Deep Unsupervised Hashing for Image Retrival

Shaif Chowdhury

May 7, 2020

1 Abstract

Recent studies on vision and learning have shown that hash codes can be used for processing massive amount of images using minimized storage and computation. In particular, hashing using deep learning can be very useful for Image retrieval in real time scenario. But, deep hashing generally needs labeled images and unsupervised hashing hasn't shown satisfactory accuracy due to either the relaxed optimization or absence of pairwise similarity. This work would focus on generating binary features from images while preserving image similarity, so that it can be used for searching or classification task. Here, we would use an activation function with an Autoencoder type architecture that can map input images to binary space. To test the competitiveness of the architecture, we evaluate the proposed model on MNIST data-set and compare the performance with state-of-the-art approaches.

2 Introduction

In recent years there has been a lot of research on hashing for image retrieval problems, due to its low storage requirement and high searching speed. But most existing deep hashing networks are supervised[LY14] and use labels[HS18] or pairwise similarity [ZC20] to get discrete features. There are quite a few ways to do unsupervised hashing, but usually they have low accuracy or require supervised[HC17] pre-training to avoid over-fitting. First, we would look at the challenges of using deep learning to hash and then discuss ways to overcome those.

Broadly there are two approaches to hashing : Locality sensitive hashing and Learning based hashing[JJ17]. LSH generally fails to preserve Semitic similarity needed for searching or classification task. Learning based hashing has proposed to overcome this, but most approaches rely on labelled training data. So the trend in hashing is to explore semi-supervised[JC10] or unsupervised hashing [Sov17]. We would mostly explore unsupervised ways here.

Proposed unsupervised models mostly need pairwise information or neighbourhood similarity. Again this makes it inadequate for real-time image retrieval task. There are a few other approaches that use hand-crafted features [CH15]. These suffer from the limitations of PCA and dimension reduction techniques : generally don't capture similarity, are not robust to noise and image transformations. other learning based approaches do the feature extraction and apply the binary constrains separately. Most deep learning based approaches have two steps : first is the dimensionality reduction probably with an encoder, and a function with binary constraint to produce and optimize hash codes. Usually a step function is used to convert learned features to binary space. But, this function is difficult to optimize with standard back propagation technique, since it's not differentiable on any non zero value. In practice a lot of approaches just ignore the binary constraint and the resulting features are binarized using some threshold. For example one can apply PCA to reduce dimension and apply step function to produce binary code. This is a 'filter' approach which leads to a sub-optimal solution. So, the challenge is to optimize the features in binary space while using gradient descent for optimization. We propose to solve this problem with an activation function that is differentiable and applies

binary constrains.

So, an autoencoder [Bal11] type architecture would be appropriate to generate hash codes from images. Autoencoders are neural networks that can compress the input into a latent-space representation, and then reconstruct the output from this representation. Our model would take advantage of convolutional deep autoencoders to do the feature extraction. We would apply the constrains over multiple dense layers to get compact binary codes.

The capability of the architecture would be tested on MNIST data-set on image retrieval task. We would compare performance with locality sensitivity hashing and Regularized autoencoder hashing[Sov17].

The rest of the paper is structured as : Section 2 which reviews the related works, Section 3 that presents the method and the architecture, Section 4 presents results on image retrieval task and Section 5 concludes the paper.

3 Related Work

Initial work in Hashing [JJ14] has been based on Locality sensitive hashing and embedded hashing with hand crafted features. there has been a lot of improvements to LSH like multi probe hashing or non linear projection etc. LSH has also been extended to include karnel[**ker**] based hash function. There is also Iterative qauntization (ITQ)[YP13], which uses iterative optimization strategy to find projections with minimal binarization loss.

Different from LSH the learning based approach [JJ17] takes advantage of machine learning algorithms to generate hash function. This approach can be divided into supervised and unsupervised. First we would discuss few of the supervised approaches. [LY14] is based on pairwise similarity. Their model creates a pairwise similarity matrix and uses a convolution neural network to create hash function.[HC16] this is a deep supervised model based on a loss function that is used to increase discriminability between the binary codes. So, performance of these supervised models generally depend on availability of class labels and pairwise similarity or triplets as ground truth.

In unsupervised hashing there are approaches based on semantic hashing and restricted Boltzmann machine. Our proposed model falls in the category of unsupervised deep hash function. So, the first step to solving this problem is to implement a feature extraction model. Here I would discuss some deep learning based feature extraction frameworks. Variational autoencoder[KW19] is a good candidate for this. Another way to extract features is to do inverse mapping[CB18] from a Generative Adversarial Network or optimizing the latent space of generative network[Szl17] with just reconstruction loss. Another solution is using Adversarial Autoencoder[**4**] which draws samples from a specifically defined prior distribution to generate meaningful images. Recent extension to this is Adversarial Variational Bayes[Lar17] which trys to establish a connection between VAE and GANs, though in practice it doesn't provide much improvement to Adversarial Autoencoder.

The papers described above are generating the latest structure in continuous space. But, the problem here requires the latent space to be in discrete space. It is infeasible to get discrete latent structure through a neural network, because it makes gradient estimation hard. Developing techniques to optimize a discrete variable is an active research area, with several large developments in the last few years.

This paper[Sov17] achieves this with a regularized autoencoder. But, this doesn't really preserve the image similarity in binary space. This paper[**5**] tries to solve it by introducing a neighbourhood structure based loss. Again this would require supervised information for neighbourhood structure. The goal should be to optimize binary features without using supervised training.

4 Method

In a nutshell, the problem is - given images of size m(i) we need a function h(x) that maps the images to a lower dimensional $b(j)$ $b \in (0,1), where j \leq i$.

Generally the function used for binarization is signum function :

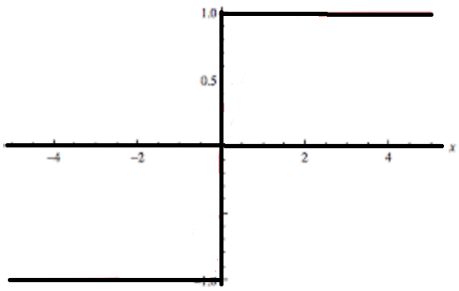
$f(x) = -1 \; for x \leq 1$

and 1 for $x \geq 1$.

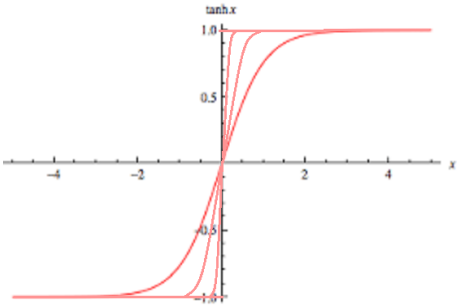
The problem with this function is that it's gradient is zero at all non zero inputs. In a neural network this leads to the vanishing gradient problem. So, to learn binary features a different activation function has to be introduced that can approximate to binary code while being optimizable with gradient descent. This can be achieved by a modified tanh function :

$tan'(x) = \frac{2}{(1+e^{-bx})-1}$

as b goes to infinity this function would approximate to sign function

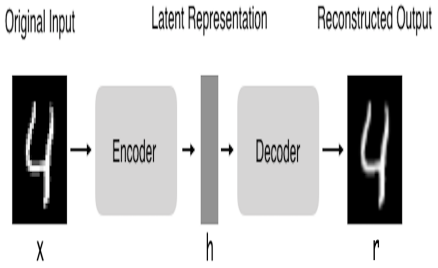


1. Signum function



2. tan'(x) function

So, an autoencoder type architecture would be needed to generate hash codes from images. Autoencoders are neural networks that can copy their inputs to their outputs. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation. We just need to convert the latent representation to binary from. For this the above mentioned tan'(x) function can be used over one or multiple layers to force the latent representation to binary. The function is differentiable and can be used with standard back-propagation.



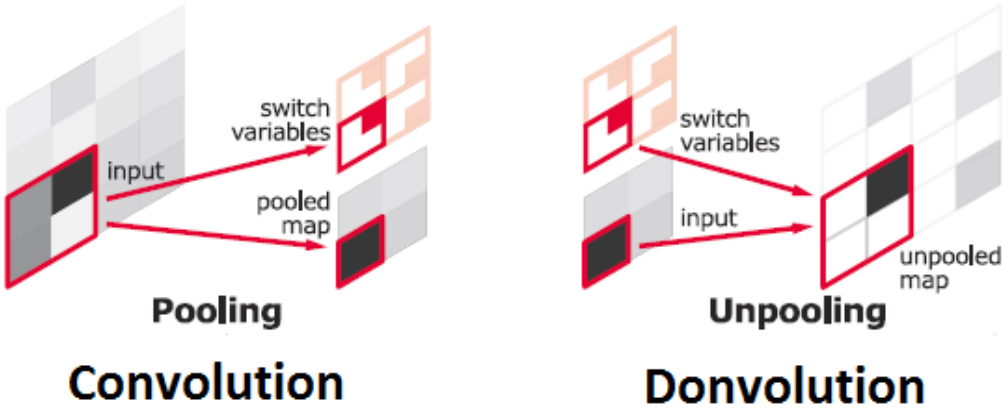
1. This is the diagram of a simple Autoencoder that takes an image as input, creates latent code using an encoder and reproduces the image using decoder

Model Architecture

So our model is based on convolutional autoencoder which would map images to latent representation and reconstruct the original image using a decoder. For RGB images the model would mostly follow VGG model [SZ15] in both encoder and decoder. That generally means convolution layers, max pooling layers followed by dense layer, filter size of (3*3), max pooling stride size (2*2).

Convolutional Encoder-Decoder

Lets assume the network has L convolution layers. In that case the output would be $x \in R$(A*B*C) where A*B is the size of the image and C is the number of channels, 1 for gray-scale. So, for encoder it would be followed by max-pool (2*2) and for decoder up-sampling (2*2). The output is given by $x = (\sigma(W * x + b))$ where W is the weight and b is the bias. σ is the activation function. We would use RELU and Leaky-RELU as activation function for convolution layers and our custom function on dense layers to apply the binary constrain. We start with a learning rate of $1 * 10^{-3}$ and linearly decrease it to $1 * 10^{-5}$. the parameters are learned with ADAM optimizer where the loss function is the mean square error between input and reconstruction.



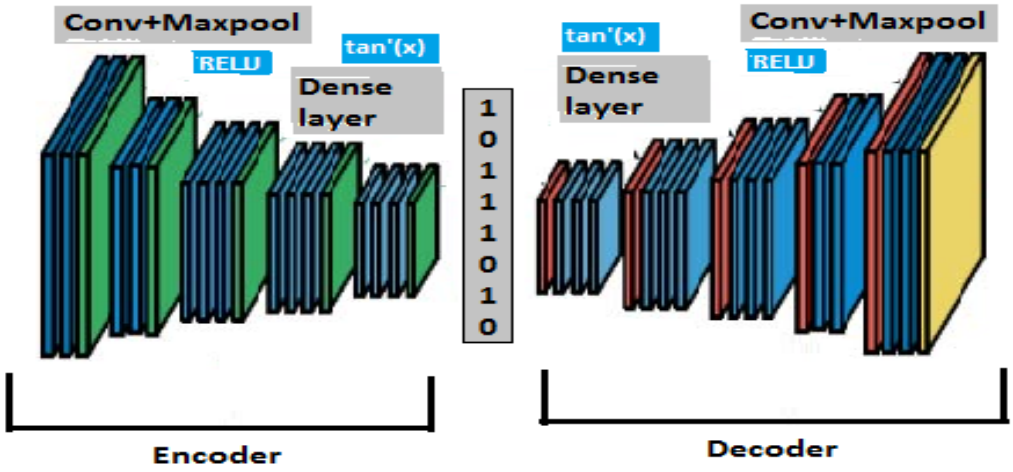
Max-pooling and Unpooing for Convolutional Autoencoder

Fully connected layer

Between the convolutional encoder and decoder we would have fully connected dense layers that would reduce the dimension and also apply the binary constrain. For larger RGB images it would better to use fully convolution layers connecting the encoder, decoder and apply binarization on the final convolutional layers. We use the tan'(x) function to apply binary constrain.

$tan'(x) = \frac{2}{(1+e^{-bx})-1}$

We would increase the value of b in each layer until the output of the final layer of the encoder is completely discrete. If the image can be reconstructed from the binary latent representation then the latent space contains enough high level information to preserve similarity.

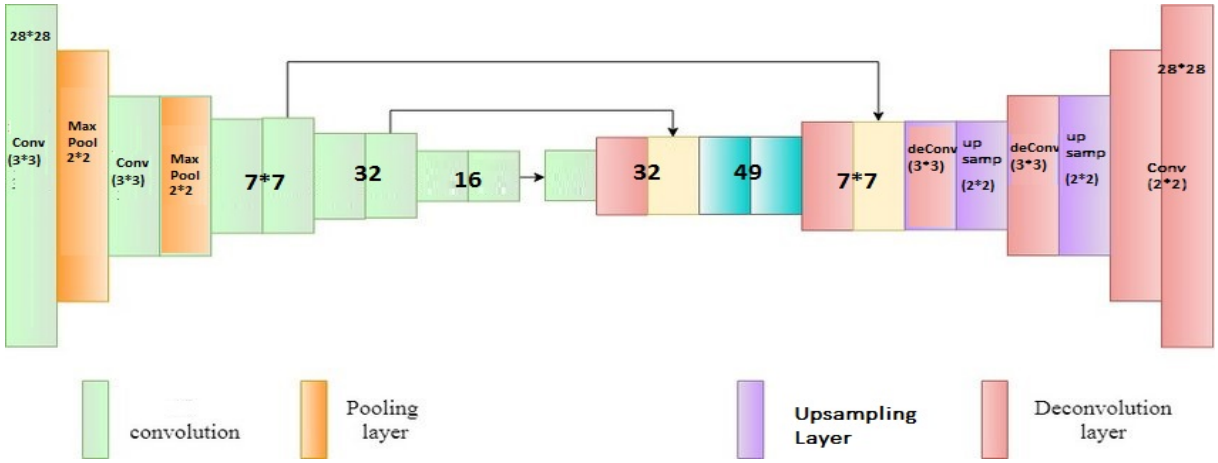


This a general model architecture. For convolution layers RELU activation function has been used. Binarization has been applied using the tan'(x) function over Dense layers

5 Result

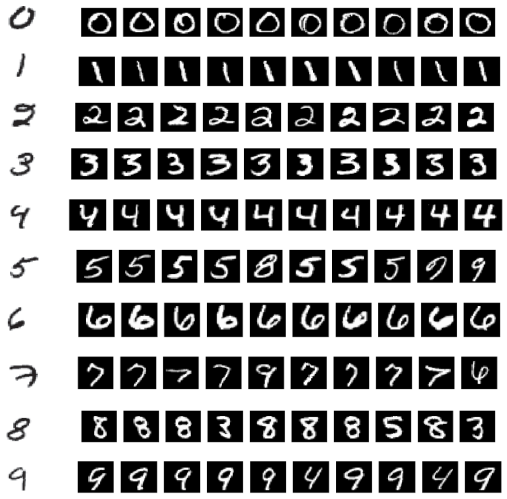
Here we would describe the datasets used, give some implementation details and finally present the results with some comparison with other models.

MNIST : Mnist [AI19] is a standardised dataset for handwritten digits and it is widely used for testing classification or image retrieval problems. It contains 70,000 28*28 images of handwritten digits ranging from 0 to 9. It is divided as 60,000 for training and 10,000 for test. We would train the model on training images. Once it is done we would use the encoder to create hash codes for the test images. Now we would randomly select an image, generate binary code and search through the codes derived from test set. This would be used as a test-case for image retrieval performance. We would use a python based hash-table to store and query into the data.



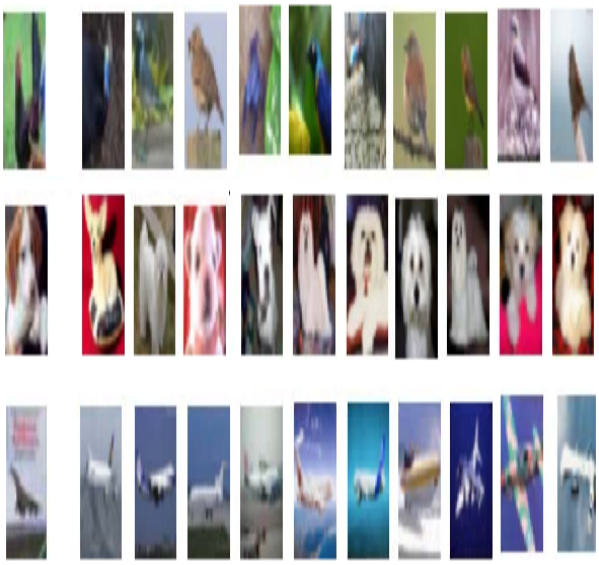
This is the architecture for training on MNIST data generating 16 bit binary code

Implementation Details : Here we describe the model architecture for training on MNIST data. Each convolution layer is set to have 16, 32, and 64 filters with kernel size of 3x3 followed by dense layers to reduce the dimension to 16. The max-pooling and unpooling layers are set to reduce the dimension of the feature maps by 2 and vice versa (i.e. their stride is of [2,2]). Next, we apply the binary constraints by gradually increasing the b value until the hash layer contains binary values Then, we fine tune the network by gently decreasing the learning rate and continue training. The evaluation is based on searching for image Hash codes. The top ten retrieved search results can be visualised based on lowest hamming distance.



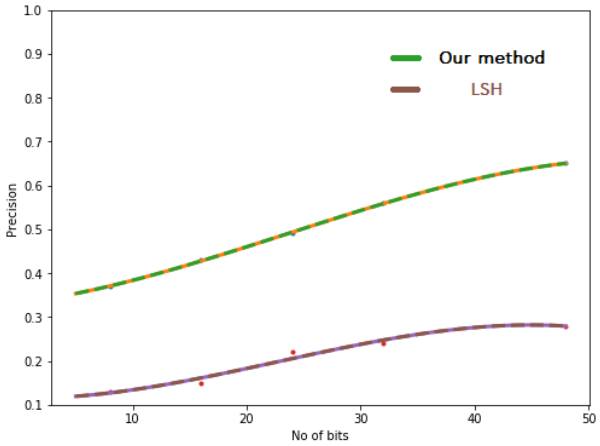
Here are the results retrieved on MNIST digits

CIFAR-10 : This dataset [ÇD15] contains 60,000 images of different classes like car, bird, dogs etc. This dataset is a bit more diverse than MNIST and would be better to check the accuracy with a precision recall curve. It contains 60,000 images of 32*32 spread across 10 different classes. Similar to MNIST we would extract binary code from test set and store it for testing image retrieval.



Here are the results retrieved on some of the CIFAR classes The metric for comparison would be accuracy among retrieved search results. A precision vs no of bit curve would be used to compare the performance with LSH over MNIST dataset.

The comparative results are based on a graph of Average precision plotted against the no of bits. The Average precision is computed on MNIST dataset based on 90 retrieved samples showing the improvement over LSH.



As it's apparent from the diagram that the model improves the precision significantly. Unlike previous hashing approaches that require pairwise information, out method would learn binary descriptors in completely unsupervised manner. So, this method is practical and flexible for various applications. Our descriptor gives higher accuracy compared to popular approaches like SIFT+KNN. This result is also competitive to deep learning based hashing approach [Sov17].

6 Conclusion

We contribution here is a architecture based on deep convolutional autoencoders to learn compact binary hash codes. The convolutional layers help learning features while keeping high level information. To turn the central layer into binary codes, a constraint binary constrain has been introduced. Experimental results demonstrate the competitiveness of our approach over state-of-the-art methods in image retrieval.

References

[JC10] Sanjiv Kumar Jun Wang and Shih-Fu Chang. “”Semi-supervised hashing for scalable image retrieval””. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010).

[Bal11] Pierre Baldi. “”Autoencoders, unsupervised learning and deep architectures””. In: *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning workshop* (2011).

[YP13] Albert Gordo Yunchao Gong Svetlana Lazebnik and Florent Perronnin. “”Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval””. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013).

[JJ14] Jingkuan Song Jingdong Wang Heng Tao Shen and Jianqiu Ji. “”Hashing for Similarity Search: A Survey””. In: *Arxiv* (2014).

[LY14] Rongkai Xia Yan Pan Hanjiang Lai1 Cong Liu1 and Shuicheng Yan. “”Supervised Hashing for Image Retrieval via Image Representation Learning””. In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence* (2014).

[ÇD15] Rasim Caner Çalik and M. Fatih Demirci. “”Cifar-10 Image Classification with Convolutional Neural Networks for Embedded Systems””. In: *ACS 15th International Conference on Computer Systems and Applications* (2015).

[CH15] Chun-Che Chen and Shang-Lin Hsieh. “”Using binarization and hashing for efficient SIFT matching””. In: *Journal of Visual Communication and Image Representation* (2015).

[SZ15] Karen Simonyan and Andrew Zisserman. “”VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION””. In: *International Conference on Learning Representations* (2015).

[HC16] Shiguang Shan Haomiao Liu Ruiping Wang and Xilin Chen. “”Deep Supervised Hashing for Fast Image Retrieval””. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[HC17] Kevin Lin Huei-Fang Yang and Chu-Song Chen. “”Supervised Learning of Semantics-Preserving Hash via Deep Convolutional Neural Networks””. In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* (2017).

[JJ17] Jingkuan Song Jingdong Wang Heng Tao Shen and Jianqiu Ji. “”A Survey on Learning to Hash””. In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* (2017).

[Lar17] Andreas Geiger Lars Mescheder Sebastian Nowozin. “”Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks””. In: *arXiv:1701.04722 [cs.LG]* (2017).

[Sov17] Frédéric Jurie Sovann En Bruno Crémilleux. “”UNSUPERVISED DEEP HASHING WITH STACKED CONVOLUTIONAL AUTOENCODERS””. In: *IEEE International Conference on Image Processing (ICIP)* (2017).

[Szl17] Piotr Bojanowski Armand Joulin David Lopez Paz Arthur Szlam. “Optimizing the Latent Space of Generative Networks.” In: *arXiv:1707.05776 [stat.ML]* (2017).

[CB18] Antonia Creswell and Anil Anthony Bharath. “”Inverting the Generator of a Generative Adversarial Network””. en. In: *IEEE Trans Neural Netw Learn Syst* (Nov. 2018).

[HS18] Yang Long Haofeng Zhang Li Liu and Ling Shao. “”Unsupervised Deep Hashing With Pseudo Labels for Scalable Image Retrieval””. In: *IEEE Transactions on Image Processing* (2018).

[AI19] Yago Saez Alejandro Baldominos and Pedro Isasi. “”A Survey of Handwritten Character Recognition with MNIST and EMNIST””. In: *applied sciences* (2019).

[KW19] Diederik P Kingma and Max Welling. “”An Introduction to Variational Autoencoders””. In: *Foundations and Trends in Machine Learning* (2019).

[ZC20] Yuewei Lin Zheng Zhang Qin Zou and Long Chen. “”Improved Deep Hashing With Soft Pairwise Similarity for Multi-Label Image Retrieval””. In: *IEEE Transactions on Multimedia* (2020).