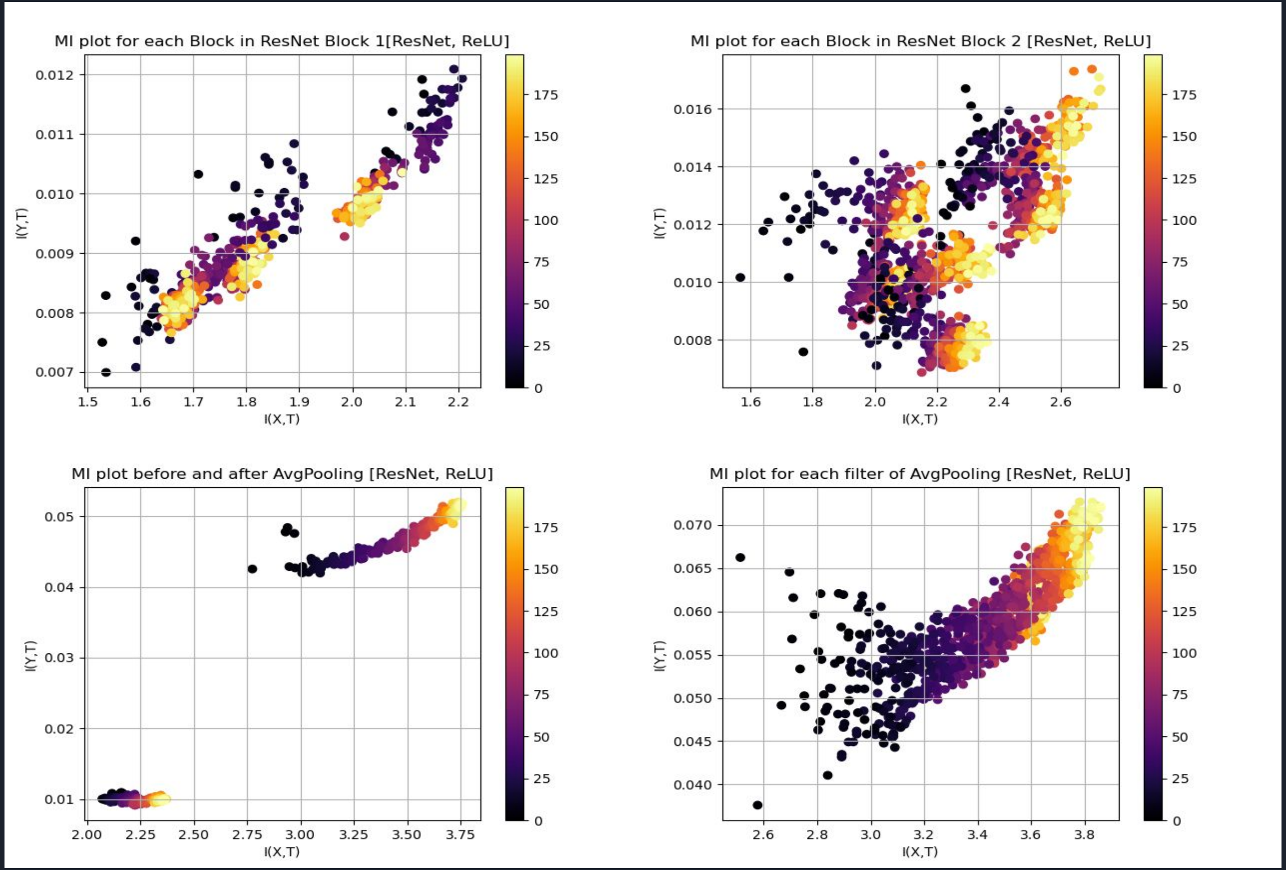




RESULTS - Binning Method

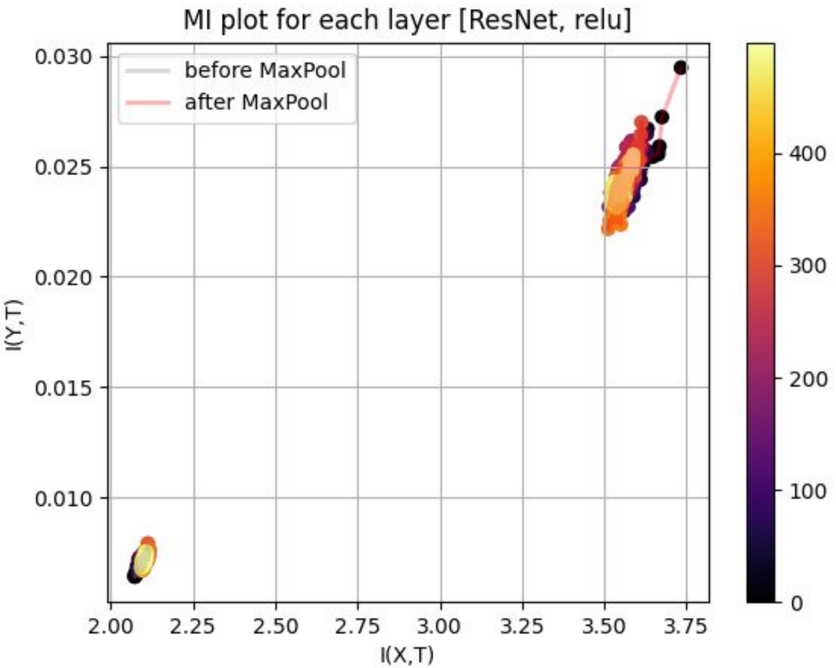
MI computed for each individual Block in ResNet

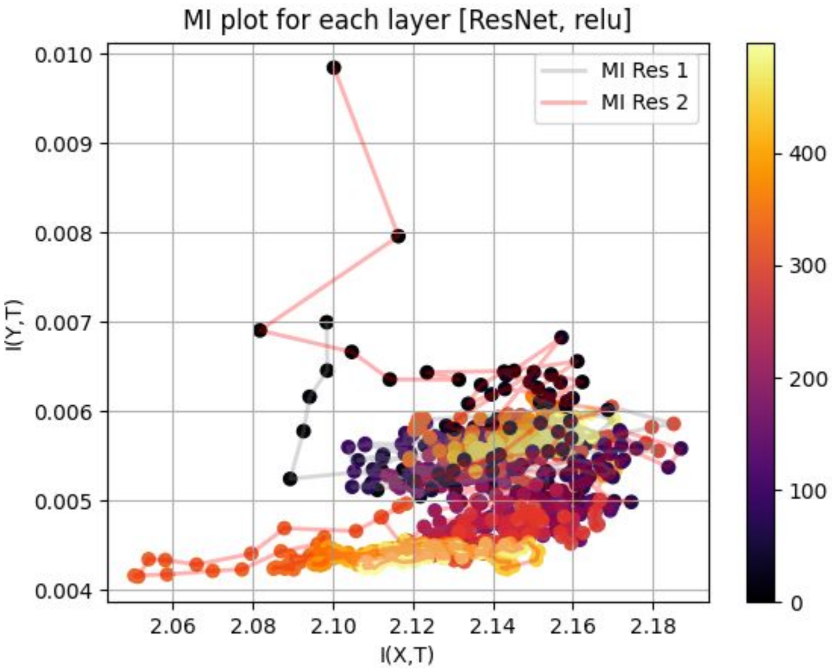
Comparison before and after AvgPooling



RESULTS - Binning Method

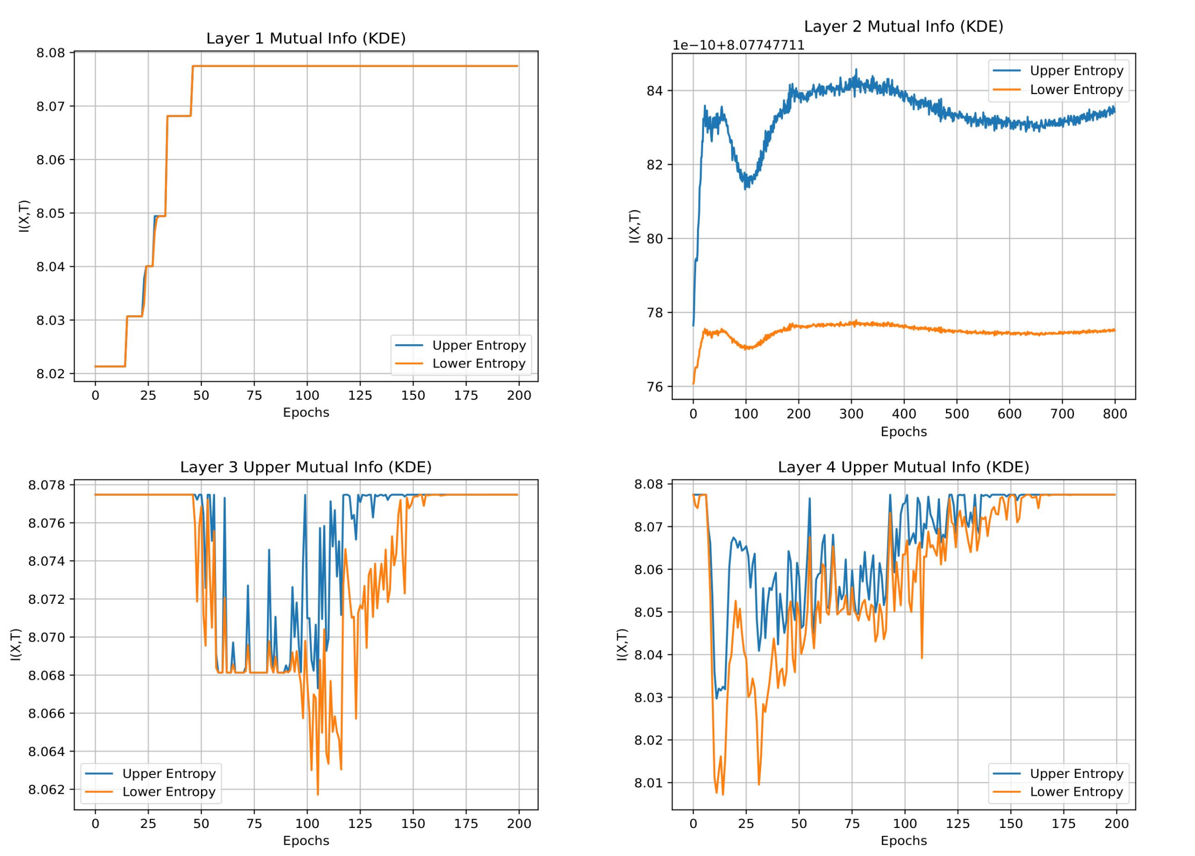
MI computed for each full layer (ResNet/ReLU)





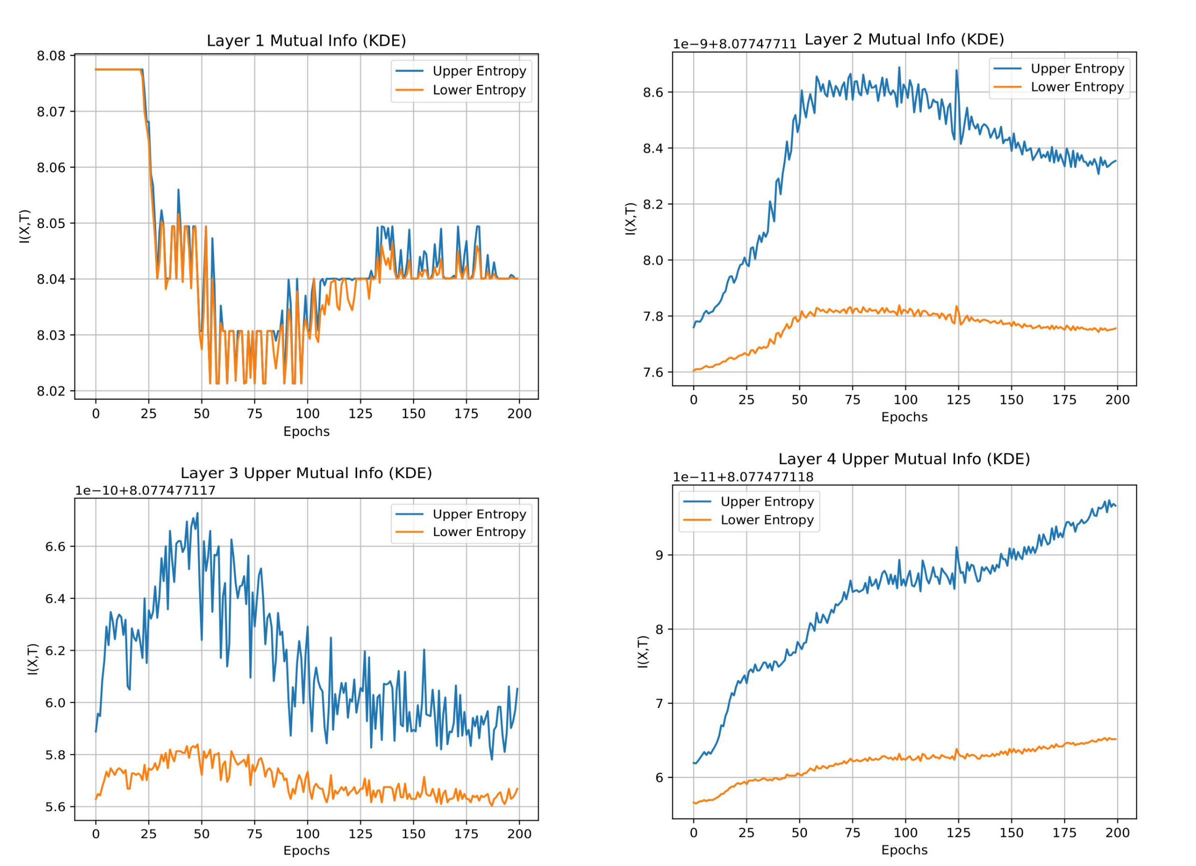
Results : KDE method - ReLU

MI computed for each Layer over the Epochs



Results : KDE method - Tanh

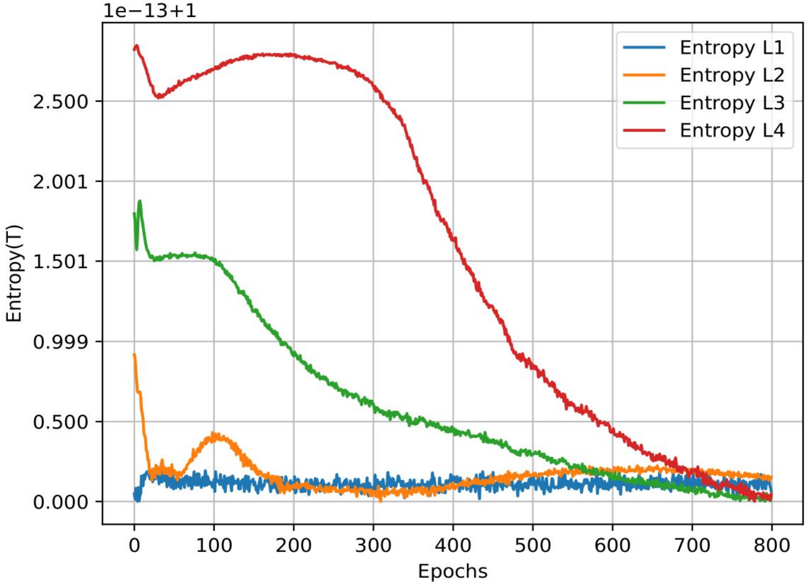
MI computed for each Layer over the Epochs

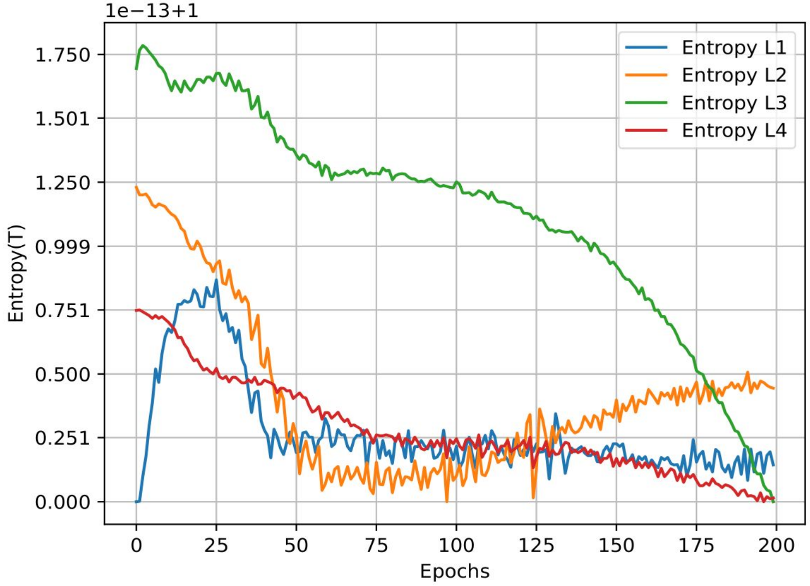


RESULTS - KDE Method

ReLU

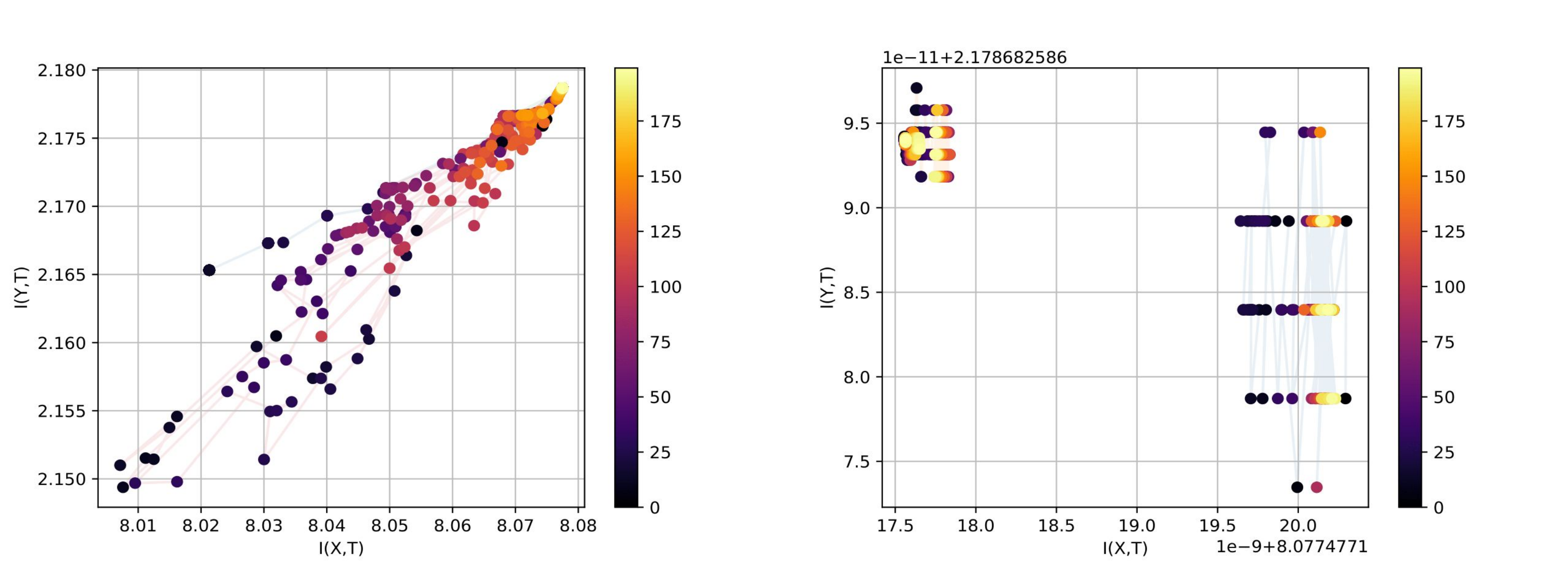
Tanh





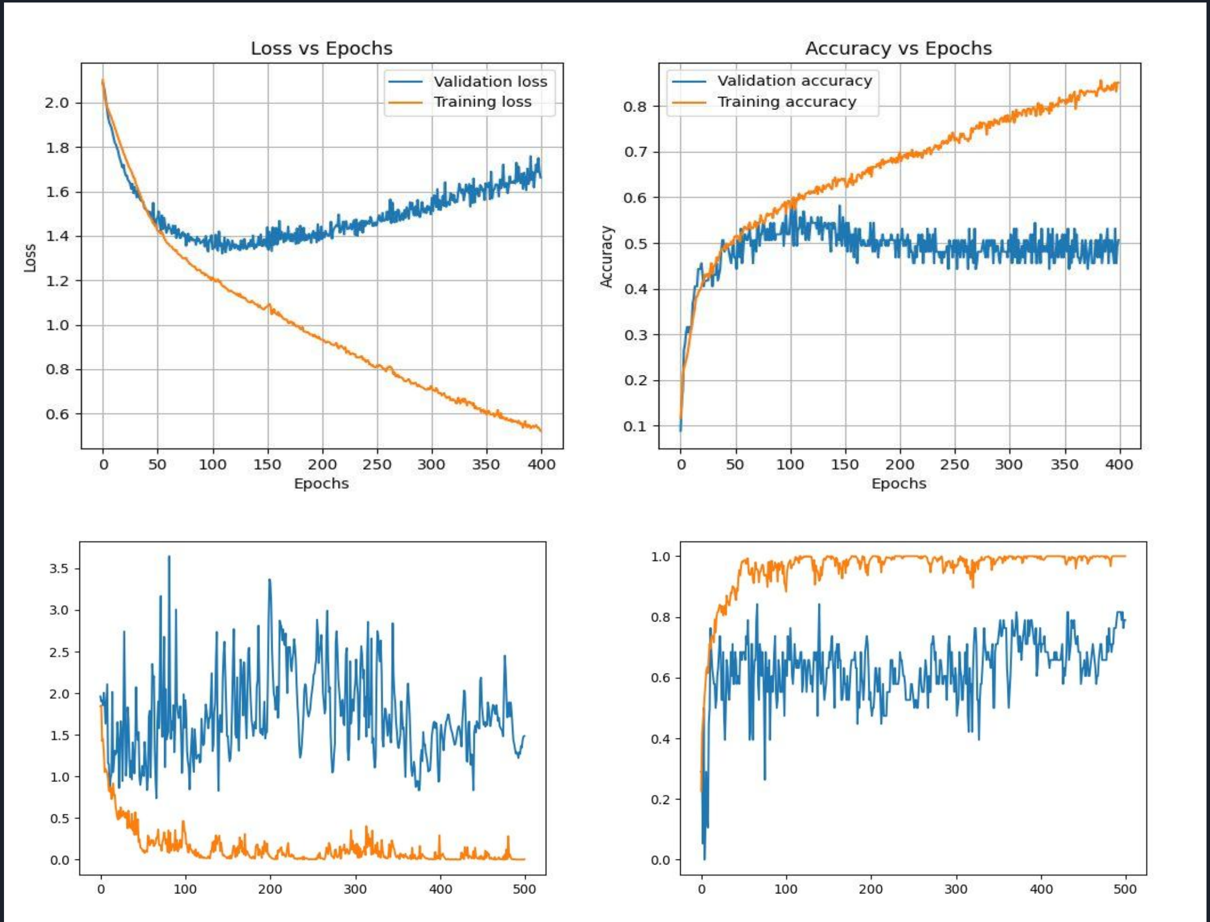
RESULTS - KDE Method

ReLU | Tanh



RESULTS training

ReLU

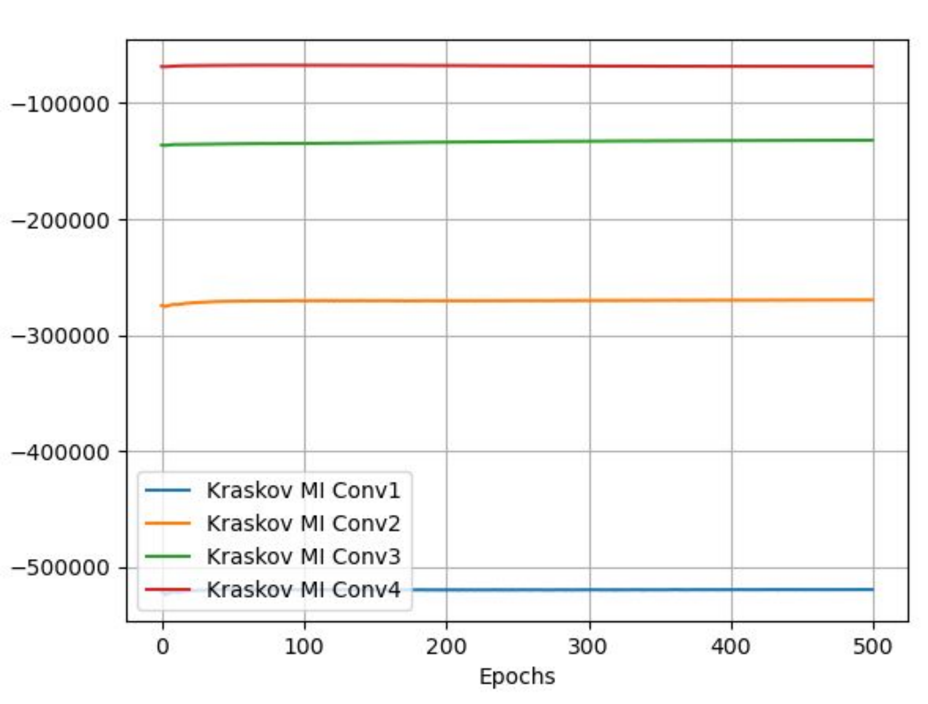


Tanh

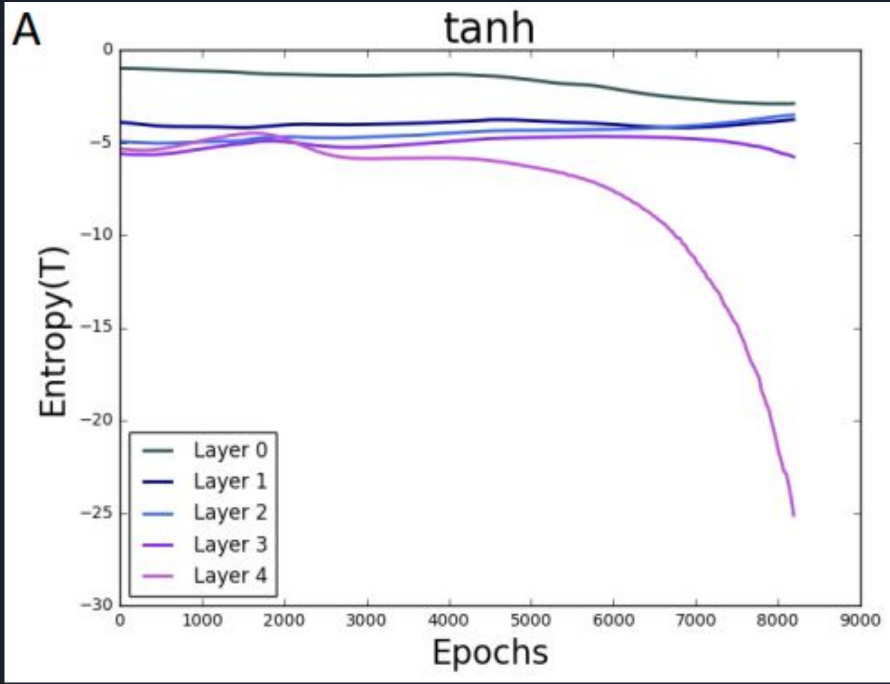
RESULTS - Kraskov Method : Tanh, SimpleCNN

For all tests : K = 2

Our Results



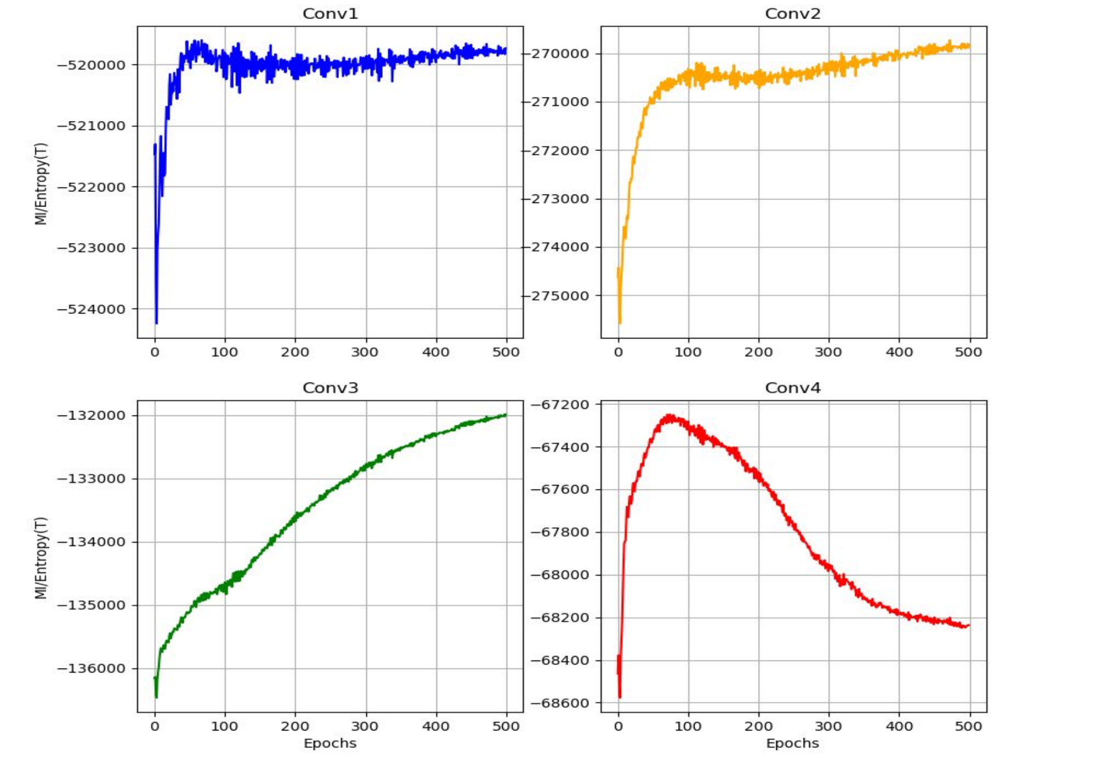
Saxe Paper



RESULTS - Kraskov Method : Tanh, SimpleCNN

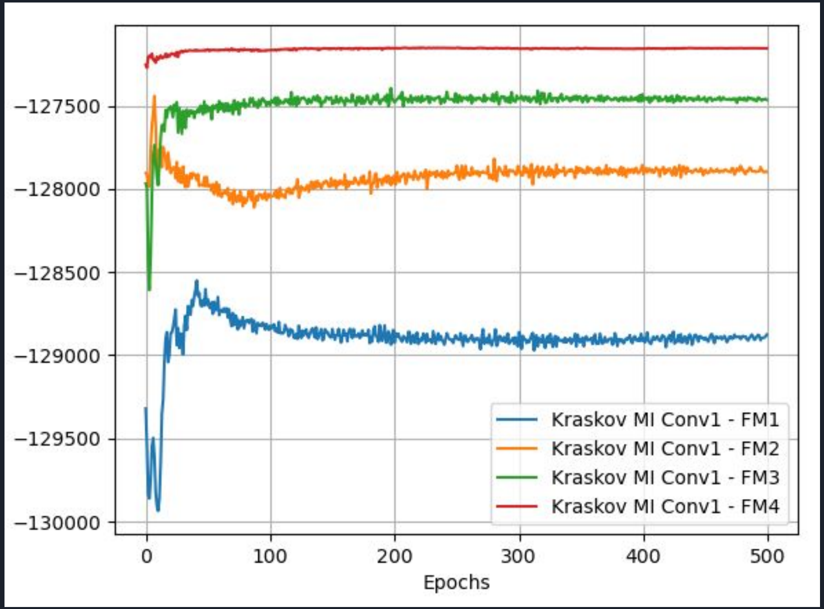
- When we plot the layers separately
- Consistent with the finding of Saxe

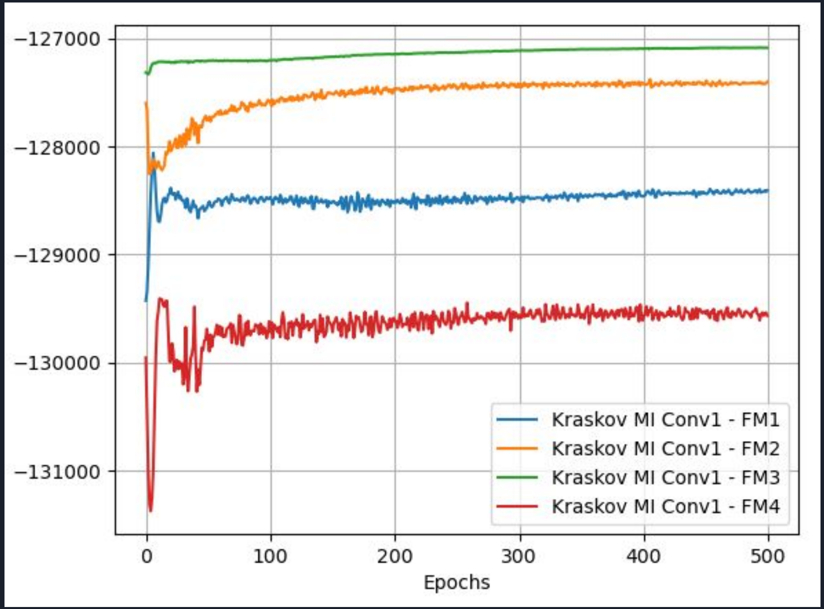
- Apparent discrepancy is caused by number of feature maps in CNN (4,8,16,32) vs NN layers (10,7,5,4,3) in Saxe



RESULTS - Kraskov Method : Tanh, SimpleCNN

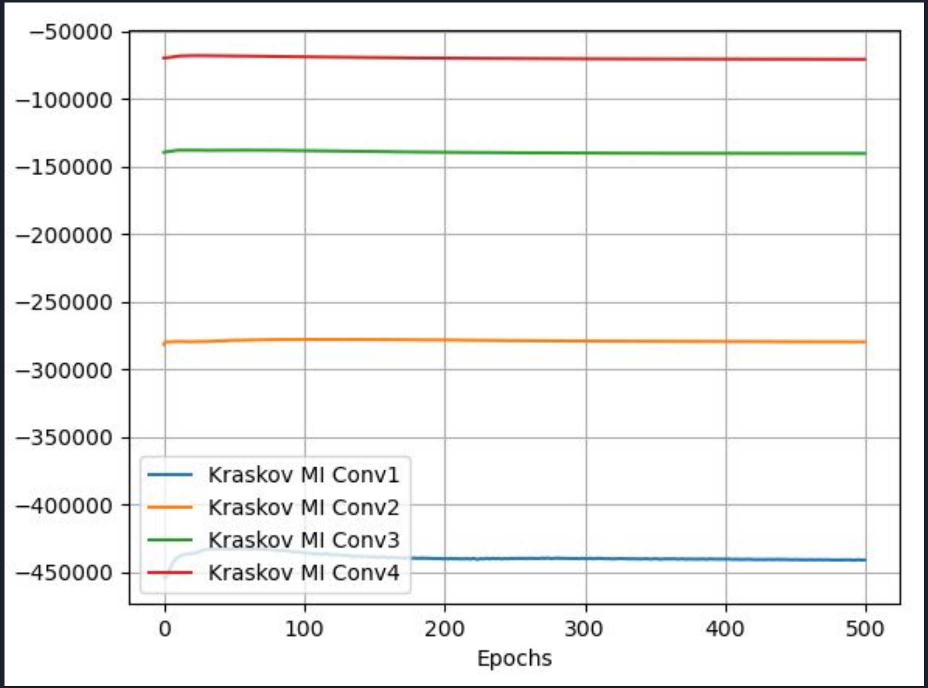
- Further tests : *Feature maps*
- How do feature maps contribute to information compression ?
- Tests for First layer : 4 feature maps



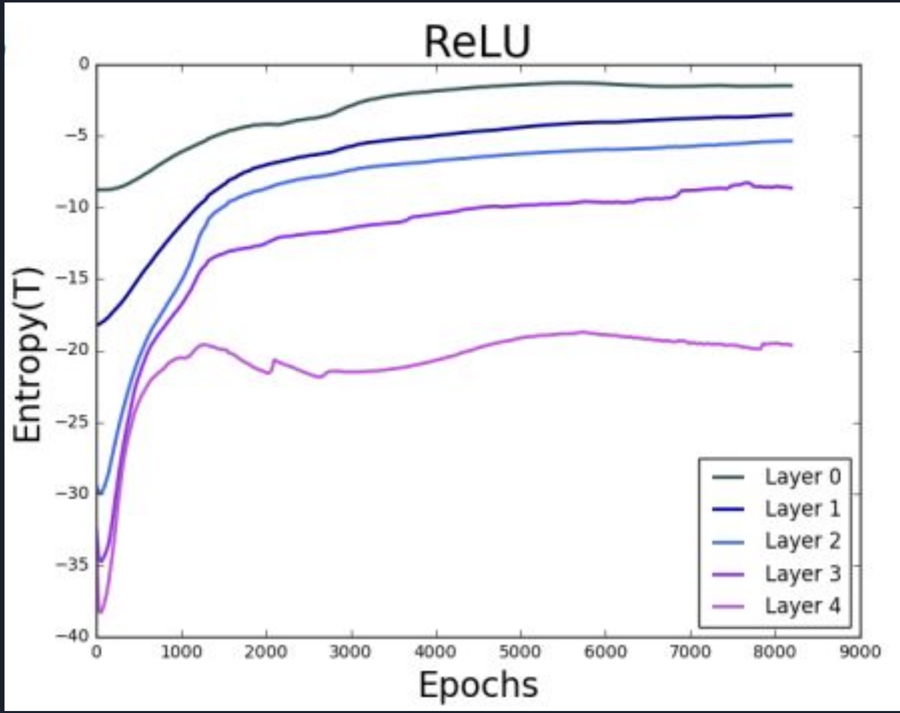


RESULTS - Kraskov Method : ReLU, SimpleCNN

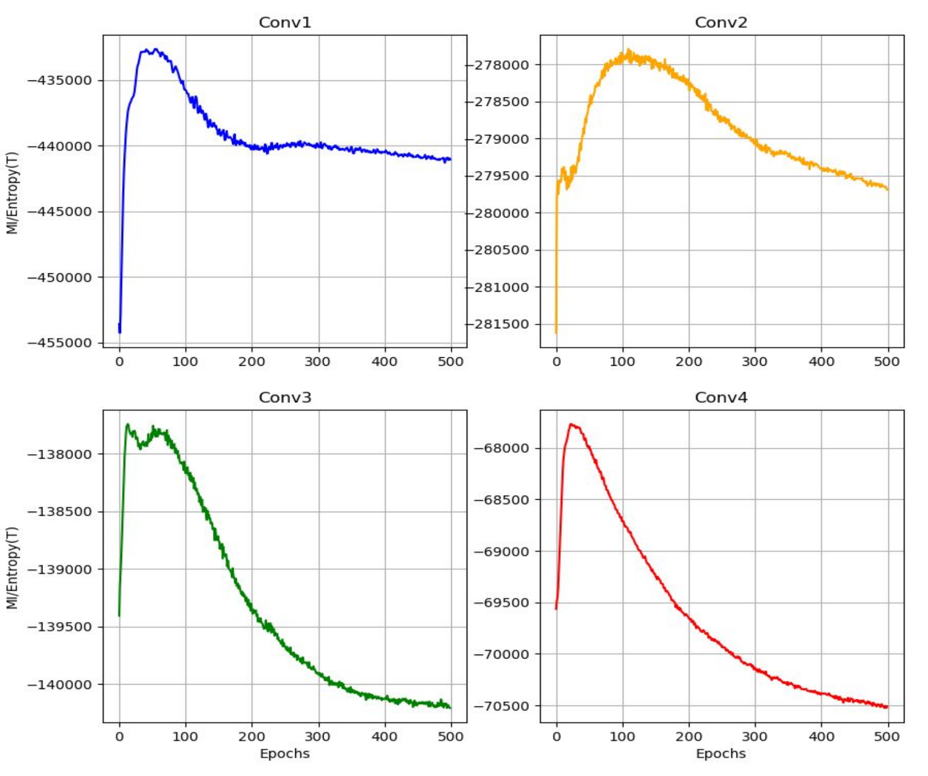
Our Results



Saxe Paper

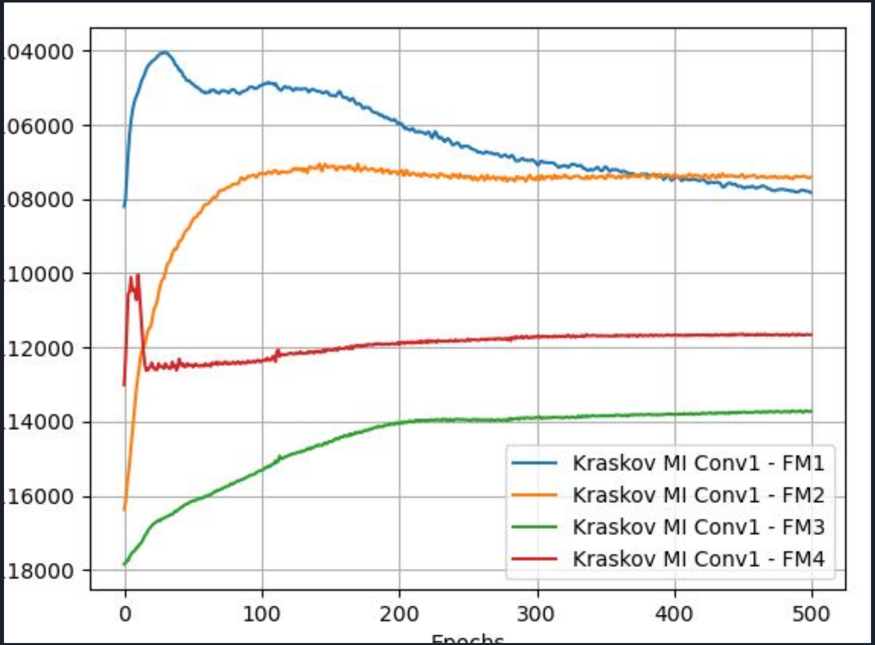


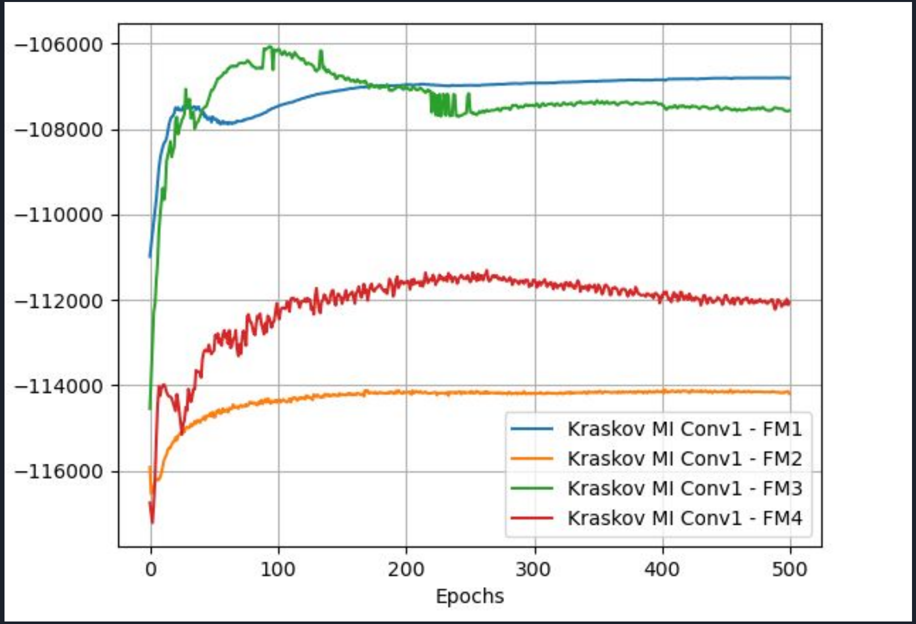
RESULTS - Kraskov Method : ReLU, SimpleCNN



RESULTS - Kraskov Method : ReLU, SimpleCNN

- Feature maps test





Conclusions

- We find similar results as those found by the Saxe paper
- Compression is present in the learning process of simple CNNs
- Pooling, however had a big influence on the MI
- The choice of ReLU/Tanh is reflected on the information plane