Afyon Kocatepe Üniversitesi Fen ve Mühendislik Bilimleri Dergisi

Afyon Kocatepe University Journal of Science and Engineering

AKÜ FEMÜBİD **23** (2023) 065102 (1458-1465) AKU J. Sci. Eng. **23** (2023) 065102 (1458-1465)

DOI: 10.35414/akufemubid.1236064

Araştırma Makalesi / Research Article

Unsupervised Image Hashing Using a Deep Convolutional Encoder-Decoder Model for Fast Image Retrieval

Enver AKBACAK*

Haliç Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği, 5. Levent Mah. 15 Temmuz Şehitler Cd. No:14/12 34060 İstanbul, Türkiye,

Corresponding author* e-mail: *enverakbacak@halic.edu.tr ORCID ID: https://orcid.org/0000-0002-6753-7887*

Geliş Tarihi: 17 Ocak 2023 Kabul Tarihi: 6 Kasım 2023

Abstract

Keywords

Unsupervised Learning; Deep Learning; Encoder-Decoder; Hash Codes.

Image hashing methods transform high-dimensional image features into low-dimensional binary codes while preserving semantic similarity. Among image hashing techniques, supervised image hashing approaches outperform unsupervised and semisupervised methods. However, labelling image data requires extra time and expert effort. In this study, we proposed a deep learning-based unsupervised image hashing method for unlabeled image data. The proposed hashing method is built in an end-to-end fashion. It consists of an encoder-decoder model. As a novel idea, we used a supervised pre-trained network as an encoder model, which provides fast convergence in the training phase and efficient image features. Hash codes are extracted by optimizing those intermediate features. Experiments performed on two benchmark image datasets demonstrate the competitive results compared to unsupervised image hashing methods.

Derin Konvolüsyonel Kodlayıcı-Kod Çözücü ile Görüntü Hash Kodlarının Çıkartılarak Hızlı Görüntü Erişiminin Gerçekleştirilmesi

Öz

Anahtar Kelimeler

Denetimsiz Öğrenme; Derin Öğrenme; Kodlayıcı-Kod Çözücü; Hash Kodları.

Görüntü hash kodlarını elde eden metotlar, yüksek boyutlu ve sayısal olan görüntü özniteliklerini, görüntüler arasındaki anlamsal ilişkileri koruyacak şekilde daha düşük boyutlu ikili kodlara dönüştürürler. Hash teknikleri arasında denetimli öğrenmeye dayalı yöntemler, denetimsiz ve yarı denetimli öğrenme metotlarına göre daha verimlidirler. Ancak denetimli öğrenmeye dayalı yöntemler görüntülerin anlamsal etiketlerini kullanırlar ve bu da ilave bir çalışma ve uzman emeği gerektirir. Bu çalışmada etiketsiz görüntüler için denetimsiz öğrenmeye dayalı bir yöntem sunulmuştur. Bu yöntem uçtan uca kesintisiz entegre bir yöntemdir. Yöntem kodlayıcı-kod çözücü tabanlıdır. Yeni bir öneri olarak, kodlayıcı kısmında önceden denetimli olarak eğitilmiş bir derin ağın bloklarını kullanmamız, eğitim aşamasında hızlı yakınsamayı ve görüntü özniteliklerinin verimli olmasını sağlamıştır. Hash kodları ise bu özniteliklerin optimize edilmesi ile çıkarılmıştır. İki bilinen görüntü veri seti ile gerçekleştirilen deney sonuçları önerilen yöntemin diğer denetimsiz öğrenme yöntemlerine kıyasla rekabetçi sonuçlar verdiğini göstermiştir.

1. Introduction

Artificial intelligence has become an increasingly demanded tool in various domains (Mchergui et al. 2022, Baduge et al. 2022, Minh et al. 2022, Nahavandi et al. 2022, Aslan and Subaşı 2022, Akalın and Veranyurt 2022, Mutlu et al. 2021, Şendir et al. 2019). Among them, the need for efficient and fast image retrieval is indispensable. Hashing methods allow quick retrieval as a simple XOR operation can calculate the Hamming distance faster. On the other hand, deep learning-based image features are more efficient. That is why deep learning-based image hashing methods have recently been prevalent (Wang et al. 2022, Singh and Gufta 2022, Patel and Kasat 2017).

Hashing methods are divided into three main groups based on image label information. Supervised image hashing methods require labelled images in the training phase (Qin et al. 2022, Passalis and Tefas 2021, Mojoo and Kurita 2021). However, image labelling is an extra process that requires additional time and expert effort. Especially for large-scale datasets, it is cumbersome. Those methods are generally more efficient than those unsupervised and semisupervised methods. Unsupervised methods do not need labelled image data (Wang et al. 2021, Yu et al. 2021, Zhang et al. 2021). The ground truth of the data is the image data itself. This is a significant advantage for learning-based and unsupervised hashing methods. However, their efficiency is generally lower than the supervised methods (Patel and Kasat 2017). On the other hand, semisupervised methods require labelling information partially (Tang et al. 2019, Shi et al. 2020, Tian et al. 2020).

We have proposed an unsupervised method in this study. It consists of an encoder-decoder model. Hash codes are generated by transforming image features extracted from an intermediate layer between the encoder and the decoder blocks. Encode-decoder models are also called autoencoders. An encoder block encodes high-dimensional image features into low-dimensional features, also called intermediate features. The decoder block reconstructs images by using those features. A loss function should compute the error between the original and rebuilt image features at the decoder's end. The output of that loss is used for the back-propagation during the training phase. The typical loss functions used at the end of the decoder models are mean square error (*MSE*) (Int. Ref. 1), also called L2 loss, or binary cross entropy loss (Int. Ref. 2).

The most common usage area of autoencoders is image denoising (Keerthi et al. 2021, Qiu et al. 2020) and image segmentation (Myronenko A. 2019, Baur et al. 2021). In the proposed method, we used a part of the Resnet50 pre-trained network (Int. Ref. 3). Since it is trained on the ImageNet dataset (Int. Ref. 4) in a supervised way, it generates efficient features for the decoder side. That is, convergency becomes fast during training. This is a significant idea for implementing transfer learning from a supervised domain to an unsupervised one. The highlights of this study can be summarized in the following.

- The proposed method is unsupervised, independent of the training phase's image label. However, efficient supervised features are transferred and then trained unsupervised. This is a significant domain adaptation idea.
- A loss function in an intermediate layer between encoder and decoder block optimizes hash codes and balances the distribution of binary values in the hash codes.
- Experiments performed on MS Coco (Int. Ref. 5) multilabel benchmark and Retinamnist (Int. Ref. 6) datasets demonstrate the competitive results in the unsupervised image hashing category.

The rest of the study consists of the following chapters. The detail of the proposed method, hash code extraction, and retrieval are presented in section II. Performance metrics and datasets used for the experiments are given in section III. Section IV provides experimental results performed on two image datasets. Finally, section V is the result and discussion section.

2. Material and Method

The proposed method mainly consists of training and retrieval phases. In the training phase, whole layers are trained, including ResNet50. In the retrieval phase, a prediction is performed, and image features are extracted from the hash layer.

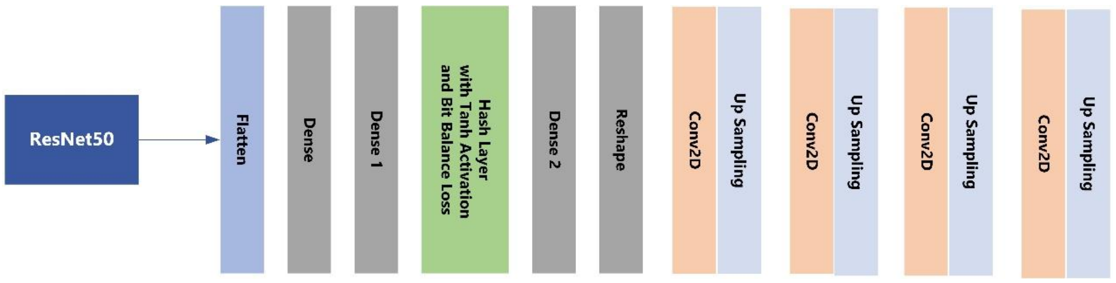


Figure 1. An overview of the proposed method. Full architecture is trained in the training phase.

Those predictions are consist of the real number. Features are then converted into hash codes by a quantization process.

2.1 Training phase

The training phase consists of encoder-decoder blocks and custom layers placed between those blocks for the hash code extraction. An overview of the training phase is illustrated in Figure 1.

The encoder block is constructed by ResNet50. It is trained for the ImagetNet dataset having 1000 categories. Resnet has residual connections that reduce the vanishing gradient descent effect and increase classification accuracy. We only removed the last fully connected (FC) layers and the classification layer from ResNet50. In that way, ResNet50 functions like an encoder. Using ResNet50 as an encoder is a wise idea that provides domain adaptation between supervised and unsupervised domains.

The standard output shape of the ResNet50 is (8,8,2048). A flattened layer is added next to it to convert that shape into a one-dimensional tensor. An encoder should encode high-dimensional inputs into low-dimensional data for compression. Hence, we used two low-dimensional FC layers next to the flattening layer. Their sizes are 2048 and 256, respectively.

Those sizes provide a soft transition for dimension reduction. We prepared a custom layer called the hash layer placed next to those FC layers. The hash layer has a tangent hyperbolic (*tanh*) activation. Given an input *x, tanh* is defined as follows,

$$tanh(x) = \frac{Sinh(x)}{Cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (1)$$

Tanh is a scaled version of the sigmoid function. Its depiction is given in Figure 2. Its mean is zero. That provides fast convergence. Figure 2 shows that outputs are constrained in the (-,1, 1) interval, whatever its input. This property is necessary for converting image features into hash (binary) codes in the quantization process in the retrieval phase.

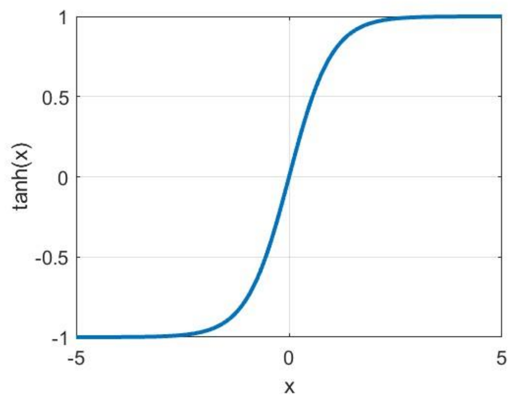


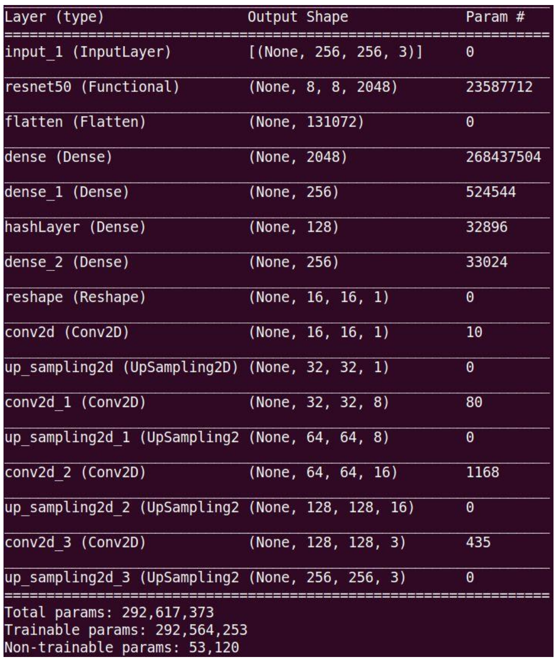
Figure 2. An illustration of the *tanh* function.

Hamming distance in image hashing methods is indispensable since XOR operation is high-speed, which offers fast retrieval. Hamming distance between two binary codes is the number of different reciprocal bits. An ideal image hashing algorithm should be unbiased for the number of zeros and ones. In other words, the number of zeros and ones should be close. This property is called the bit-balance property. Bit-balance has a pleasant characteristic for providing unbiased Hamming distances. We have defined a loss function over the hash layer that provides bit-balance property.

Let K $\boldsymbol{X}^{KxL} \in R$ represent the hash layer's input data, where *K* is the number of outputs of the FC layer before the hash layer, and *L* represents the number of inputs of the hash layer. Since the activation function of the hash layer is *tanh,* its output data is $\boldsymbol{Y} = tanh(\boldsymbol{X})$. Based on this information, a bit-balance loss is defined as follows,

$$Loss_1 = \sum_{i=1}^{L} Y_i > 0 \tag{2}$$

In equation 2, *Y* varies in the (-1,1) interval. Minimizing the loss function during training will force the mean value of the outputs (*Y*) to be zero.



```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 3)]     0
_________________________________________________________________
resnet50 (Functional)        (None, 8, 8, 2048)        23587712
_________________________________________________________________
flatten (Flatten)            (None, 131072)            0
_________________________________________________________________
dense (Dense)                (None, 2048)              268437504
_________________________________________________________________
dense_1 (Dense)              (None, 256)               524544
_________________________________________________________________
hashLayer (Dense)            (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 256)               33024
_________________________________________________________________
reshape (Reshape)            (None, 16, 16, 1)         0
_________________________________________________________________
conv2d (Conv2D)              (None, 16, 16, 1)         10
_________________________________________________________________
up_sampling2d (UpSampling2D) (None, 32, 32, 1)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 8)         80
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 64, 64, 8)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 16)        1168
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 128, 128, 16)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 128, 128, 3)       435
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 256, 256, 3)       0
=================================================================
Total params: 292,617,373
Trainable params: 292,564,253
Non-trainable params: 53,120
```

Figure 3. The layer shapes of the proposed method.

The loss in equation 2 is minimized with the regression loss, defined at the end of the decoder during the training phase. That provides bit-balance property. The Dense layer next to the hash gives the decoder's output identical to the original image sizes.

The decoder model starts with a reshape layer. This layer converts one-dimensional image features into two-dimensional image-like tensors. The rest of the layers consist of convolution and upsampling layers. Upsampling layers gradually increase the size of tensors. Finally, the last upsampling layer reconstructs images as identical size to the original images. The hierarchical structure and the shape of the layers are presented in Figure 3.

A regression loss is placed at the decoder's end to reduce the error between original and reconstructed images. This loss is called the mean square error (*MSE*) in literature (Wang Z et al. 2009).

Let *N* represent the number of images, $\boldsymbol{Z}_{true}$ represent the input images, and $\boldsymbol{Z}_{pred}$ be the output images taken from the output of the decoder. *MSE* error between them is given by,

$$Loss_2 = \frac{1}{N}\sum_{i=1}^{N}(Z_{true_i} - Z_{pred_i})^2 \tag{3}$$

The overall loss function to be minimized during the training phase is as follows,

$$Loss = \alpha Loss_1 + \beta Loss_2 \tag{4}$$

We investigate the optimum α and β weights by assigning values between zero and one. However, we did not observe any impact of those weights on the loss value and retrieval result. For this reason, we assign identical values to each loss.

2.2 Retrieval phase

Hash codes are generated after the network is fully trained. Then a snapshot of the weights and the model are taken. As seen in Figure 4, We do not use the loss functions and decoder module. They are only used in the training phase to optimize the network parameters. Extraction is issued by performing predictions from the hash layer and quantizing them. Since the activation function of the hash layer is a *tanh*, predictions are between (-1, 1) intervals.

Suppose that the prediction taken from the hash layer is represented by $\boldsymbol{y}_{pred}$. Image hash codes can be extracted by the following quantization process,

$$\boldsymbol{H} = \boldsymbol{y}_{pred} > 0. \tag{5}$$

As seen in Figure 4, retrieval can now be performed by finding Hamming distance between a query and the whole dataset items. We chose the number of outputs of the hash layer as 32,48, 64 for the Retinamnist dataset and 64,128,256 for the Coco dataset. That is, corresponding hash codes are generated.

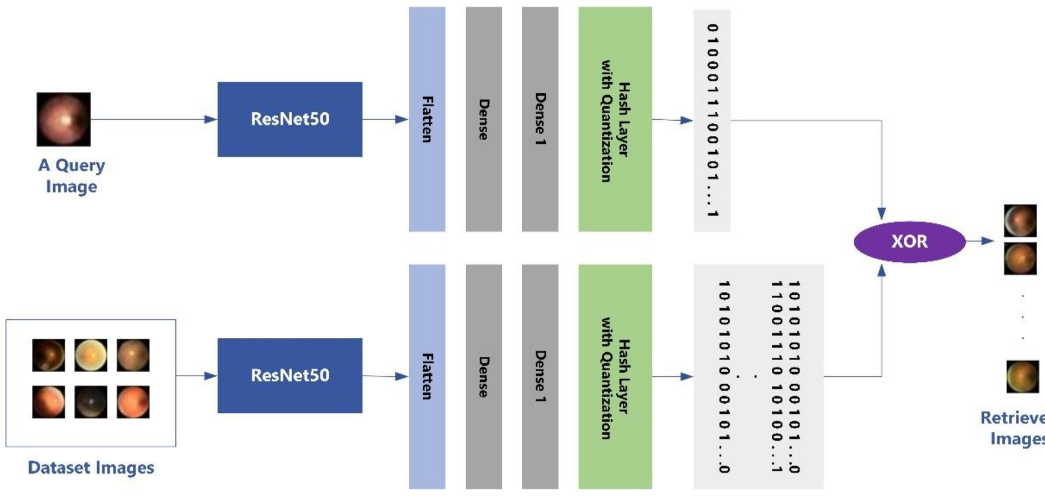


Figure 4. The retrieval phase of the proposed method.

3. Performance Metrics and Datasets

3.1 Performance metrics

Precision (*P*), Average Precision (*AP*), and Mean Average Precision (*MAP*) scores are used for the evaluations. They are calculated as follows. A retrieved image is evaluated as true-positive (*tp*), relevant, if it holds all the labels presented in the query; Otherwise, false negative (*fn*), irrelevant.

The precision scores for *k* retrieved images are obtained as follows.

$$P@k = \frac{tp@k}{tp@k+fp@k} \tag{6}$$

where *tp@k* denotes the number of relevant items among retrieved items, and *fp@k* denotes irrelevant items. Then, an *AP* score for each query set is obtained as follows.

$$AP = \frac{1}{R_k} \sum P_s \, \alpha_s \tag{7}$$

where R_k is the number of relevant items among k retrieved items, P_s is a precision score for each item retrieved, and α_s is an indicator factor. It equals one if the *r-th* retrieved sample is *tp*. Otherwise, it is set to zero.

MAP scores are obtained by calculating the mean values of the average precision scores obtained for top k returned images for a single query. For Q queries, a MAP score is obtained as follows,

$$MAP@Q = \frac{1}{Q}\sum_{i=1}^{Q}(AP)_i \tag{8}$$

3.2 Datasets

Coco 2017 (Int. Ref. 5) is a multi-label image recognition dataset. There are 80 categories in the dataset. Each image contains at least one class. 78220 images are used as the training set and 5000 for the test set.

The Retinamnist (Int. Ref. 6) consists of 1600 images. An image presents one of the 5-level gradings of diabetic severity. The size of images is 28x28 RGB. There are 1080 train images, 120 validation and 400 test.

4.Experimental Results

The proposed encoder-decoder model has been implemented in Tensorflow's Keras library (Int. Ref. 7). The training part of the datasets is used during training. Since this is an unsupervised method, the ground truths are also the training images. The network is trained by Adam optimizer (Int. Ref. 8) with 100 epochs. The learning rate is set to 0.01, and the batch size to 32. The training phase is repeated for the various size of the hash layer, such as 16, 32, 64, 128, and 256. During training, *MSE* loss reduced the error between the ground truths and predicted images. Another loss defined over the hash layer provides bit-balance property.

Loading the whole dataset into the memory consumes the computer's resources. If you use a large-scale dataset, this is inevitable. To overcome that issue, we prepared custom Python generators. Although there are ready to use Keras generators, we designed our own. This provides us with more control, customizations, and easy error detection. Thanks to the Python generators, only the current batches of images are loaded into the memory during training.

Hash codes should be extracted via the best-trained sample of the network. So after training, the snapshot of the network model and network parameters are saved. Each time snapshots are taken for the various sizes of the hash layer. Hash codes are extracted by performing predictions using train and test images. The location of the prediction is the hash layer.

Retrieval metrics used for the top returned images in Hamming radius ***r***. A Hamming distance vector is created between a query and the whole items in the dataset. What we mean by ***r*** is that only those returned images retrieved whose Hamming distance to the query is less or equal to the Hamming radius ***r***. For evaluations, we randomly chose 500 queries from the test split of the Coco and 150 queries from Retinamnist datasets.

Suppose that 20 images are retrieved. For each retrieved image, a precision score, and for 20 images, an average precision score is calculated. For the whole query images, a mean average precision score is obtained. Evaluations are repeated for 16, 32, and 64-bit hash codes for Retinamnis; for Coco, it is 64,128 and 256-bit. We use different hash code sizes for datasets because of varying dataset sizes. High-dimensional hash codes should always be used in large-scale datasets. The map scores for those hash codes and various Hamming radii are presented in Tables 1-2. There is no recent hashing study using the Retinamnist dataset. However, the most recent unsupervised studies using Coco dataset compared with the proposed method and presented in Table 3.

Table 1. MAP scores of top retrieved images in specific Hamming radius. Results are obtained by using Coco dataset.

Size (Bits)	map@5	map@10	map@15	map@20
64	0.7454	0.7444	0.7406	0.7364
128	0.8351	0.8306	0.8254	0.8103
256	0.8417	0.8402	0.8369	0.8323

Table 2. MAP scores of top retrieved images in specific Hamming radius. Results are obtained by using the Retinamnist dataset.

Size (Bits)	map@5	map@10	map@15	map@20
16	0.4250	0.4071	0.3802	0.3671
32	0.7415	0.6443	0.5422	0.4602
64	0.8360	0.7576	0.6982	0.6354

Table 3. Comparison of the proposed method with the most recent studies using the Coco dataset (the mAP score for different hash code lengths are reported).

Size (Bits)	Yu, Y. et al. 2021	Zhang, X. et al. 2021	Zhang H. et al. 2020	Ours
16	0.507	0.745	0.661	**0.514**
32	0.630	0.753	0.694	**0.685**
64	0.689	0.763	0.712	**0.734**

As seen in Table 3, the results are competitive. The related codes of this study, such as training, hash code extractions, hash codes, and evaluations, are publicly available (Int. Ref. 9).

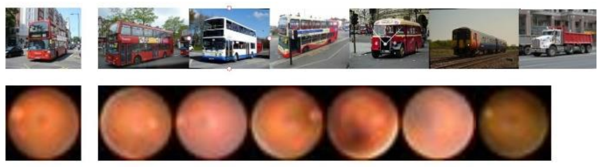


Figure 5. Retrieval results are obtained from Coco and Retinamnist datasets, respectively. The left column presents query images. The query image from Retinamnist is a level-1 diabetic severity.

5 Result and Discussion

Supervised image hashing methods generally outperform unsupervised and semisupervised methods in terms of retrieval efficiency. However, supervised methods use label information which is an extra job. In this study, we propose a solution for improving the retrieval efficiency of unsupervised image hashing. The novel idea is to implement a domain adaptation between the supervised and unsupervised domains. ResNet50, a known supervised pre-trained network, is used as an encoder model. However, thanks to the decoder model, images are trained without label information. This adaption provides fast convergence and efficient hash codes. Experiment results confirm that idea. As an example, two retrieval results performed for both datasets are presented in Figure 5. As can be seen, the retrieval results are visually similar to the query to a large extent.

The proposed method has transferred the accuracy and fast convergence of supervised methods to unsupervised ones. This way, supervised-level results are obtained without using label information.

The idea obtained from that study can apply to video hashing. We plan to prepare the same or similar domain adaptation approach for video hashing.

Acknowledgements:
The numerical calculations reported in this paper were fully performed at Bursa Technical University High-Performance Computing Lab.

References

Akalın, B. and Veranyurt, Ü., 2022. Sağlık 4. O ve Sağlıkta Yapay Zekâ. *Sağlık Profesyonelleri Araştırma Dergisi*, ***4*****(1)**, 57-64.

Aslan, F. and Subaşı, A., 2022. Hemşirelik Eğitimi ve Hemşirelik Süreci Perspektifinden Yapay Zeka Teknolojilerine Farklı Bir Bakış. *Sağlık Bilimleri Üniversitesi Hemşirelik Dergisi*, ***4*****(3)**, 153-158.

Baduge, S.K., Thilakarathna, S., Perera, J.S., Arashpour, M., Sharafi, P., Teodosio, B., Shringi, A. and Mendis, P., 2022. Artificial intelligence and smart vision for building and construction 4.0: Machine and deep learning methods and applications. *Automation in Construction*, ***141***, 104440.

Baur, C., Denner, S., Wiestler, B., Navab, N. and Albarqouni, S., 2021. Autoencoders for unsupervised anomaly segmentation in brain MR images: a comparative study. *Medical Image Analysis*, ***69***, 101952.

Keerthi Nayani, A.S., Sekhar, C., Srinivasa Rao, M. and Venkata Rao, K., 2021. Enhancing image resolution and denoising using autoencoder. In *Data Analytics and Management: Proceedings of ICDAM* (649-659). Springer Singapore.

Mchergui, A., Moulahi, T. and Zeadally, S., 2022. Survey on artificial intelligence (AI) techniques for vehicular ad-hoc networks (VANETs). *Vehicular Communications*, ***34***, 100403.

Minh, D., Wang, H.X., Li, Y.F. and Nguyen, T.N., 2022. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, 1-66.

Mutlu, İ.N., Koçak, B., Kuş, E.A., Ulusan, M.B. and Kılıçkesmez, Ö., 2021. Machine Learning-Based Computed Tomography Texture Analysis of Lytic Bone Lesions Needing Biopsy: A Preliminary Study. *Istanbul Medical Journal*, *22*(3).

Myronenko, A., 2019. 3D MRI brain tumor segmentation using autoencoder regularization. In Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part II 4 (311-320). Springer International Publishing.

Nahavandi, D., Alizadehsani, R., Khosravi, A. and Acharya, U.R., 2022. Application of artificial intelligence in wearable devices: Opportunities and challenges. *Computer Methods and Programs in Biomedicine*, ***213***, 106541.

Patel, F.S. and Kasat, D., 2017, February. Hashing based indexing techniques for content based image retrieval: A survey. In *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)* (279-283). IEEE.

Shi, X., Guo, Z., Xing, F., Liang, Y. and Yang, L., 2020. Anchor-based self-ensembling for semi-supervised deep pairwise hashing. *International Journal of Computer Vision*, ***128***, 2307-2324.

Singh, A. and Gupta, S., 2022. Learning to hash: A comprehensive survey of deep learning-based hashing methods. *Knowledge and Information Systems*, ***64*****(10)**, 2565-2597.

Şendir, M., Şimşekoğlu, N., Abdulsamed, K.A.Y.A. and SÜMER, K., 2019. Geleceğin teknolojisinde hemşirelik. *Sağlık Bilimleri Üniversitesi Hemşirelik Dergisi*, ***1*****(3),** 209-214.

Tang, X., Liu, C., Zhang, X., Ma, J., Jiao, C. and Jiao, L., 2019, July. Remote sensing image retrieval based on semi-supervised deep hashing learning. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium* (879-882). IEEE.

Tian, X., Zhou, X., Ng, W.W., Li, J. and Wang, H., 2020. Bootstrap dual complementary hashing with semi-supervised re-ranking for image retrieval. *Neurocomputing*, ***379***, 103-116.

Yu, Q., 2020. Improved denoising autoencoder for maritime image denoising and semantic segmentation of USV (J). *China Communications*, ***17*****(3)**, 46-57.

Wang, J., Liu, W., Kumar, S. and Chang, S.F., 2015. Learning to hash for indexing big data—A survey. *Proceedings of the IEEE*, ***104*****(1)**, 34-57.

Wang, Y., Song, J., Zhou, K. and Liu, Y., 2021. Unsupervised deep hashing with node representation for image retrieval. *Pattern Recognition*, ***112***, 107785.

Wang, Z. and Bovik, A.C., 2009. Mean squared error: Love it or leave it? A new look at signal fidelity measures. *IEEE signal processing magazine*, ***26*****(1)**, 98-117.

Yu, Y., Yang, L. and Wang, S., 2021, November. Deep hash image retrieval method based on anti-autoencoder. In *2021 7th International Conference on Systems and Informatics (ICSAI)* (1-5). IEEE.

Zhang, H., Gu, Y., Yao, Y., Zhang, Z., Liu, L., Zhang, J. and Shao, L., 2020. Deep unsupervised self-evolutionary hashing for image retrieval. *IEEE Transactions on Multimedia*, ***23***, 3400-3413.

Zhang, X., Wang, X. and Cheng, P., 2021. Contrast-based unsupervised hashing learning with multi-hashcode. *IEEE Signal Processing Letters*, ***29***, 219-223.

References of Internet

1. https://en.wikipedia.org/wiki/Mean_squared_error, (13.10.2023)
2. https://neptune.ai/blog/cross-entropy-loss-and-its-applications-in-deep-learning, (21.10.2022)
3. https://viso.ai/deep-learning/resnet-residual-neural-network, (03.11.2022)
4. https://www.image-net.org, (11.11.2022)
5. https://www.kaggle.com/datasets/awsaf49/coco-2017-dataset, (18.08.2022)
6. https://medmnist.com, (18.08.2022)
7. https://keras.io, (09.11.2022)
8. https://keras.io/api/optimizers/adam, (09.11.2022)
9. https://github.com/enverakbacak/AE, (11.01.2023)