MEASEURE EFFICIENCY OF IMAGES OBTAINED THROUGH AUTOENCODERS

Advanced Machine Learning

Master degree in Data Science
Faculty Science of University of Lisbon

Ana Araújo nº 59457
Pedro Almeida nº 58844

1. Introduction

Images are rich sources of information and have become increasingly prevalent in various domains, ranging from healthcare and autonomous vehicles to social media and e-commerce. Extracting meaningful features from images is crucial for effective machine learning algorithms to understand and make accurate predictions or decisions. However, using images in machine learning poses unique challenges due to their high dimensionality, variability, and complex structures. In this introduction, we will discuss some of the key problems encountered when working with images in machine learning and provide references to relevant literature addressing these challenges.

1.1 High Dimensionality

Images typically consist of a large number of pixels, resulting in high-dimensional input data. The high dimensionality poses challenges in terms of computational complexity, memory requirements, and the curse of dimensionality, where the sparsity of data becomes a hindrance. Dimensionality reduction techniques, such as principal component analysis (PCA) or convolutional neural networks (CNNs), have been widely employed to mitigate this issue and extract relevant features effectively [1][2].

1.2 Advantages of Using Autoencoders to Reduce Dimensionality:

Autoencoders, a type of unsupervised learning neural network, offer several advantages when it comes to reducing dimensionality in machine learning:

- Nonlinear Mapping: Autoencoders can capture complex and nonlinear relationships within high-dimensional data. Unlike linear dimensionality reduction techniques such as PCA, autoencoders can learn intricate patterns and representations in the data, enabling more accurate and expressive feature extraction [1].
- Preserving Information: Autoencoders aim to reconstruct the input data at the output layer, ensuring that the most relevant and informative features are retained. By reconstructing the input, autoencoders inherently preserve important characteristics and structures of the data, leading to a more faithful representation of the original information [2]
- Unsupervised Learning: Autoencoders do not require labeled data for training, making them suitable for unsupervised learning tasks. This is particularly advantageous when labeled data is limited or expensive to obtain. By learning from unlabeled data, autoencoders can extract useful features and reduce dimensionality in a data-driven manner [3].
- Robustness to Noise: Autoencoders can handle noisy or incomplete data effectively. The reconstruction process in autoencoders encourages the learning of robust representations that are less sensitive to noise and missing values. This property makes autoencoders well-suited for dimensionality reduction in real-world scenarios where data quality may be compromised [4].
- Adaptability to Various Data Types: Autoencoders can be applied to different types of data, including images, text, and time series. They are flexible in capturing the inherent structures and patterns within diverse data modalities, making them applicable in various domains such as computer vision, natural language processing, and sensor data analysis [5].

Autoencoders are commonly trained using element-wise loss. However, element-wise loss disregards high-level structures in the image which can lead to embeddings that disregard them as well. [6]

1.3 Problem: Measure efficiency of images obtained through autoencoder

With numerous applications, from autonomous driving to facial recognition, image classification is crucial in computer vision. In this project, we want to create a precise image classification model that can distinguish between pictures of animals. Each animal category's thousands of labeled images make up the dataset. To train and assess our model, we will use machine learning techniques that we have studied in this course.
Initially, we consider two datasets from Kaggle: cats and dogs[1] and cheetahs and lions[2] for image classification between cats, dogs, lions and cheetahs. However, the dataset Cheetahs and lions present a lot of biodiversity images and few images that contain either lions or cheetahs, leading to unlabeled images. Moreover, the lack of image quality in this dataset leads to a higher baseline noise, making it difficult to classify this data between cheetahs and lions. Thus, for this project, it was only considered the dataset of cats and dogs.

[1]https://www.kaggle.com/datasets/samuelcortinhas/cats-and-dogs-image-classification

[2] https://www.kaggle.com/datasets/mikoajfish99/lions-or-cheetahs-image-classification

The problem statement is to measure the efficiency of images obtained through autoencoders that classify between a cat and a dog in an image as accurately as possible.

1.2.1 The dataset:
The datasets is from Kaggle data base. The dataset is a compilation of over a 1000 images of cats and dogs scraped off of Google images. The dataset contains both training and test sets, and the images are different sizes and resolutions. In order to extract features from the images and train a classifier to categorize them into one of the four animal categories. The image sizes range from 100x100 pixels to 2000x1000 pixels.

2. Approach Implementation

Our approach is to split the dataset into three parts, training, testing and validation, where the training part of this dataset is used to train the autoencoder and the classifiers to classify the images in one of the two animal categories. The model classifiers chosen are: Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM) and K-Nearest Neighbors (K-NN). The test part is used mostly to fine tune model's hyperparameters and the autoencoder validation and to also make some representations.
The reducing images with autoencoders will be applied at different levels: 50%, 75% and 87.5%. For each classifier it will be evaluated the accuracy, recall, Precision, F1 score and the running time. In addition, it will also performed the evaluation of each classifier with the resizing of images at the same levels for comparison and to study the impact of autoencoders.

3. Implementation

3.1 Data Preparation:

We concatenated each animal's data sets and shuffled them. We preprocessed the images by resizing them to a fixed size of 64 x 64 and assuring that all images respected the rules, for example having a shape equal to (64,64,3).

3.2 Data Splitting:

The dataset was splitted between training, test and validation sets and the pixel values were normalized to be between 0 and 1. Regarding splitting: the training set has 2364 images, the test set presents 1246 images and the validation set presents 581 images.

3.3 Define the baseline:

To create a baseline for comparison with the experiments that will be conducted later, we trained four classifiers—Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), and K-Nearest Neighbors (K-NN)—to classify the images into one of the two animal categories. In addition to the classifiers, we conducted a brief grid search to identify the more appropriate hyperparameters. The evaluation metrics and the time ideal hyperparameters for each classifier are shown in Table 1 in annex A. Additionally, it was also measuring the elapsed time for each classifier, in order to get an idea of the time taken by each classifier for training and making predictions on the given dataset.

3.3 Model Training with autoencoders

First, we specify the image's input shape as (64, 64 3), which represents an image with three RGB color channels and dimensions of 64 pixels in height and width. The encoder and decoder in the model cooperate to compress and reconstruct the input image.

Max-pooling layers are defined after convolutional layers in the definition of the encoder layers. While extracting more complex information, these layers gradually shrink the input image's spatial dimensions.

The layers of the encoder and decoder are identical; however they are arranged differently. The encoded representation is transformed back into a format that is appropriate for upsampling by dense and reshape layers.
Deconvolutional layers, sometimes referred to as upsampling layers, are used to expand the feature maps' spatial range. They aid in reassembling the original image from the encoder's compressed representation. By moving filters over the image and recording various patterns at each place, convolutional layers evaluate it. They give the model the ability to recognize and extract useful elements from the input image, assisting it

in comprehending the visual information and generating predictions based on that material. The image is then gradually rebuilt using convolutional layers and upsampling layers. [7, 8]

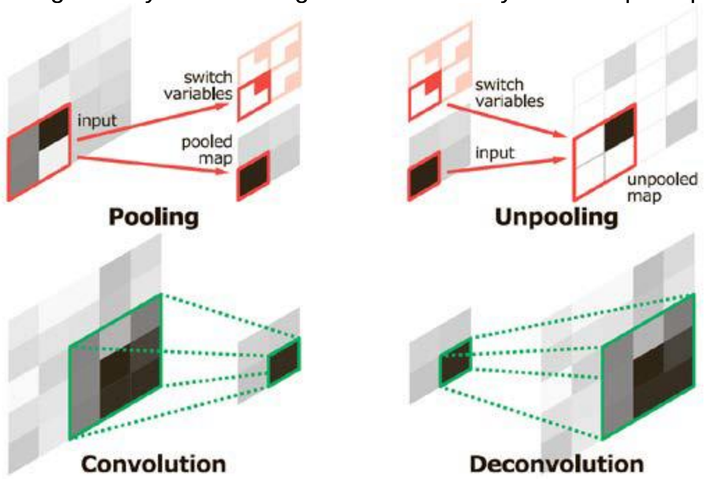

Fig. 1 Illustration of deconvolution and unpooling layers [8]

The image is rebuilt using a 3x3 convolution with three output channels in the decoder's final layer. The model's non-linearity was added using the ReLU activation function.

The rmsprop optimizer and the mean squared error (MSE) loss function are used to create the autoencoder model. It modifies these parameters to optimize the model's performance and reduce the mean squared error, which is the prescribed loss function. [9]

The MSE loss drives the model to accurately rebuild the image by measuring the pixel-wise difference between the input and output images. Since the discrepancies are squared, it penalizes bigger differences more severely. [10]

The training data are both the input and the target for reconstruction as the model is developed. The model is trained using a batch size of 1 across 30 epochs. The training data was shuffled. During training, the performance of the model is checked using the validation data parameter.

The trained autoencoder model is then used to encode a set of randomly chosen test images, which are later decoded to produce the reconstructed image.

In this project we trained 3 autoencoder models with different size reductions of the images: 50% (32 x 32), 75% (16x16) and 87.5% (8x8). In addition, we also reduced the original images to the same reduction levels for a comparison between encoded image against the resizing the images at the same level reduction.

4. Model evaluation and Results

The results are shown in Table 1 of Annex A.

4.1 Classifiers Models with Original Images:

Regarding the outcomes produced using the original images, the DT achieved a relatively low accuracy of 58.1%. It shows that using the original images as features did not produce effective prediction results. With accuracy rates of 63.6% and 61.6%, respectively, RF and SVM performed slightly better. K-NN had the lowest accuracy (55.5%), indicating that using just the image's attributes may not be enough to effectively classify data using nearest-neighbour methods.

4.2 Encoded images vs resized images at 50%:

DT achieved an accuracy of 58.3% with a recall of 48.9%, precision of 58.9%, and F1 score of 53.5% using encoded images with a 50% decrease. Over the original images, there is just a slight performance increase.

With a 69.0% accuracy rate, RF significantly outperformed DT. It suggests that the autoencoder's encoded characteristics have greater predictive value for the RF model. SVM's accuracy score of 69.9% is comparable to RF's, showing that it can use encoded information for classification well. K-NN achieved a 58.2% accuracy rate, which is on par with DT. It implies that the K-NN algorithm did not greatly benefit from the reduction in feature space.

When using the downsized images, the accuracy for DT and RF fell to 54.7% and 54.7%, respectively. It suggests that essential visual information may be lost if images are reduced to 32x32 pixels. SVM outperformed DT and RF with an accuracy of 60.1%, a small improvement. It implies that, with just modest accuracy gains, SVM can still use the scaled images for classification.

K-NN outperformed DT and RF with reduced images, achieving an accuracy of 59.0%. It implies that K-NN is comparatively more resistant to the loss of visual data.

4.3 Encoded images vs resized images at 75%:

DT was able to achieve an accuracy of 60.6% with encoded images that had been reduced by 75%, demonstrating a modest increase over the 50% reduction case. The F1 score, recall, and precision all increased.
The accuracy of RF was 69.7%, which is comparable to the 50% reduction case. It implies that the performance of the RF model was not considerably affected by the additional reduction.
The autoencoder's encoded features were informative for SVM-based classification, as shown by the maximum accuracy for SVM of 71.4%.
K-NN's accuracy of 58.2% was in line with its performance in the case of a 50% reduction.
The accuracy for the 75% reduced images further decreased to 56.1% for DT and 56.1% for RF. It implies that a greater loss of crucial visual information was caused by a reduction in feature space.
SVM had a 62.5% accuracy rate, showing that it can still classify the scaled images even with only modest accuracy improvements.
K-NN's accuracy of 58.5% is a marginal improvement over the case of 50% decrease.

4.4 **Encoded images vs resized images at 87.5%:**
DT was able to reach an accuracy of 61.8% with encoded images that had an accuracy of 87.5%, which represents a minor improvement over the case of 75% reduction. The F1 score, recall, and precision all increased.
The accuracy of RF was 68.3%, which is comparable to the situation of a 75% reduction. It implies that the performance of the RF model was not considerably affected by the additional reduction.
The greatest accuracy for SVM was 71.4%, showing that the encoded features are still useful for SVM-based classification.
K-NN's accuracy of 58.2% was comparable to its performance in the earlier reduction scenarios.
The accuracy for the 87.5% resized images dropped to 56.8% for DT and 56.8% for RF, which is similar with the trend seen in earlier reduction scenarios.
SVM had a 56.9% accuracy rate, which shows that the scaled images are barely useful for SVM-based categorization.
K-NN's accuracy of 58.5% represents a slight gain over the case of a 75% reduction.

The four classifiers performed differently when dimensionality reduction was accomplished using an autoencoder. While DT and K-NN demonstrated only modest improvements from the encoded features, RF and SVM consistently benefited. Resized images produced less accurate results as compared to encoded data, emphasizing the value of keeping visual information. SVM regularly outperformed the other examined models in terms of accuracy, indicating how well it can use encoded features. It's crucial to remember that the performance improvements came at the expense of longer computation times, as evidenced by how long it took to fit the models.

5. Final comments:

As previously stated, the research did not evaluate the lion and cheetah dataset because to its poor quality. Nevertheless, given these creatures resemble cats in certain ways, it would be fascinating to include a data set with them.
In this analysis, we explored the performance of different classifiers using original images, encoded features obtained from an autoencoder with different reduction percentages, and resized images. Here are some final comments:

- **Impact of Autoencoder and Dimensionality Reduction:**

  The results of dimensionality reduction using an autoencoder were inconsistent. But the overall trend was that encoded data performed better than the original and the resized, in terms of metrics results at least for the same proportions.

  There was some improvement in the classification metrics when reducting 50%, 75%, and 87.5% of the original data size, compared to its counterparts the encoded data performed generally better but with a significant trade-off between performance and time, this increase in the metrics was made at cost of time.

- **Performance with Resized Images:**

  For most classifiers, resizing the images to a lower size (32x32 pixels) decreased their accuracy. This shows that crucial visual information required for precise categorization was lost when the images were reduced and there was the advantage of using an autoencoder for dimensionality reduction, the critical information tend to be kept.

- **Superiority of SVM classifier:**

  Among the studied models, SVM consistently had the highest accuracy levels. This shows that, in comparison to other classifiers, SVM was better able to make use of the encoded attributes or reduced pictures for successful classification.

- **Time Complexity:**

  It is important to note that the amount of time needed for fitting the models was raised by the inclusion of an autoencoder and bigger feature spaces (encoded features). When deciding on a strategy for image classification, it is important to consider the trade-off between accuracy and processing time.

Achieving high accuracy requires making the appropriate classifier choice. SVM consistently outperformed other classifiers in our investigation, indicating its applicability to the task of classifying images. In conclusion, further experimentation, and fine-tuning of the models and their hyperparameters could potentially lead to better results. Additionally, exploring other feature engineering techniques, such as extracting handcrafted features or using pre-trained models, might provide alternative approaches to improve the accuracy of image classification tasks.

In conclusion, more investigation of the models and their hyperparameters may produce better outcomes. Additionally, investigating other feature engineering methods, such using models that have already been trained, may offer different solutions to enhance the accuracy of image classification tasks.

6. References:

[1] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

[2] Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In Advances in neural information processing systems (pp. 153-160).

[3] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, 11(Dec), 3371-3408.

[4] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning (pp. 1096-1103).

[5] Baldi, P., & Sadowski, P. (2013). Understanding dropout. In Advances in neural information processing systems (pp. 2814-2822).

[6] Pihlgren, G. G., Sandin, F., & Liwicki, M. (2020). Improving Image Autoencoder Embeddings with Perceptual Loss. *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–7. https://doi.org/10.1109/IJCNN48605.2020.9207431

[7] Turchenko, V., Chalmers, E., & Luczak, A. (2017). A deep convolutional auto-encoder with pooling-unpooling layers in caffe. *arXiv preprint arXiv:1701.04949*.

[8] Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision* (pp. 1520-1528).

[9] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[10] Q. Zhu, H. Wang and R. Zhang, "Wavelet Loss Function for Auto-Encoder," in IEEE Access, vol. 9, pp. 27101-27108, 2021, doi: 10.1109/ACCESS.2021.3058604.

Annexes - A

Table 1. *Accuracy results and the best hyperparameters achieved for each model classifier.*

Classifier	Accuracy	Best Parameters	Time (s)
DT	0.581059	max_depth = 7	126.09

RF	0.636437	max_depth = None
		n_estimators = 200
			708.06

SVM	0.616372	C = 10
		Gamma = 0.001
			932.12

k-NN	0.555377	n_neighbors = 7	5.22

Table 2. *Model evaluation for each model classifier.*

Method	Classifier	Accuracy	Recall	Precision	F1 score	Time (s)
Encoded Images with 50% Reduction	**DT**	0.583477	0.489437	0.588983	0.534615	163.52
	RF	0.690189	0.721831	0.669935	0.694915	94.25
	SVM	0.698795	0.707746	0.686007	0.696707	307.64
	K-NN	0.581756	0.619718	0.565916	0.591597	10.44
Resize Images to 50%	**DT**	0.547332	0.654930	0.529915	0.585827	**20.93**
	RF	0.586919	0.588028	0.575862	0.58188	37.24
	SVM	0.600688	0.750000	0.569519	0.647416	36.64
	K-NN	0.590361	0.718310	0.563536	0.631579	0.78
Encoded Images with 75% Reduction	**DT**	0.605852	0.598592	0.596491	0.597540	83.20
	RF	0.697074	0.693662	0.688811	0.691228	64.09
	SVM	0.714286	0.707746	0.707746	0.707746	144.98
	K-NN	0.581756	0.602113	0.568106	0.584615	4.17
Resize Images to 75%	**DT**	0.605852	0.598592	0.596491	0.597540	83.20
	RF	0.624785	0.640845	0.610738	0.625430	16.32
	SVM	0.569707	0.813380	0.539720	0.648876	6.03
	K-NN	0.585198	0.735915	0.557333	0.634294	0.33
Encoded Images with 87.5% Reduction	**DT**	0.617900	0.598592	0.611511	0.604982	83.49
	RF	0.683305	0.679577	0.674825	0.677193	63.70
	SVM	0.714286	0.707746	0.707746	0.707746	151.85
	K-NN	0.581756	0.602113	0.568106	0.584615	4.65
Resize Images to 87.5%	**DT**	0.567986	0.616197	0.552050	0.582363	4.38
	RF	0.602410	0.595070	0.592982	0.594025	16.30
	SVM	0.569707	0.813380	0.539720	0.648876	6.31
	K-NN	0.585198	0.735915	0.557333	0.634294	1.54