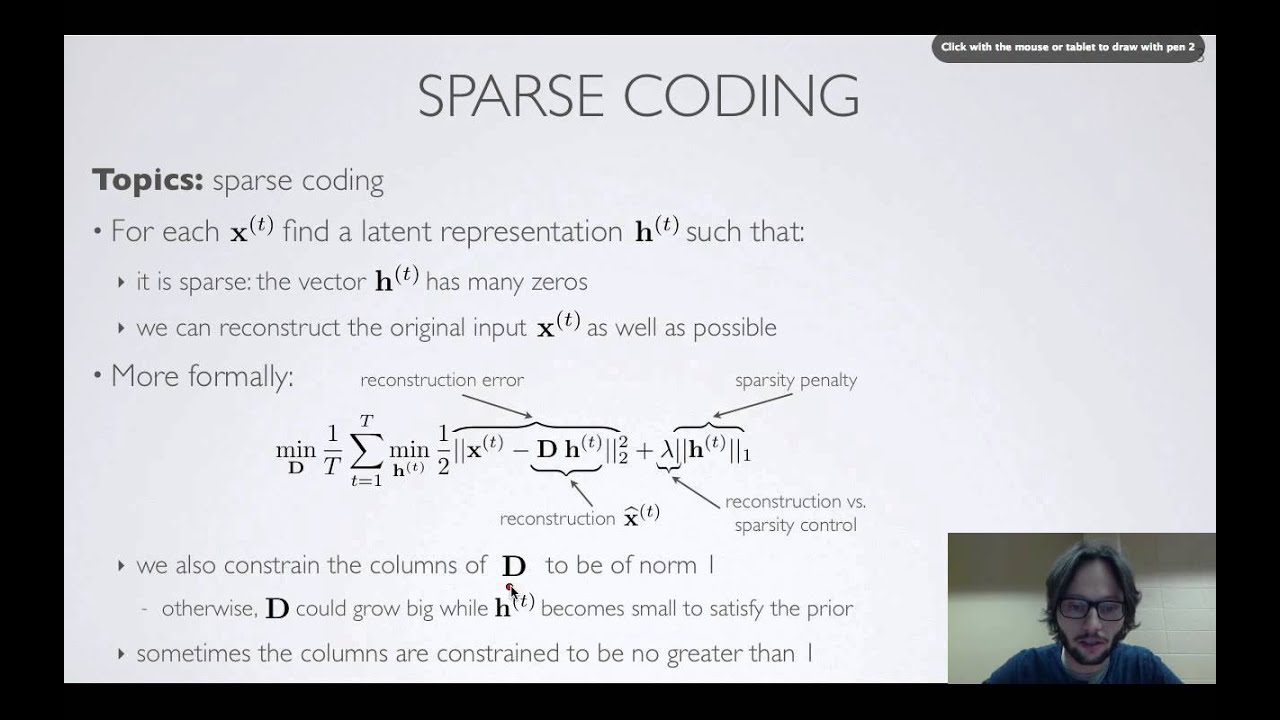
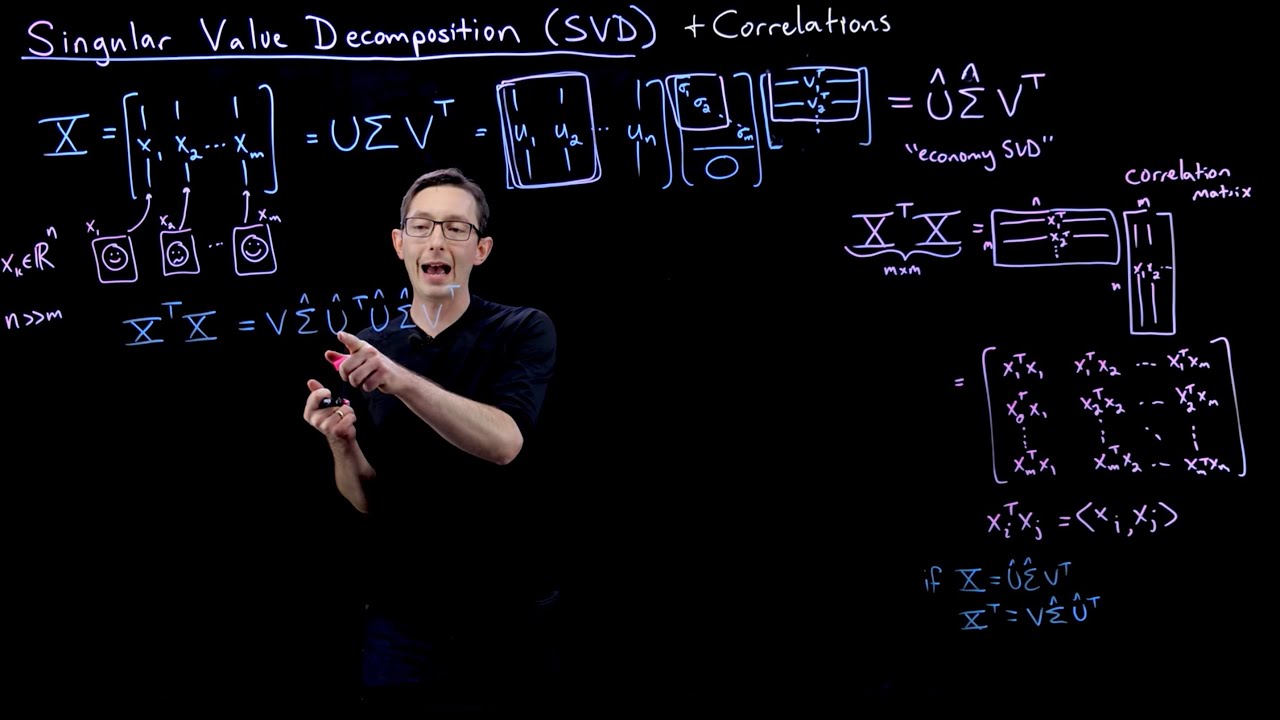
* Semantic segmentation – this is a method that is used for classifying the different classes in the image. This method for classification inherently cannot differentiate between the objects from the class or in other words it cannot differentiate between instances of the same class.
* One way in which semantic segmentation is generally done is by using one hot encoding. For each of the labels (objects) in the image we will create a convolution (layer) and then individually perform one hot encoding and then some of the pixels will be zeros (pixels where that particular object is not present is marked by 0s) and ones (pixels where that particular object is not present is marked by 0s). We do this for all the layers and then us the max function along all the layers by comparing the corresponding pixels values and creating a single image which contains all the objects listed.
* Then we will make the use downsampling and upsampling methods because it is computationally expensive to maintain the original dimensions throughout the model when passing it from one layer to another. The most common method of downsampling an image is - pooling which itself is of two types – Max (in this type of downsampling we select the maximum from a group of pixel values) and average pooling (instead computing the maximum of these pixel values we instead replace that grid with the average value of those pixel values). The common methods of upsampling are - Nearest Neighbour in this method suppose there is a 2x2 image then a new image shall be created in which each of the pixel will be duplicated to form 4 pixels containing same values and by doing this we have just changed our image from a 2x2 image to a 4x4 image and effectively upsampled the image. Another method for doing this is by using bed of nails. In this method for upsampling consider the following example. We have a 2x2 image and then we have predetermined places where these 4 pixel values will go in a 4x4 image and then the rest of the image will be filled with zeros. Another way around this problem is to make the use of max unpooling. In this method firstly we will apply max pooling for downsampling the image and then keep a track of the places where the maximum values occurred and then we will do the same thing that was done in the bed of nails but instead of placing the values in predetermined places we will place them at these places that we were keeping a track of and the rest of the places will initialized by zeros. But the most common method for upsampling is transposed convolutions. For this method consider a 2x2 matrix and a filter using which these convolutions are to be performed. Each element from the 2x2 matrix is multiplied with all the elements of the 4x4 matrix and the elements which overlap, will be sumed up and this will help in our process of upsampling.
* Sparse Coding – It is used for unsupervised learning. It is the learning where we have only inputs which are not labelled in other words we have only x vectors and no y vectors. In this model we have to find a hidden layer h(t) for each x(t) such that this hidden layer is sparse ie has many zeros in it and we want it to lean as many features of the input as possible. For the latter task we will try to reconstruct the image. For this we use the following optimization. Reconstruction error is required to be as small as possible so that the reconstructed image is a better representation of the actual image. D here is called the dictionary. If we are trying to classify or reconstruct an image then this dictionary will contain the edges of that number. Now there is a trade-off between reconstruction and sparsity. If the sparsity of the hidden vector is more than the image that is to be reconstructed will have less similarities to the original image and vice-versa. This trade-off can be controlled by the lambda parameter. The columns of the dictionary are also required to have a L1 norm of 1. This is done to prevent the hidden vector from becoming too small or sparse and Dictionary becoming too large. This is basically used to find parts of the image.



Singular value decomposition (SVD) – Singular value decomposition is used to find the three matrices that could represent an image vector, X. This image vector (X) is made by sometimes downsampling the image or using it as it is and the each of the pixels of this image are converted into a corresponding vector which contains each of the pixel values that are stacked up to form a column vector and then stacking all these column vectors would lead to the formation of the X matrix. This is the matrix that is to be represented by using three other matrices, namely U, S and V, that when multiplited together form the matrix X. Consider for example we are trying to find the recognize the image of person using this method. In this case the dimension of the column of U matrix will be the same as that of the X matrix. All the columns of U are arranged in a way such that as we move along the columns, the first ones are the most important and contain the so called eigen values for the faces, which form the bases for the image representation of images in X. Both the U and V matrices are unitary matrices which means when we multiply the matrix with its transpose we get identity matrix. The S matrix also contains elements in a hierarchical order of importance. The column of the transpose of the V vector will tell us the the exact mixture of the eigen values of the faces (columns of U matrix) that is to be taken to reconstruct the first image. The importance of the columns of U is encoded in the values of the sigma matrix, S. The X matrix has m linearly independent columns and the number of columns that U has is equal to n as a result only m out of n will be useful and all of the rest will be redundant. This is the reason that the lower value of sigma matrix, S has all the value as zeros and the rest of the values are along the diagonal which has a length of m. This is done to ensure the dimensionality of matrices X and the resultant after matrix multiplication match. The matrix multiplication will give us sigma\_1\*v1\*u1 +sigma\_2\*v2\*u2 +….+ sigma\_m\*u\_m\*v\_m and the rest of the values will be zeros because the entries of the sigma matrix are zeros below the m points. But since the sigmas are in decreasing order of importance, the values that they will have will be smaller and smaller so we can leverage this fact and choose only r terms whose contributions to the recreation of our image is significant. This will reduce the computational cost. They multiplication that will be performed between v\_i and u\_i will be an outer product instead of an inner product. The values of the U and V which will contain m values because of the zeros that we will get are called the economical SVD values. The values of the U and V which will contain r values because of the zeros that we will get are called the tilda SVD values. Ekard Young theorem states that the forbenius norm of the original image and the reconstructed image will be the least when the values of the reconstructed X matrix are chosen to be equal to the U\_tilda, V\_tilda and sigma\_tilda.



GANs (Generative Adversarial Networks)

These networks generally consists of two parts the generative model and the discriminator model. In this type of modelling the network finds a mapping from the input space to the output space via a possible distribution that the model learns from the training of the data. We input a noisy image into the network and then use it to generate image in case of computer vision that are based on the distribution tatt the model has learned during the training time. The generative model is used to find the mapping or find a so called function that can be used to describe the data that is presented to it. The discriminator model outputs a single number that is used to tell the whether the image that is created came from the generator’s distribution or from the input distribution. We train the model in such a way that generator’s loss function is the discriminator network and the generative model is improved by this continuous race between the generative model and the discriminator model. sThe adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons. To learn the generator’s distribution pg over data x, we define a prior on input noise variables pz(z), then represent a mapping to data space as G(z; θg), where G is a differentiable function represented by a multilayer perceptron with parameters θg. We also define a second multilayer perceptron D(x; θd) that outputs a single scalar. D(x) represents the probability that x came from the data rather than pg. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize log(1 − D(G(z))). The loss function that was used is a complex one in which G is trying to minimize the loss and D is trying to maximize the loss. Whenever one of the model fails to provide the desired output the other is awarded and that model is penalized. This way both the models keep on competing among each other and this helps in mutual improvement of both the models.

Min\_(G) max\_(D) V (D, G) = Ex∼pdata(x) [log D(x)] + Ez∼pz(z) [log(1 − D(G(z)))].

The above equation is the loss function.