EE476 Audio-Visual Perception Model

Homework 3

Bereket Eshete 20150923

Due: 11:59 pm, April 27, 2018

Build 2-Layer CNNs using Sparse Auto encoders

In this homework, we will build the CNN architecture and train weights by PCA and sparse AE. We use previous implementation of Sparse Auto encoder (SAE) in homework 2.

We use the following CNN architecture and filter sizes in this homework.

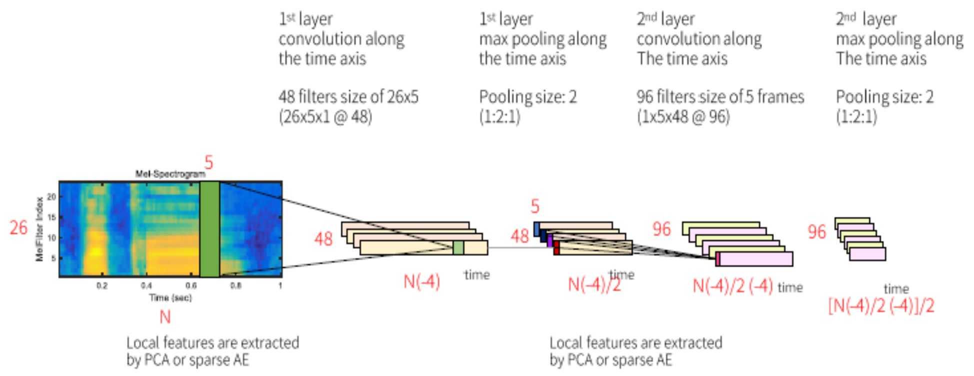

1st layer convolution along the time axis
48 filters size of 26x5 (26x5x1 @ 48)
1st layer max pooling along the time axis
Pooling size: 2 (1:2:1)
2nd layer convolution along The time axis
96 filters size of 5 frames (1x5x48 @ 96)
2nd layer max pooling along The time axis
Pooling size: 2 (1:2:1)
Mel-Spectrogram
26
5
N
48
N(-4)
time
5
48
N(-4)/2
96
N(-4)/2 (-4)
96
[N(-4)/2 (-4)]/2
Local features are extracted by PCA or sparse AE
Local features are extracted by PCA or sparse AE

Fig 1. CNN architecture given for audio

In this homework we will deal with audio data; hence, we use temporal convolution in both layers along time axis. If one is dealing with video instead of audio, one will have an option to use independent dimension or additional dimension approach. Which one brings the better result depend on the data and the parameters. Hence, one should use independent or additional dimension approach carefully to get the best result. However, here we will only deal with audio data and we will focus on using auto encoders to build our 2 layered convolutional neural network. The utility of auto encoders here is to help us train the weight, which means we could also use alternative unsupervised training methods such as PCA.

We will first see the overview of CNN with the obtained result with learning rate 0.1 with iteration 50,000. The diagram in the next page is specific to this example and while using different data one may use different dimension filters from those shown here. Next, we will see the learning convergence of the auto encoders used, following, we will see feature maps and extraction filters. Finally, we conclude by seeing how to increase the performance of our convolutional layer.

Overview

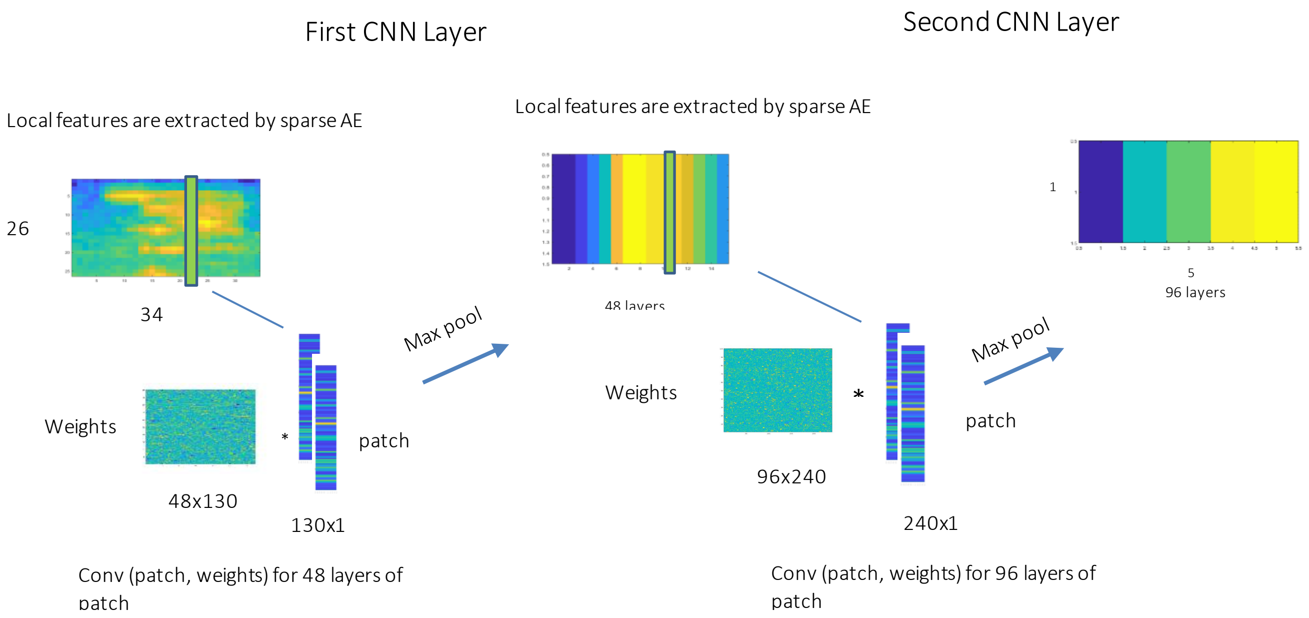
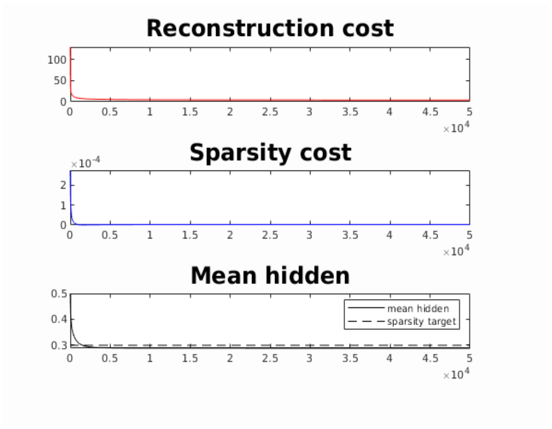
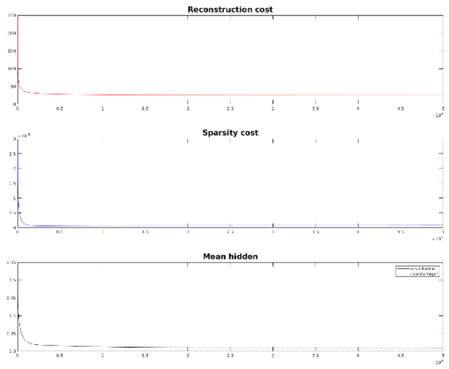

First CNN Layer
Second CNN Layer
Local features are extracted by sparse AE
26
34
Weights
48x130
*
patch
130x1
Max pool
Conv (patch, weights) for 48 layers of patch
Local features are extracted by sparse AE
48 layers
Weights
96x240
*
240x1
patch
Max pool
1
5
96 layers
Conv (patch, weights) for 96 layers of patch

Fig 2. Diagram from the results obtained from the homework

1. Learning Convergence

From the training data, we plot the cost functions using Visualization.m from homework 2. We use the auto encoder two times one for each CNN layer to train our weight, hence we can expect two cost function graphs. The results are shown below.

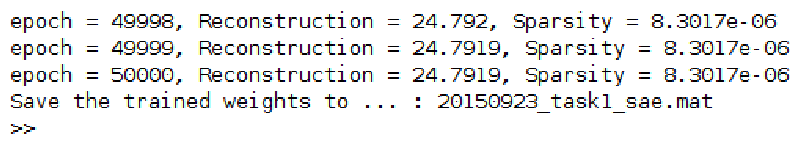




(a) (b)

Fig 3. (a) Training data cost function graph for 1st layer CNN. (b) Training data cost function graph for 2nd layer CNN

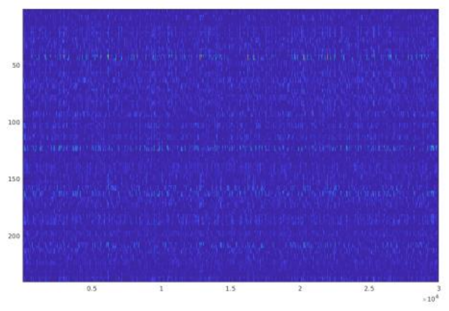
From the graphs, we learn that the cost function for both reconstruction and sparsity drop very quickly and stay almost constant: therefore, it is possible to use smaller iterations (epoch) to train the weights and obtain very similar trained weights. By looking the graphs closely, see appendix(1), we can observe both sparsity and reconstruction cost are higher for the second CNN layer than for the first CNN layer.

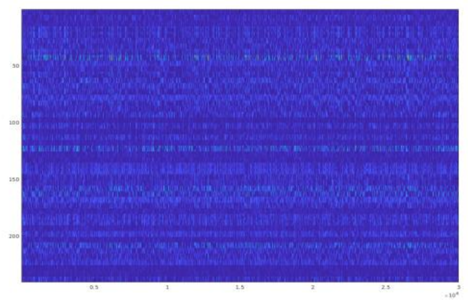


```
epoch = 49998, Reconstruction = 24.792, Sparsity = 8.3017e-06
epoch = 49999, Reconstruction = 24.7919, Sparsity = 8.3017e-06
epoch = 50000, Reconstruction = 24.7919, Sparsity = 8.3017e-06
Save the trained weights to ... : 20150923_task1_sae.mat
>>
```

By looking the final output above, we observe the reconstruction cost for the last CNN layer is around 24.792 and sparsity cost is around 8.3017×10^{-6}. From the graph, we can predict these values will not change much my increasing iteration of the auto encoder. This means increasing iteration of the encoder 50,000 will not improve the performance of our two-layered CNN.

2. Feature maps



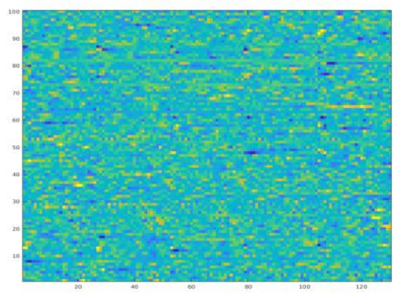


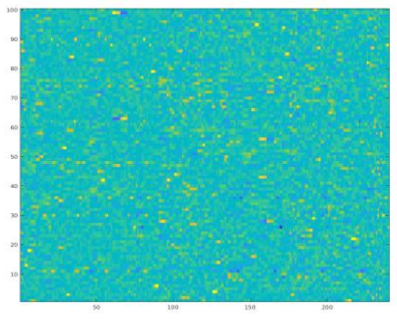
(a)130X30000 (b) 240x30,000

Fig 4. (a) data_audio for 1st layer CNN. (b) data_audio for 2nd layer CNN

We can observe the audio data by using a shell command imagesc(data_audio). The images are the inputs for out autoencoders by calling train_sae(data_audio). The weights in autoencoder are used to extract features from these input(feature maps). The first image shows input for first CNN layer autoencoder and the second image also used for second CNN layer autoencoder input.

3. Feature extraction filters





(a)48x130 (b) 96X240

Fig 5. (a) Trained weights for 1st layer convolution. (b) Trained weights for 2st layer convolution

Sparse auto encoder from homework 2 trains the weights shown here, the weights are trained with 100 hidden layer and 50,000 epoch. The weights are used to extract feature from the audio input file. We then take the weights for convolution with the cropped patches of the input, forming convolutional layer. In this homework, the weights are trained in the following way in main_task1.m with the function train_sae(data_audio, sae_config). Where, ‘data_audio’ is the input and ‘sae_config’ is layer number converted to string. The ‘sae_config’ here is used to notify the autoencoder, which layer the CNN, is in process. This will help the auto encoder save the different ‘weight and cost’ output file for every CNN layer.

4. Future improvements

The first two commonly improvement methods GPU and Batch normalization may not help you improve this layer. First, Increasing GPU will not improve this CNN because increasing iteration above 50,000 will barley reduce cost function. Second Batch normalization can help but only if you have more layers than two CNN layers, as it prevents diminishing gradients. However, we only have two layers hence batch normalization is not a choice too. Now we ruled out two methods that will not improve our CNN, the next two may help improve this 2 layered CNN. First is sparsity regularization, by varying the sparsity coefficients and sparsity targets, one may get a better result with certain sparsity parameters. Second is using improved algorithm such as using efficient convolutions’ conv and conv2’ during convolution. Matlab also provides deeplearning toolbox that will help compute your calculations efficiently with better built in algorithms and hence improve your CNN performance.

Questions

1. In case PCA for each architecture, draw eigenvalues of X^TX

We have a relation between singular value decomposition and eigenvectors as follows.

Given svd of $X=USV^T$

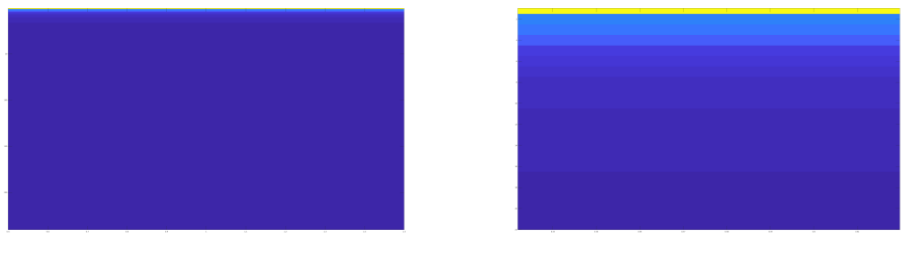
$X^TX = VS^2V^T$

The eigenvalues of X^TX can be found by, eigen diag matrix (Λ) = S^2

In pca2.m the following code is given:

```
[u,S,PC] = svd(Y);
% calculate the variances
S = diag(S);
V = S .* S;
```

By comparing the equation, we see that V is eigenmatrix of Y^TY. hence we draw the eigenvalues using a shell command imagesc(V). We get the following result.



(a) (b)

Fig 6. (a) drawing eigen value (b) zooming in at the top to observe gradient of the eigenvalues

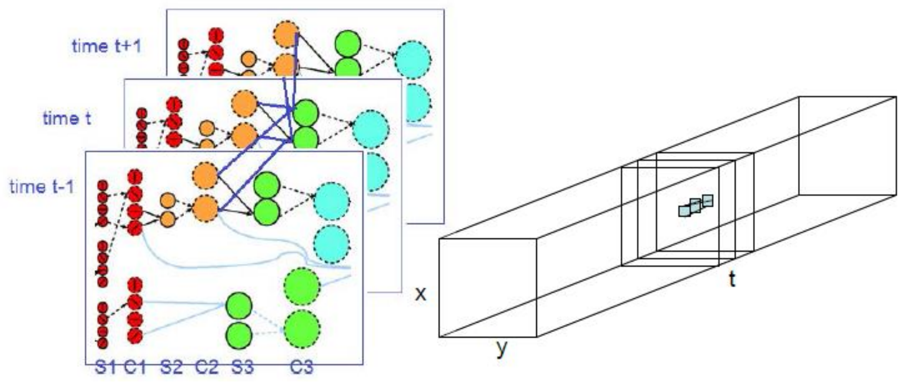
2. In case sparse AE for each architecture, draw your learning curves

See fig (3) or Appendix

3. For video data, briefly discuss the differences between architecture (1) and architecture (2)

These two architecture are the two types of time-Domain Feature Extraction.

Architecture (1) follows Independent dimension approach; we do spatial convolution followed by pooling in the first layer and temporal convolution followed by pooling in the second layer. Here we extract temporal features of spatial (or spectral) feature coefficients. However, we extract sequentially, First spatial/spectral features, then temporal features. As in figure (7) below, first extract spatial features along the x-y plane then along the time dimension extract temporal features.

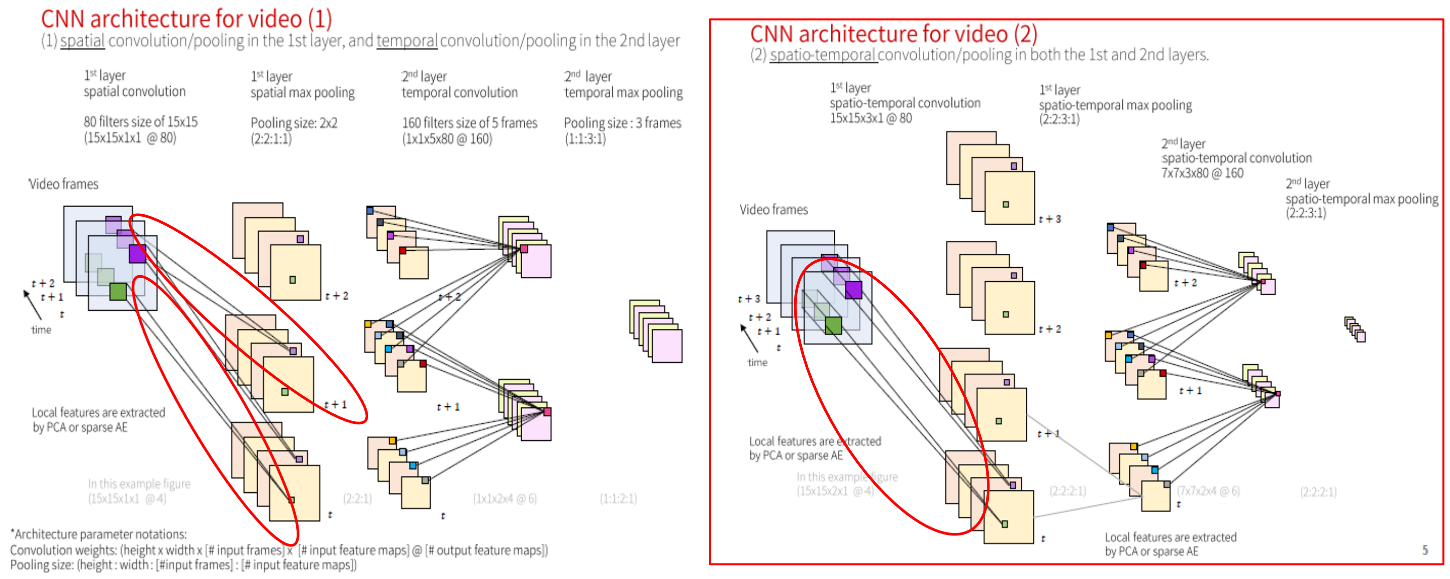


Fig(7) Independent dimension approach

Architecture (2) follows additional dimension approach spatio-temporal convolution/pooling in both layers. We extract spatial and temporal features at the same time not sequentially like architecture one.

This table below summarizes the differences between the two architectures

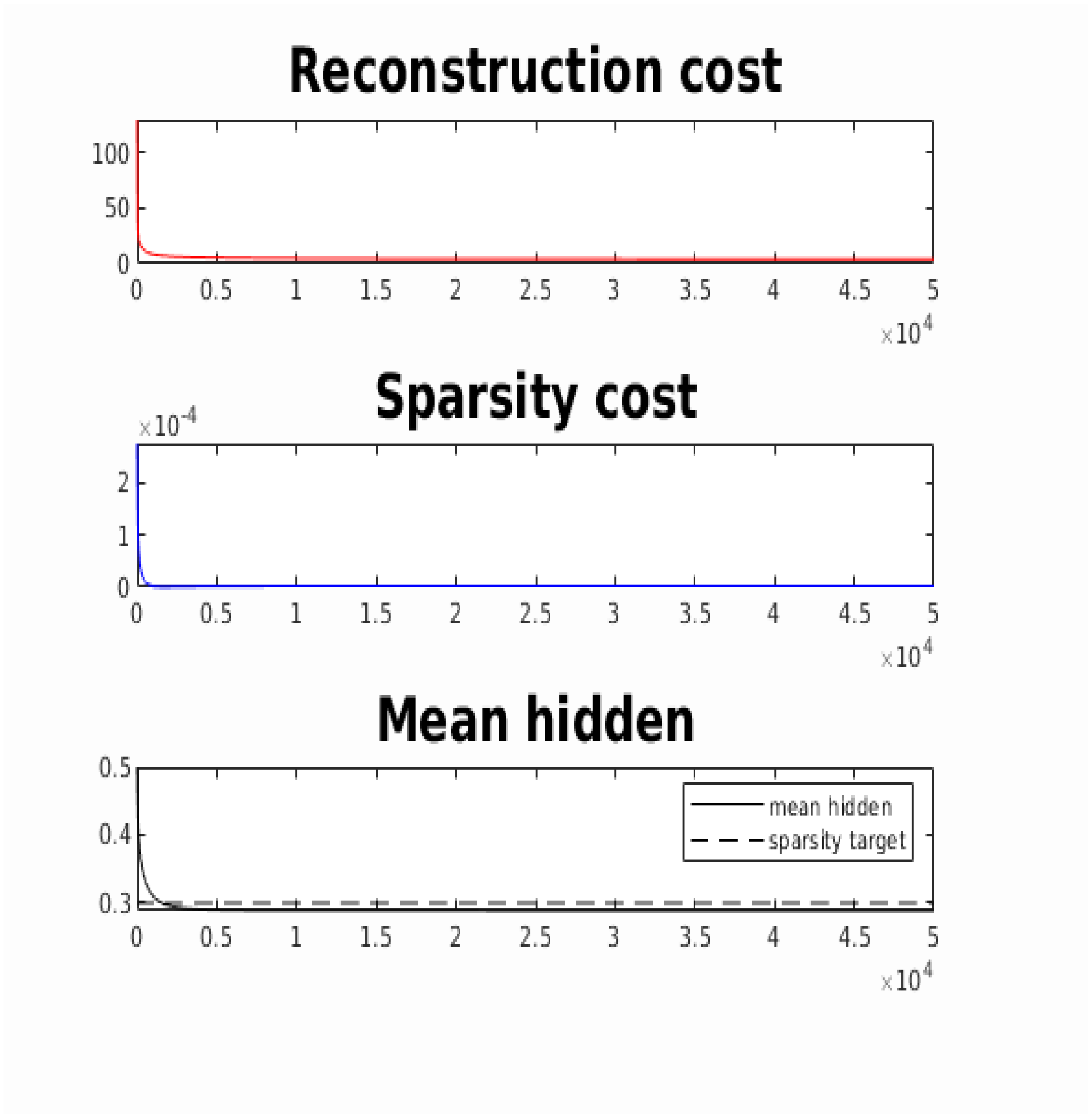
Architecture (1) – spatial convolution/pooling in the first layer and temporal convolution/pooling in the second layer.	Architecture (2) – spatio-temporal convolution/pooling in both layers
Independent dimension approach	Additional dimension approach
D dimension data Nothing special about this method, similar to normal convolution, first extract with spatial then extract temporal	Considers (D+1) dimension data - Time frequency representation - spatio- temporal representation 3D convolutional filters into separate spatial and temporal components yields significantly advantages in accuracy.


CNN architecture for video (1)
(1) spatial convolution/pooling in the 1st layer, and temporal convolution/pooling in the 2nd layer
1st layer spatial convolution
80 filters size of 15x15 (15x15x1x1 @ 80)
1st layer spatial max pooling
Pooling size: 2x2 (2:2:1:1)
2nd layer temporal convolution
160 filters size of 5 frames (1x1x5x80 @ 160)
2nd layer temporal max pooling
Pooling size : 3 frames (1:1:3:1)
Video frames
t + 2
t + 1
t
time
Local features are extracted by PCA or sparse AE
In this example figure (15x15x1x1 @ 4)
(2:2:1)
(1x1x2x4 @ 6)
(1:1:2:1)
*Architecture parameter notations:
Convolution weights: (height x width x [# input frames] x [# input feature maps] @ [# output feature maps])
Pooling size: (height : width : [#input frames] : [# input feature maps])
CNN architecture for video (2)
(2) spatio-temporal convolution/pooling in both the 1st and 2nd layers.
1st layer spatio-temporal convolution 15x15x3x1 @ 80
1st layer spatio-temporal max pooling (2:2:3:1)
2nd layer spatio-temporal convolution 7x7x3x80 @ 160
2nd layer spatio-temporal max pooling (2:2:3:1)
Video frames
t + 3
t + 2
t + 1
t
time
Local features are extracted by PCA or sparse AE
In this example figure (15x15x2x1 @ 4)
(2:2:2:1)
(7x7x2x4 @ 6)
(2:2:2:1)
Local features are extracted by PCA or sparse AE
5

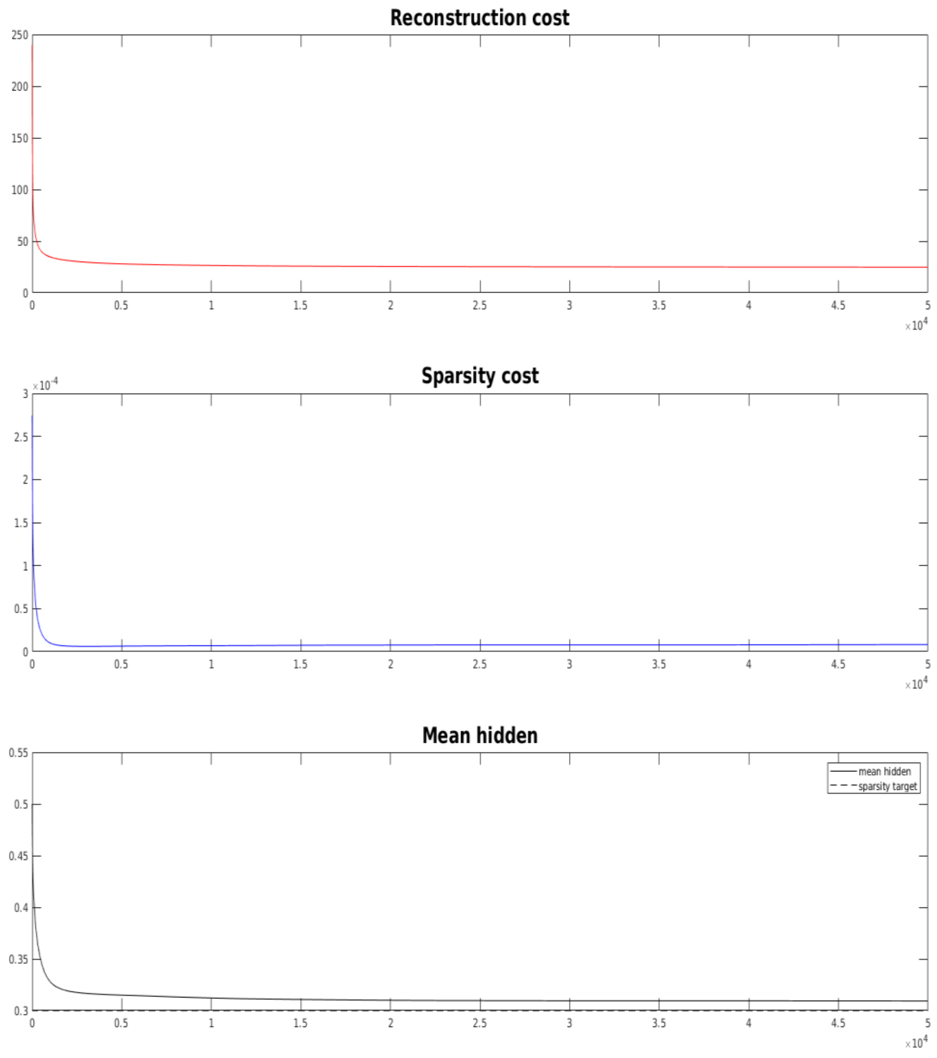
Fig(8) architecture (1) extracts features separate sequentially first along space axis then along time axis, however architecture (2) extracts space-time features simultaneously during convolution.

Appendix

(1) Cost graph of SAE for first CNN layer



(2) Cost graph for SAE for second CNN layer


Reconstruction cost
×10^4
Sparsity cost
×10^-4
×10^4
Mean hidden
mean hidden
sparsity target
×10^4