Unsupervised Training using a Temporal Auto-Encoder Framework

Ross Goroshin Joan Bruna Arthur Szlam Yann LeCun

April 30, 2014

NEW YORK UNIVERSITY

Unsupervised Training using Video Data

- How can we train on the massive amounts of unlabeled video data available?
- Video is temporally coherent, thus it is reasonable to assume that neighboring frames are semantically similar
- Use temporal coherence to develop new unsupervised learning objectives and train architectures (CNNs) that are able to satisfy them
- Video data can be considered as a form of 'correct' data augmentation, i.e. the variations observed are the variations on which to learn invariance/equivariance

Slowness

- Extract features from individual frames that vary slowly with time, i.e. if $z_i = G_w(x_i)$ then $min \; \|z_{i+1} - z_i\|_p$
- Slow feature analysis:

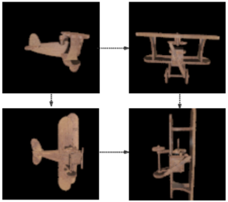
$$\text{Let } y_j(t) := g_j(x(t))$$
$$min \; \Delta(y_j) := \left\langle \dot{y}^2 \right\rangle_t$$
$$s.t. \; \left\langle y_j \right\rangle_t = 1 \; and \; \forall i < j : \left\langle y_i, y_j \right\rangle_t = 0$$

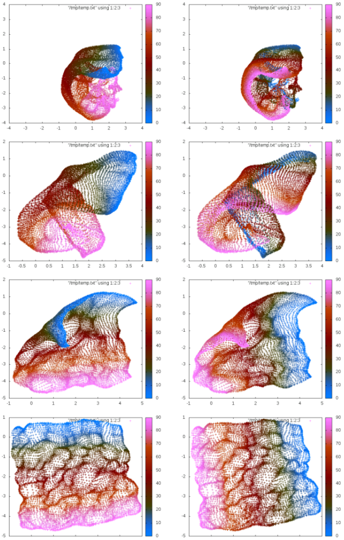
- DrLIM:

$$\text{Let } D_w(X_1, X_2) = \|G_w(X_{i+1}) - G_w(X_i)\|_2$$
$$L = (1 - Y)\frac{1}{2}D_W^2 + Y\frac{1}{2}(max(0, m - D_W))^2$$

Slowness as Metric-Learning

- 2-dimensional manifold living in a $\approx 10,000$-dimensional space (96x96 images)
- Similarity relationships can be naturally assigned via adjacent frames in a video

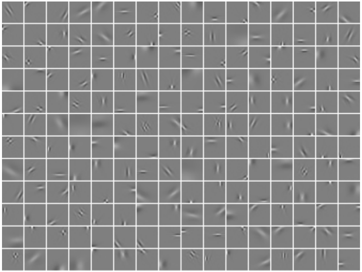




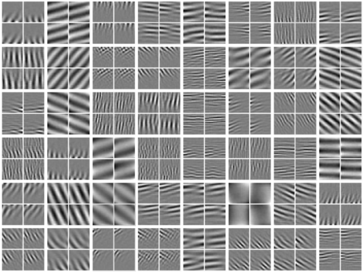
Fully Connected Slow-Feature Auto-Encoders

Replacing the contrastive term in DrLIM with reconstruction lead to the slow-feature auto-encoder:

$$L_{sample} = \sum_{i=1}^{2} \frac{1}{2} \| x_i - W_d \; z_i \|^2 + \alpha | z_2 - z_1 |$$



No Pooling

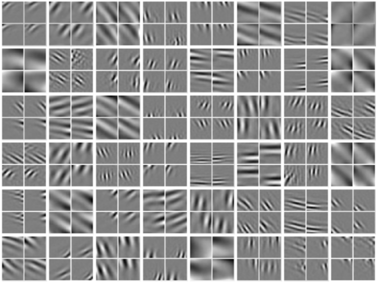


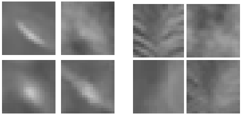
L_2-Pooling

Fully Connected Slow-Feature Auto-Encoders

Introducing L_1 induces selective (localized, independent) features

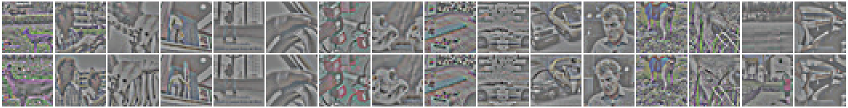
$$L_{sample} = \sum_{i=1}^{2} \frac{1}{2} \|x_i - W_d\, z_i\|^2 + \alpha \left| \sqrt{\sum_N (z_2)^2} - \sqrt{\sum_N (z_1)^2} \right| + \beta (|z_1| + |z_2|)$$



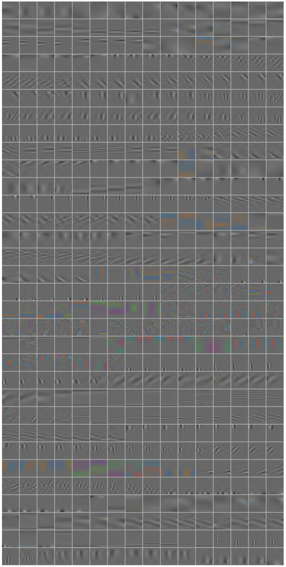


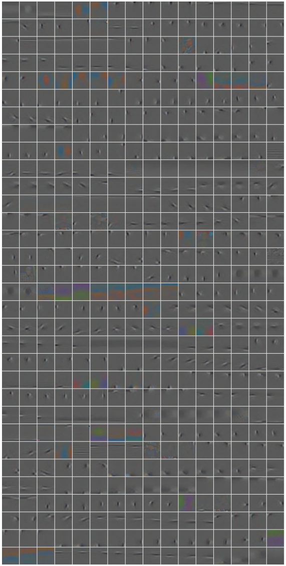
input	pinv
kNN input	kNN code

YouTube Dataset

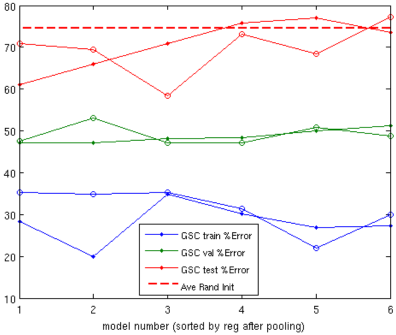


YouTube Features





- Pre-training helps, but fully connected features can't compete with convolutional networks



Convolutional Feature Learning

- Convolutional dictionaries are massively over-complete, which makes sparse inference potentially more difficult
- In the convolutional setting it may be necessary to have more sophisticated encoder (inference) to infer sparse codes

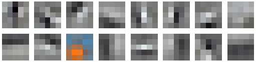


Figure: 1. Sparse Coding using FISTA Inference

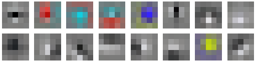
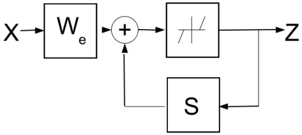
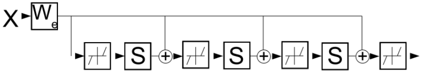


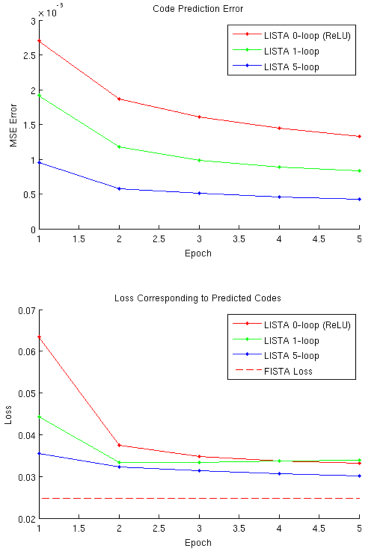
Figure: 2. Sparse Auto-Encoder using LISTA Encoder

- Proposed Concern: weak encoder inference may prevent learning sparse features
- Proposed Solution: use a more powerful network specifically designed to perform sparse as the encoder

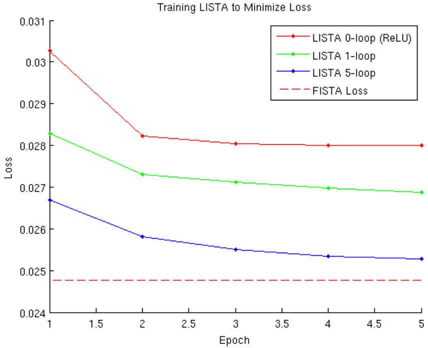




Code Prediction Performance

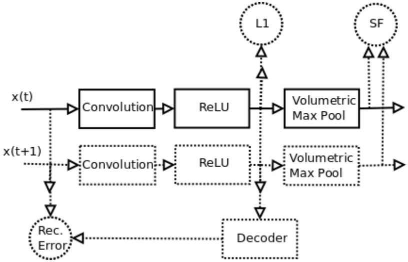


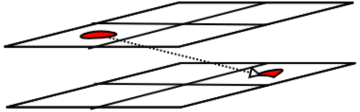
Inference Performance

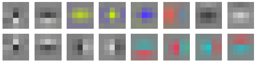


However, in *my* experiments I did not find that adding more than one-loop helps minimize the loss in dictionary learning

Training a Single Stage CNN







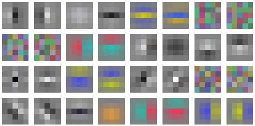


Figure: 16 and 32 filter convolutional dictionary learned with volumetric max pooling in space (4x4) and features (2)

Thank You

THE END