An empirical study on the effects of different types of noise in image classification tasks

Gabriel B. Paranhos da Costa, Welinton A. Contato, Tiago S. Nazare, João E. S. Batista Neto, Moacir Ponti
Instituto de Ciências Matemáticas e de Computação (ICMC) – Universidade de São Paulo (USP)
São Carlos/SP – 13566-590, Brazil
Email: {gbpcosta, welintonandrey, tiagosn}@usp.br, jbatista@icmc.usp.br, ponti@usp.br

***Abstract*—Image classification is one of the main research problems in computer vision and machine learning. Since in most real-world image classification applications there is no control over how the images are captured, it is necessary to consider the possibility that these images might be affected by noise (e.g. sensor noise in a low-quality surveillance camera). In this paper we analyse the impact of three different types of noise on descriptors extracted by two widely used feature extraction methods (LBP and HOG) and how denoising the images can help to mitigate this problem. We carry out experiments on two different datasets and consider several types of noise, noise levels, and denoising methods. Our results show that noise can hinder classification performance considerably and make classes harder to separate. Although denoising methods were not able to reach the same performance of the noise-free scenario, they improved classification results for noisy data.**

I. Introduction

The study of noise in visual data is a matter of major interest within the image processing and computer vision communities. Due to that many different denoising algorithms were developed for both image [1] and video [2] restoration. These methods are able to improve image quality in applications raging from microscopy [3] to astronomy [4]

Over the last decades, image classification has motivated the development of many image descriptors (e.g. LBP [5], HOG [6]) and, more recently, representation learning techniques [7]. Nonetheless, the preprocessing stages of the image classification pipeline – that could incorporate and benefit from denoising techniques – have been neglected [8, 9, 10]. Moreover, little has been done to measure the impacts of different types of noise in image classification [11], which can hinder the deployment of computer vision systems in scenarios where image quality varies.

Considering the above-mentioned gaps, in this paper we experimentally measure the effects of different types of noise on image classification and investigate denoising algorithms help to mitigate this problem. By doing so, we analyse our results based on the following topics:

1) Is the performance of a classifier hampered by noise when using the LBP and HOG methods to describe the image dataset?
2) The decrease in performance is due to the fact that noise makes it harder to separate the classes or does the model learned from images without noise is not robust enough to deal with noisy images?
3) Can denoising methods help in these situations?

Our results show that classifiers suffer to generalise to different noisy data and image classification becomes harder when dealing with noisy images. Though denoising algorithms can help to mitigate the effects of noise, they may also remove important information, reducing classification performance.

II. Related work

Ponti et al. [8] divide image classification in five stages (see Figure 1) and show that the method used to convert the images from RGB to grayscale can have a substantial impact on classification performance. They also demonstrate that RGB to grayscale conversion can be used as an effective dimensionality reduction procedure. Their results show that early stages of the classification pipeline – despite being neglected in most image classification applications – can directly influence classification performance. Some other papers [10, 9] also point out the importance of these early stages. Nonetheless, as in [8], they only focus on RGB to grayscale conversion and do not consider noisy images.

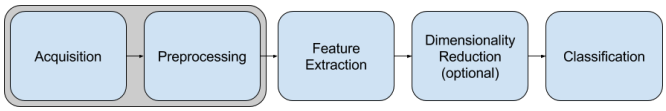

Fig. 1: Classification pipeline. Our study focuses on the first two stages (highlighted by the gray box). This image was based on Figure 1 of [8].

Dodge and Karam [11] analyse how image quality can hamper the performance of some state-of-the-art deep learning models by using networks trained on noise-free images to classify noisy, blurred and compressed images. Their results show that image classification is directly affected by image quality. Similarly, Kylberg and Sintorn [12] evaluate noise robustness of several LBP variants. Given that on both these papers the classifiers are trained in noise-free images, it is not possible to infer if the learned models are not able to deal with noisy images or if noise makes the classes harder to separate.

III. Technical background

A. Local binary patterns

Local Binary Patterns (LBP) [5] is a texture-based image descriptor that, due to its success, has several variants and

improved versions [13, 14]. In this paper, we employ the version that uses uniform patterns and it is invariant to gray scale shifts, scale changes and rotations. This variant achieves good results while generating low dimensional features.

The *LBP descriptor* is the distribution (a histogram) of texture patterns extracted for every pixel in an image. Thus, prior to computing the LBP descriptor, it is necessary to compute a texture pattern representation for each pixel. Such texture representation is called *LBP code* and it is based on the difference between a pixel and its neighbors. These neighbors can be arranged in a circle or in a square. A neighborhood is defined by the parameters R and P, where P is the number of neighbors and R is the radius of the circular neighborhood (or the side of the square neighborhood). If one of the neighbors is not at the center of a pixel, its value needs to be obtained via interpolation.

The LBP code (a binary code) for a pixel g_c and its neighbors is defined as follows:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p, \quad (1)$$

where s is the sign function and $g_0, \ldots, g_{P-1}$ are neighbors of g_c. This LBP code is invariant to grayscale shifts, because it is based on the differences of pixels and not in absolute values. Also, since only the sign of the difference result is considered, the code is invariant to scale. On the other hand, such code is not invariant to rotation.

It is possible to achieve some invariance to rotation by using the following LBP code:

$$LBP^{ri}_{P,R} = \min\{ROR(LBP_{P,R}, i) \mid i = 0, ..., P-1\}, \quad (2)$$

where $ROR(c, i)$ is the result of i circular right bit-wise shifts applied to the code c. As an example, if $c = 01110010$ and $i = 2$, then $ROR(c, i) = \mathbf{10}011100$. By always considering the minimum of all possible bit-wise shifts, a code that is more robust to rotations can be obtained.

Pietikäinen et al. [15] discovered that when LBP patterns are considered circularly, they usually contain two or less bit transitions (patterns with such characteristic were named uniform). The other patterns – that have more than two transactions – occur rarely and were called non-uniform.

Finally, to obtain the LBP descriptor – up to now we were talking about LBP codes – a histogram is computed. In this histogram, each uniform pattern has its own bin, while there is one bin for all the non-uniform patterns.

B. Histogram of oriented gradients

Based on evaluating well-normalized local histograms of image gradient orientations in a dense grid, Histogram of Oriented Gradients (HOG) [6] takes advantage of the distribution of local intensity gradients or edges directions to characterize the local object appearance and shape. This is done by diving the image window into small connected regions, called cells, in which a local histogram of gradient directions or edge directions is computed over all pixels. The final representation is obtained by combining the histograms computed in all cells of the image. HOG descriptors are particularly suited for human detection [6].

To extract HOG descriptors from an image, firstly, gradient values must be computed. This is most commonly done by filtering the color or intensity data of the image using the one-dimensional centered point discrete derivative mask in the horizontal ($[-1, 0, 1]$) and vertical directions ($[-1, 0, 1]^T$). Then, the image is divided into small cells of rectangular (R-HOG) or circular shape (C-HOG). Each pixel contained by a cell is used in a weighted manner to create a orientation-based histogram. This histogram is created for each cell and its bins are evenly spread over the orientation of the gradients. The range of the orientation can be defined over 0 to 180 degrees or over 0 to 360 degrees, depending if the gradient is "signed" or "unsigned". The contribution of a pixel to each bin of the histogram is weighted based on the magnitude of the gradient or some function of this magnitude.

To increase robustness to illumination and contrast changes, gradient strengths are locally normalized by grouping cells together into blocks. Some methods commonly used for normalization are: ℓ_2-norm (Equation 3), hysteresis-based ℓ_2 normalization [16] or ℓ_1-sqrt (Equation 4), where ν is the non-normalized vector containing all histograms of a given block, $\|\Delta\nu\|_k$ is its k norm for $k = 1, 2$ and e is a small constant.

$$f = \frac{\nu}{\sqrt{\|\Delta\nu\|_2^2 + e^2}} \quad (3)$$

$$f = \sqrt{\frac{\nu}{(\|\Delta\nu\|_1 + e)}} \quad (4)$$

Blocks typically overlap, which means that a cell can contribute to more than one block, and, therefore, to the final descriptor. The size and shape of the cells and blocks and the number of bins in each histograms are set by the user.

C. Median filter

The Median filter replaces each pixel value by the median pixel value in a $n \times n$ neighborhood centered on it. This filter can be described by the following equation:

$$\hat{z}(x, y) = median(z_k \mid k = 1, ..., n \times n), \quad (5)$$

where z_k for $k = 1, ..., n \times n$ are the pixel values within the neighborhood centered on (x, y).

D. Non-Local Means

The Non Local Means (NLM) originally presented in [17] has inspired several variations. In this paper we use the windowed version as proposed by Buades et al. [18]. Given a noisy image v, this NLM variant defines a restored version pixel i as a weighted average of all pixels inside of a window of size $s \times s$ centered on i using the following equation:

$$NL[v](i) = \sum_{j \in S_i} w(i, j)v(j), \quad (6)$$

where the weight $w(i,j)$ measures the similarity between pixels i and j and S_i is the $s \times s$ search window (s is an user-defined parameter). Each $w(i,j)$ is computed as follows:

$$w(i,j) = \frac{1}{Z(i)} e^{-\frac{||v(\mathcal{N}_i) - v(\mathcal{N}_j)||^2_{2,a}}{h^2}}, \quad (7)$$

where $\mathcal{N}_i$ and $\mathcal{N}_j$ are $p \times p$ regions centered at i and j (p is an user-defined parameter) and h is an user-defined parameter that represents filtering level. To compute the similarity between $\mathcal{N}_i$ and $\mathcal{N}_j$ an Euclidean distance weighted by a Gaussian kernel with standard deviation defined by the user-defined parameter a is used.

IV. Experiments

A. Experimental setup

To evaluate if noise hampers classification performance we generated noisy versions of two datasets (Corel and Caltech101-600) using different levels of three types of noise: Gaussian, Poisson and salt & pepper. Moreover, to understand the impacts of employing a denoising algorithm as preprocessing, we restored these noisy images using two denoising methods: Median filter and Non-Local Means. All these operations were performed on both, training and test, sets of both datasets.

We trained a different linear Support Vector Machines (SVMs) for every training set version. Given that every training set version only has one type of noise (or no noise at all), a model specialized on each level of each type of noise was created.

Then, these models were used to classify every version of the test set. As with the training sets, each test set version only contains one type of noise (or no noise at all). This allows the experiments to measure how well a model learned on a particular noisy training set performs on other types of noisy images (see Figure 2 for a diagram that summarizes this setup). In addition, by training a model using a certain type of noise (and noise level) and then evaluating its performance on a test set with the same characteristics, it is possible to make a superficial analysis on the linear separability of the problem (since linear SVMs were used).

Considering the selected datasets have more than two classes, we trained SVM models using a "one-vs-all" approach. Furthermore, to evaluate their performance, an average F1-score weighted by the number of instances in each class was used. This performance measure was chosen because it addresses the problem of evaluating the classification of unbalanced domains, that is, when classes have different number of instances.

B. Datasets

1) Corel[1]*:* a dataset containing 10800 RGB images of 80 classes, where each class has at least 100 images. Sample images from this dataset are shown in the first row of Figure 3.

[1]The Corel dataset is available at: https://sites.google.com/site/dctresearch/Home/content-based-image-retrieval

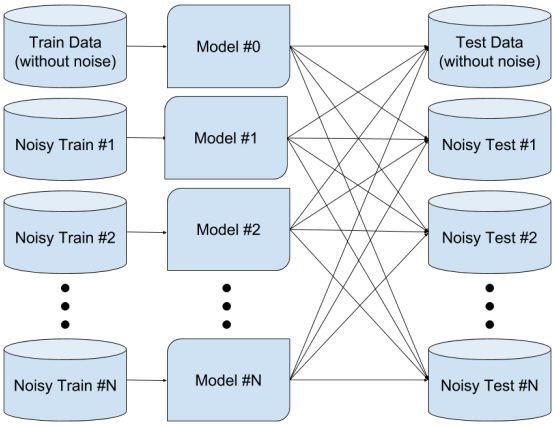

Fig. 2: Experimental setup diagram. A different model is trained for every noisy version of the training set. Then, these models are evaluated on all versions of the test set.

2) Caltech101-600[2]*:* a subset [8] of Caltech101 [19] containing 6 classes (*airplanes, bonsai, chandelier, hawksbill, motorbikes*, and *watch*), each one with 100 examples. Images from this dataset can be seen in the second row of Figure 3.

Fig. 3: Sample images from the Corel (first row) and Caltech101-600 (second row) dataset [8].

C. Reproducibility remarks and parameter values

Regarding the insertion of noise to the images, we considered three types of noise: Gaussian, Poisson and salt & pepper. First, for the Gaussian noise, we used zero mean and five different values for the standard deviation ($\sigma = \{10, 20, 30, 40, 50\}$). Figure 4 shows an example of different levels of Gaussian noise. Secondly, since the Poisson is a noise dependent signal, to adjust the intensity of the Poisson noise applied to each image, it was necessary to multiply the image by a scale factor after generating the noise, controlling its effect on the image[3]. In our tests we used five different levels for the Poisson scale factor (scale = $\{10, 10.5, 11, 11.5, 12\}$). Finally, the salt & pepper noise was applied to each pixel with five different probabilities: $p = \{0.1, 0.2, 0.3, 0.4, 0.5\}$.

For the image descriptors, the parameters were fixed for all datasets and all noise types and levels. The LBP method

[2]The Caltech101-600 dataset is available at: http://www.icmc.usp.br/pessoas/moacir/data/

[3]An in depth explanation about the scale factor for the Poisson noise can be found at: https://ruiminpan.wordpress.com/2016/03/10/the-curious-case-of-poisson-noise-and-matlab-imnoise-command/

(a) Original (b) $\sigma = 20$ (c) $\sigma = 50$

Fig. 4: Example image from the Caltech101-600 dataset with different levels of Gaussian noise.

was computed using radius $R = 1$ and circular neighborhood $P = 8$, while, for the HOG method, 8 possible gradients orientations were considered and each cell was composed by a region of 16×16 pixels, while each block contains a single cell. To obtain fixed-size feature vectors using the HOG descriptor, all images in the dataset were resized. This process was carried out in three steps. First, considering the number of rows and columns, we computed the values of the bigger and the smaller dimension of every image. Second, we obtained the smallest value for the bigger and the smaller dimensions among all images within the dataset. Thirdly, given b (the smallest value for the bigger dimension) and s (the smallest value for the smaller dimension), we resized all images in a dataset so that they end up with their bigger dimension equals to b and their smaller dimension equals to s. This procedure reduces the distortion caused by the resize.

Concerning denoising methods, all NLM restored images were generate using $p = 7$, $s = 21$ (which are recommended by the original paper [17]) and $h = 25$. With the Median filter we used a neighborhood of 11×11 pixels. Examples of the images obtained after applying a denoising method can be seen in Figure 5. During the classification stage, the parameters used to train each SVM were selected using a grid-search performed in a 5-fold cross validation on the training set.

(a) Original (b) $\sigma = 20$ (c) $\sigma = 50$

Fig. 5: Images from the Caltech101-600 dataset with different Gaussian noise levels and after applying NLM.

Due to reproducibility purposes, the code used in our experiments is available online[4].

V. Results and Discussion

Using the experimental setup presented earlier, in this section we analyse our results considering the three questions presented in Section I. As can be noticed by the heatmaps presented in Figures 7, 8, 9 and 10, the HOG descriptor obtained better results in both datasets. However, our goal is not to compare the descriptors, but rather analyse the impact of noise in image classification by shedding some light on the following questions.

[4]Repository url: https://github.com/gbpcosta/wvc_2016_noise

1) Is the performance of a classifier hampered when using the LBP and HOG methods to describe a noisy image dataset?

To answer this question we created Figures 7, 8, 9 and 10. Each one of these figures is a heatmap representing the F1-score levels obtained by a classifier in all versions of a dataset (noisy, original and restored). It is possible to observe that the best results were obtained by classifiers trained and tested using noise-free images. This means that, for the analysed scenarios, image classification using LBP and HOG descriptors classified by a linear SVM, is hampered when using noisy images as input. Additionally, the higher the noise level the lower the F1-score (see Figure 6) if we consider a model trained with the original (noise-free) train data. This effect was observed also in previous studies [12, 11], but for other descriptors, classifiers, datasets, and types of noise.

Please notice that the darkest color in the heatmaps is defined by the best result obtained in that dataset during the experiments and **not** by 1.0 (the best possible F1-score value). For that reason the scale of Figures 7 and 9 is different from the one of Figures 8 and 10.

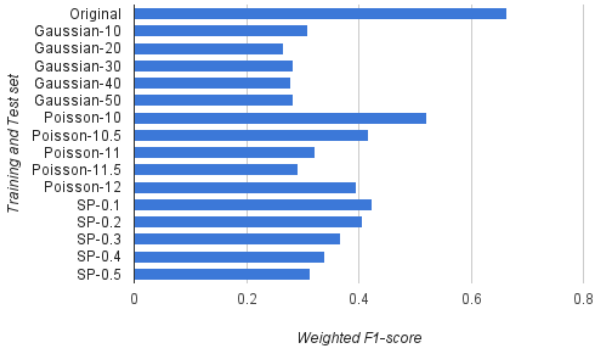

Fig. 6: LBP results for the Caltech101-600 when both training and testing was performed with the same type and level of noise.

2) The decrease in performance is due to the fact that noise makes it harder to separate the classes or does the model learned from images without noise is not robust enough to deal with noisy images?

If we look at Figure 11 it is possible to see that the models trained in a specific noise configuration have the best performance for a test set with the same noise configuration. Nevertheless, if we compare these best results for every noise level (as shown in Figure 6), the best F1 – for both descriptors in both datasets – are obtained when training and test sets are noise-free. Therefore, given that all these models where build after a grid search and that linear SVMs were used, our results indicate that the classes become less linearly separable in the presence of noise.

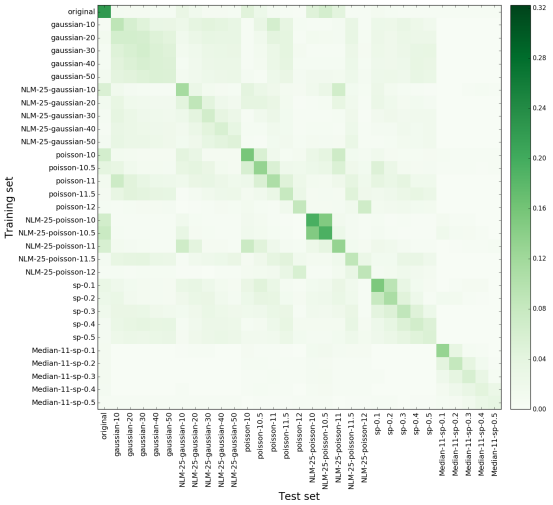

Fig. 7: LBP results for the Corel dataset.

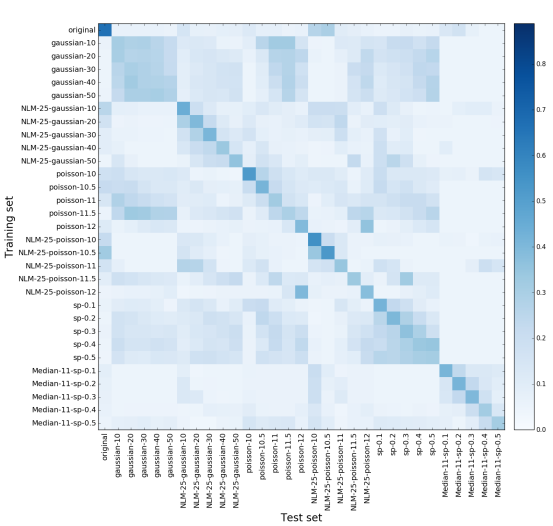

Fig. 8: LBP results for the Caltech101-600 dataset.

Those results also show that LBP and HOG are sensitive to noise, which might cause it to produce different feature spaces for the same data under different levels of noise. Thus, the SVM model might not have been able to create a classifier that could be sufficiently general for noisy future data, due to hindered class representation.

3) Can denoising methods help in these situations?

Overall, the use of denoising methods improved the classification performance when both training and test sets were affected by the same type of noise. However, the achieved result was not as good as the one obtained using the original dataset, probably due to the loss of detail and texture caused by these methods. Note, however, that models created with images after denoising did not perform well when tested with noisy images.

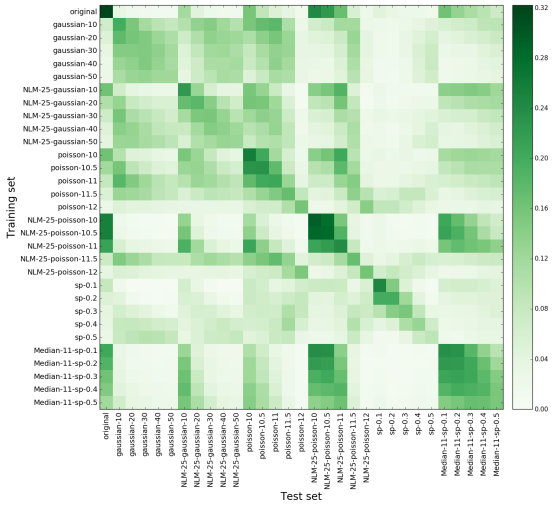

Fig. 9: HOG results for the Corel dataset.

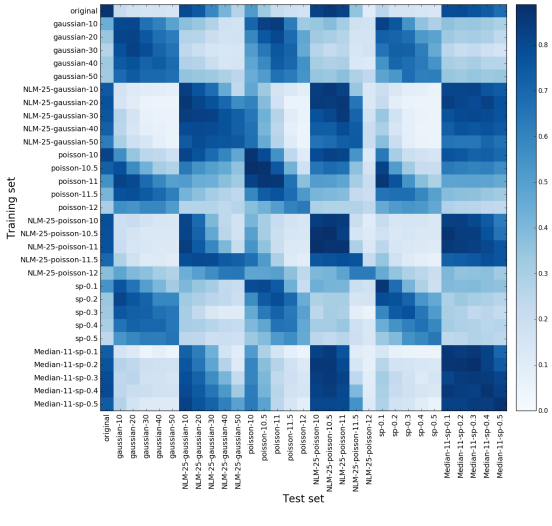

Fig. 10: HOG results for the Caltech101-600 dataset.

A. *Supplementary material*

Due to the size restrictions, not all results were presented in this paper. These results are available at: https://github.com/gbpcosta/wvc_2016_noise.

VI. Conclusion

Results presented in the previous section show that test classifiers in images with a different type of noise not only confuses the models, but also causes the problem become harder. This is noticeable on the diagonal of each heatmap, where none of the classifiers were able to overcome the performance of the classifier trained and tested with the original dataset.

When denoising is applied, the results obtained by classifying images from the same category (same type of noise or denoising method) were slightly better then the ones achieved by classifying noisy images. However, due to the smoothing

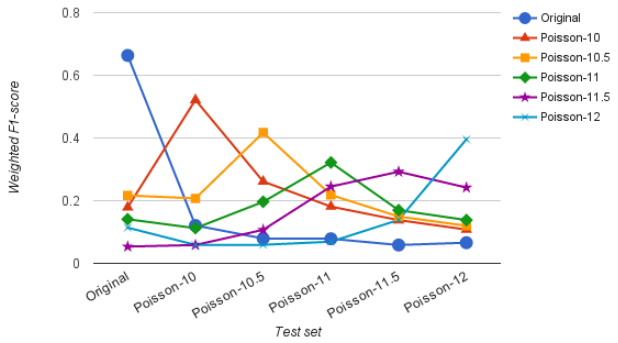

Fig. 11: Comparison of the LBP results for the Corel dataset when training and testing is performed using images affected by the Poisson noise.

caused by these methods, these results did not match the classification performance of the original dataset.

Future work include the analysis of the effect of noise in video descriptors, since temporal information might help overcome the difficulty of describing noisy data. The analysis performed in this paper should also be extended to include more datasets, descriptors and denoising methods, mainly to include deep learning methods, since these represent the state-of-the-art of image classification. Finally, the use of image quality metrics such as PSNR and SSIM can be important on comparing degraded images.

Acknowledgment

This work was supported by FAPESP (grants #2014/21888-2, #2015/04883-0 and #2015/05310-3).

References

[1] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Bm3d image denoising with shape-adaptive principal component analysis." in *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*, Saint Malo, France, 2009.

[2] W. A. Contato, T. S. Nazare, G. B. Paranhos da Costa, M. Ponti, and J. E. S. Batista Neto, "Improving non-local video denoising with local binary patterns and image quantization," in *Conference on Graphics, Patterns and Images (SIBGRAPI 2016)*, 2016.

[3] M. Ponti, E. S. Helou, P. J. S. G. Ferreira, and N. D. A. Mascarenhas, "Image restoration using gradient iteration and constraints for band extrapolation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 1, pp. 71–80, Feb 2016.

[4] S. Beckouche, J.-L. Starck, and J. Fadili, "Astronomical image denoising using dictionary learning," *Astronomy & Astrophysics*, vol. 556, p. A132, 2013.

[5] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in *ICPR94*, 1994, pp. A:582–585.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[7] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1798–1828, 2013.

[8] M. Ponti, T. S. Nazaré, and G. S. Thumé, "Image quantization as a dimensionality reduction procedure in color and texture feature extraction." *Neurocomputing*, vol. 173, pp. 385–396, 2016.

[9] M. Ponti and L. C. Escobar, "Compact color features with bitwise quantization and reduced resolution for mobile processing," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 751–754.

[10] C. Kanan and G. W. Cottrell, "Color-to-grayscale: does the method matter in image recognition?" *PloS one*, vol. 7, no. 1, p. e29740, 2012.

[11] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, Jun. 2016, pp. 1–6.

[12] G. Kylberg and I.-M. Sintorn, "Evaluation of noise robustness for local binary pattern descriptors in texture classification." *EURASIP J. Image and Video Processing*, vol. 2013, p. 17, 2013.

[13] L. Nanni, A. Lumini, and S. Brahnam, "Survey on LBP based texture descriptors for image classification," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3634–3641, Feb. 2012.

[14] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[15] M. Pietikäinen, T. Ojala, and Z. Xu, "Rotation-invariant texture classification using feature distributions," *Pattern Recognition*, vol. 33, pp. 43–52, 2000.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[17] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 60–65.

[18] A. Buades, B. Coll, and J. Morel, "Non-Local Means Denoising," *Image Processing On Line*, vol. 1, 2011.

[19] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.