On Automating Data Augmentation for Deep Image Denoising

Eduardo Fernandes Montesuma[1]*, Florian Lemarchand[1] and Maxime Pelcat[1]

Abstract
Despite their successful applications, deep learning models need a considerable amount of data to be trained. When enough data is not available, data augmentation is an outstanding technique that populates datasets with new artificial samples. However, to obtain a good augmentation policy it is often needed to enlarge the number of hyperparameters to tune. Hence, a technique to automatically choose the best policies to augment data can both save time and increase performances of models. This paper explores how policies predicted by the existing AutoAugment technique can be extended to enhance deep image denoising. By using a Denoising Autoencoder to restore artificially corrupted data on MNIST, EMNIST and Challenge datasets, the experiments shown that augmentation policies based on geometrical transformations were able to increase the PSNR of restored images by 9.1%, 8.9% and 1.2% respectively, on the best case. Moreover, the performance of denoising models is analyzed under a variable amount of available training data, showing that the usage of data augmentation is more critical in small datasets.

Keywords
Image Denoising — Deep learning — Image processing – Automatic Data Augmentation

[1] *Department of Electronics and Industrial Informatics, INSA Rennes, Rennes, France*
***Corresponding author**: edumontesuma@gmail.com

Contents

Introduction

The training of machine learning models has evolved as both processing and storing capabilities of computers have increased. With increasing complexity, deep learning models became the state-of-the-art in many tasks of classical signal processing, such as image processing.

Among the branches of image processing, image restoration deals with the digital restoration of corrupted images. The sources of these corruptions are multiple, such as noise, blur or even occlusions. Moreover, this field has great importance in areas such as medicine [1] or astronomy [2].

Despite Deep Neural Networks (DNNs) represent the current state of the art in image processing, these networks have a major drawback: most deep learning algorithms require a huge amount of data, which is a serious constraint for some applications, such as medicine [3]. The main concern involving the training of DNNs with small datasets is lack of generalization. In more precise terms, once trained, the model fails to correctly classify new samples. In that case, data augmentation is proposed as an avenue to enhance learning models.

Data augmentation consists in artificially creating new data samples from existing ones. Indeed, as remarked in [4], this method is regarded as an *implicit regularization method*, meaning that it reduces the generalization error without explicitly constraining the model's capacity.

Nevertheless, a significant effort is required when de-

signing augmentation policies for a particular dataset. This involves domain knowledge, and possibly adds new hyper-parameters to tune. Hence, a method to automatically select the best augmentations according to an evaluation metric can save time in machine learning projects. This was the approach taken by [5], which designs the AutoAugment algorithm to search optimal augmentation policies in the context of image classification.

Motivated by the state-of-the-art results yielded by such algorithm, the present paper shows an adaptation of AutoAugment to an image denoising context. Moreover, we seek to investigate how augmentation policies act on diverse datasets, how effective they can be on diverse datasets, and how their performance is affected under a variable amount of data.

This paper is divided as follows, in Section 1, state-of-the-art in image denoising and data augmentation are discussed. In Section 2, the adaptations done to the AutoAugment algorithm to predict optimal augmentation policies for image denoising are described. In Section 3, image denoising through multiple datasets using the policies predicted by the AutoAugment algorithm are explored. Finally, Section 4 presents the paper conclusions and perspectives.

1. Related Work

This section discusses the state-of-the-art for both denoising and data augmentation methods. Acknowledging that nowadays the best methods make use of DNNs, the discussion is based on learning-based models.

1.1 Image denoising

Image denoising is a particular case of a bigger problematic, called image restoration. In image restoration, one has a corrupting function $\mathbf{y} = \eta(\mathbf{x})$, where $\mathbf{x}$ is the original image, $\mathbf{y}$ is the degraded image and η is a stochastic degradation process. Hence, a mathematical model for a type of noise describes how η corrupts the data $\mathbf{x}$.

Using these definitions, there are plenty of noise models. Among them, this paper targets two commonly used ones: the Additive White Gaussian Noise (AWGN) model, and the Salt and Pepper (s&p) model. It is worth mentioning that the presented methods would extend to other noise models.

For the first type, the mathematical assumption used is, $\eta(\mathbf{x}) = \mathbf{x} + \varepsilon$, where ε is the noise component, distributed according a normal distribution $\mathcal{N}(\mathbf{0}, \sigma^2)$. Commonly, the degree of degradation in a AWGN noised image is measured as a percentage of the image maximum pixel value, that is, $\sigma \in [0, 100]\%$, where 100% represents the image maximum pixel value.

For the s&p noise, $0 \leq p \leq 1$ is the probability of a pixel to be disturbed. When disturbed, a pixel has equal probability to become white (salt) or black (pepper). The intensity parameter of this noise is, hence, the probability of disturbance. Furthermore, the degree of degradation of a s&p noise is expressed in terms of percentage. For instance, for $p = 0.2$, approximately 20% of image pixels are affected by such noise.

To measure the level of corruption in a noised image, one needs to have the noisy image $\mathbf{y}$ and the ground-truth $\mathbf{x}$. The Peak Signal to Noise Ratio (PSNR) is adopted as main metric of noise level, which is measured in decibels as,

$$PSNR(\mathbf{y}, \mathbf{x}) = 10 log_{10}\left(\frac{(max_{i,j}\mathbf{x}_{i,j})^2}{MSE(\mathbf{y}, \mathbf{x})}\right), \tag{1}$$

$$MSE(\mathbf{y}, \mathbf{x}) = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W}(\mathbf{y}_{i,j} - \mathbf{x}_{i,j})^2. \tag{2}$$

To perform restoration on images contaminated with noise, there are two approaches: classical signal processing techniques, and learned models. While learning methods rely on massive amounts of data to predict input reconstructions, the classical approach rely on algorithmic strategies such as filtering.

Concerning classical approaches for image denoising, most of them rely on filtering as a way to separate the original signal from the noise component. For instance, this approach was taken in [6] by employing bilateral filters and [7], by using collaborative filtering.

Nevertheless, a part of classical approaches represent a half-way between common signal processing techniques and deep learning algorithms. For instance, statistical approaches, such as K-SVD [8] and Principal Component Analysis (PCA), are data-driven algorithms that learn data representations that exclude the noise components present in corrupted data.

Finally, deep learning based image denoising rely in the feature extraction done by network layers. This is the case for the Stacked Denoising Autoencoder technique proposed in [9], where the neural network is trained to learn features robust to noise. Such approach was further explored, by forcing the autoencoders to learn sparse representations [10].

1.2 Data augmentation

Data augmentation policies have been widely employed to enhance classification performance on pattern recognition tasks, such as Digit Recognition [11], or even other methods, such as Super-Resolution Imaging [12] and Image Denoising [13].

To define an augmentation policy, one has to first define the set of transformations that compose them. By following the definitions of [5], a data transformation is a triple $T = (t, p, m)$, where $t \in \mathcal{T}$ is the transformation type, $p \in \mathcal{P}$, its probability, and $m \in \mathcal{M}$ its magnitude. It follow, then, that a policy is a set of transformations $\pi = \{T_1, \cdots, T_n\}$.

It is noteworthy that in such setting, transformations are stochastic. For a triple $T = (t, p, m)$, and a batch of data, transformation t is applied to each image in the batch with probability p and magnitude m. That way data variability is improved on each batch. To illustrate how policies act on batches, Figure 1 is presented.

To define how well augmentation policies have performed, it is necessary to define an evaluation metric, $\mathcal{L}$. A policy π_1 is said to be better than another π_2, if it yields a higher score in terms of the defined metric.

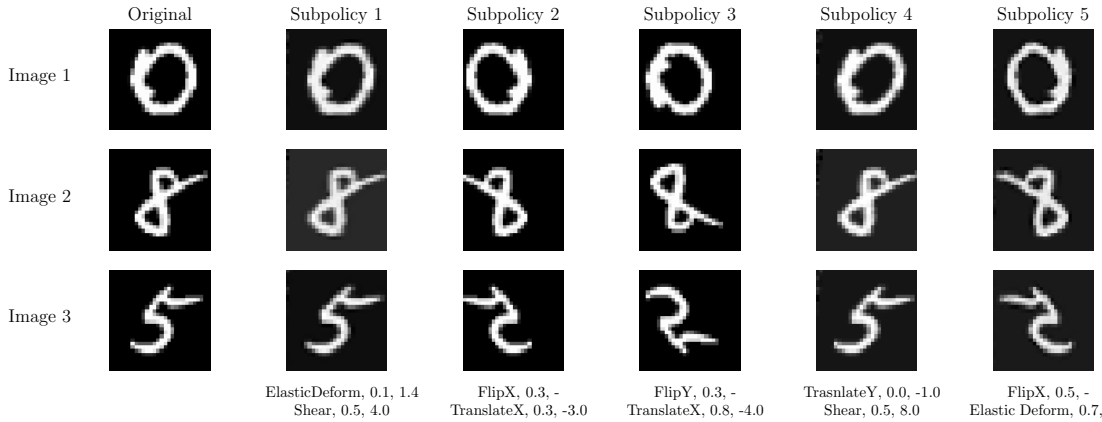

Figure 1. Figure adapted from [5]. At the bottom, for each sub-policy the name of its composing transformations, followed by its respective probability and magnitude are shown. Notice that a same batch of images is transformed into different samples on each sub-policy.

Given that the intent of applying data augmentation is to improve the generalization of learning models, $\mathscr{L}$ is often defined as the validation classification accuracy, for classification problems. In order to evaluate the quality of denoising models, however, the PSNR metric is used.

The problem of choosing transformations can be posed as a discrete optimization problem, that is, choosing a particular policy consisting of n transformations is a search among $(|\mathscr{T}| \times |\mathscr{P}| \times |\mathscr{M}|)^n$ choices. The optimal policy is then chosen based on $\mathscr{L}$, $\hat{\pi} = argmax\ \mathscr{L}(\pi)$.

Therefore, the search space scales exponentially with the number of number of transformations in a policy. Hence, brute-force algorithms can be rather inefficient and as time-consuming as picking transformation by hand.

With this motivation, authors in [5] have defined an algorithm based on reinforcement learning to find $\hat{\pi}$, the AutoAugment technique, which rely on Reinforcement Learning to predict optimal policies.

Under the reinforcement learning framework, one has an agent, and an environment which it interacts with. The agent can take actions A_t onto the environment, which yield a correspondent reward r_t. By doing so, the agent manages to change the systems state from S_t, to S_{t+1}.

By using these definitions, The problem environment is the neural network training, which henceforth shall be called *child network*. According to the algorithm developed in [5], there is a second neural network representing the agent, which the authors have defined as the *controller network*.

In such setup, the actions the agent can take are training the *child network* following a given augmentation policy π. By doing so, it yields an reward, which is the evaluation metric $\mathscr{L}$. The objective of the *controller network* is to predict optimal policies, in order to maximize its rewards. This algorithm is summarized on Figure 2.

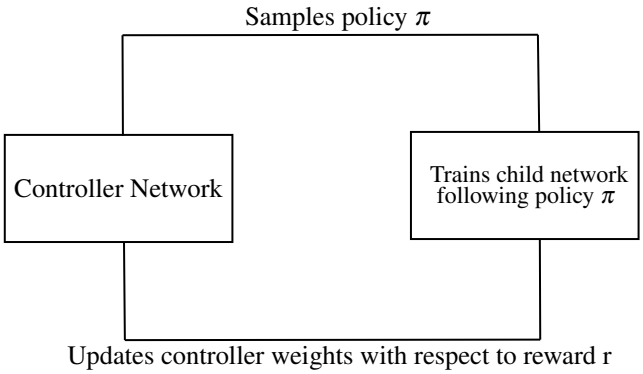

Figure 2. Scheme of the strategy adopted in [5] to find optimal augmentation policies. The controller is a neural network trained to optimize the child network evaluation metric.

2. Methodology

In this section, the algorithm proposed by [5] is used to predict optimal augmentation policies for the denoising problem. In Section 2.1 the transformations used to form sub-policies are discussed. In Section 2.2, the architecture used for image denoising is presented. Finally, in Section 2.3, the *controller network* architecture and its training algorithm are detailed.

2.1 Image Transformations

To artificially augment the available data samples, the approach presented in [5] was taken. Being so, pairs of transformations are used to define a sub-policy. A group of five sub-policies is called a policy.

Each transformation can have one among a predefined list of types. The set of available transformations is shown on Table 1. Alongside its type, each transformation in a sub-policy has another two attributes: a probability of application, and a magnitude.

The hole of probability is to increase data diversity in

each batch, as transformations are not always applied. Moreover, the magnitude parameter regulates the transformation intensity. Figure 1 shows an example of policy application.

Transformation Type	Min	Max
Shearing[1]	-20	20
Horizontal Translation[2]	-0.2	0.2
Vertical Translation[3]	-0.2	0.2
Rotation[4]	-15	15
Horizontal Flip[5]	N.A.	N.A.
Vertical Flip	N.A.	N.A.
Elastic Deformation[6]	0	7

Table 1. Available transformations for AutoAugment algorithm.

Being so, each transformation is represented by a type, a discretized probability value (between 0, and 1), and a discretized magnitude value (between a minimum and a maximum, depending on each transformation). The discretization is took uniformly on each interval, having a length of 11. As consequence, the search space consists of $(7 \times 11 \times 11)^{10} \approx 1.9 \times 10^{29}$ possible policies.

Concerning the transformations, shearing, translations and rotation are affine geometric transformations, meaning that for each of these transformations and image $\mathbf{x}$, the resulting transformed image $\mathbf{y}$ may be expressed as,

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}, \tag{3}$$

where $\mathbf{A}$ is a matrix, and $\mathbf{b}$, a vector.

Elastic Deformation was proposed by [11] in the context of MNIST dataset classification. For each pixel (i, j) in $\mathbf{x}$, a random displacement field is defined by $\mathbf{u} \sim \mathcal{N}(0, \sigma^2)$. The transformation is then defined as,

$$\mathbf{y} = \mathbf{x} + \alpha\mathbf{u}. \tag{4}$$

Note that both the random field intensity α, as its variance σ^2 are hyperparameters.

2.2 Child Network

The purpose of the *child network* is to perform denoising on input images. As the controller training process is independent of *child network* architecture, its architecture is chosen to be kept simple in order to optimize performance. For that reason, a simpler approach was taken, than those mentioned in Section 1. Therefore, an autoencoder is used to denoise images, whose architecture is shown on Table 2.

[1] Computed in pixels.
[2] Computed as a percentage of image width.
[3] Computed as a percentage of image height.
[4] Computed in degrees.
[5] Flip transformations do not have an intensity value.
[6] Corresponds to the random field intensity α, for a fine-tuned variance value of $\sigma = 2$.

Encoder network		
Layer	Kernel Size	# of filters
Convolutional Layer 1	3	16
Max Pooling Layer 1	2	-
Convolutional Layer 2	3	8
Max Pooling Layer 2	2	-
Convolutional Layer 3	3	8
Max Pooling Layer 2	2	-
Decoder network		
Layer	Kernel Size	# of filters
Convolutional Layer 1	3	8
Up-sampling Layer 1	2	-
Convolutional Layer 2	3	8
Up-sampling Layer 2	2	-
Convolutional Layer 3	3	16
Up-sampling Layer 2	2	-

Table 2. *Child network* architecture summary

As discussed in [9], the intent of autoencoders is to learn an useful input representations through a pair of parametric mappings, $\mathbf{f}_{\theta_1} : \mathscr{X} \to \mathscr{H}$ the encoder part, and $\mathbf{g}_{\theta_2} : \mathscr{H} \to \mathscr{Y}$ the decoder part, where $\mathscr{X}$ is the input space, $\mathscr{Y}$, the output space and $\mathscr{H}$ is the hidden or latent space. The principal attention to those kinds of networks is upon the latent space $\mathscr{H}$, where the input representations lies.

Although the original intention of the authors was not to provide an algorithm for denoising, such architecture can still be used for that end, since by training the autoencoder using the proposed settings, it learns to reconstruct a clean approximation to the corrupted input.

2.3 Controller Network

The design of the *controller network* follows the description of an actor method [14], which work with parametric families of policies, that is, $\pi = \pi_\theta$, where θ is the family parameter. By using this definition, a neural network can be used to predict policies, by letting letting θ be the its weights.

The particular choice of neural network architecture is not important for the algorithm, however the same approach as reported in [5, 15] was taken. In such setting, a Long Short-Term Memory (LSTM) is used to predict each sub-policy, as shows Figure 3.

In order to train the *controller network*, the same method as [5] and [15] has been implemented, using the Proximal Policy Optimization (PPO) algorithm [16].

Since policies are predicted using a neural network, they can be parametrized through the network's weights, here expressed as θ. The PPO algorithm, then, manages to update the weights θ through the minimization of a surrogate loss given by Equation 5.

Being $r_t = \frac{\pi_\theta}{\pi_{\theta_{old}}}$, the probability ratio between the policies predicted by θ, and those predicted by the old parameter vector θ_{old}, the surrogate loss is expressed as,

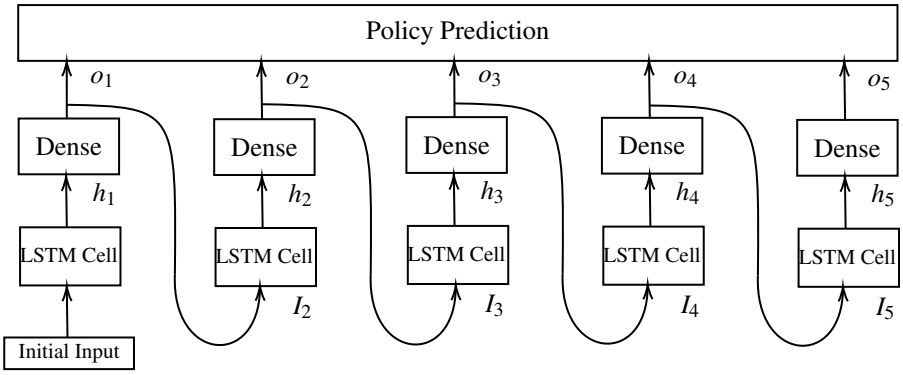

Figure 3. Schematic of controller architecture. An initial input is given to the first LSTM Cell, for which its output is fed into a dense, fully connected layer, whose output is a probability distribution over transformations within a sub-policy. This distribution is then fed into the next step, in order to acquire the probability distribution for the next sub-policy. As mentioned early, each policy is composed by 5 sub-policies, hence the policy prediction step is based on five sub-policy distributions.

$$L^{PPO} = \hat{\mathbb{E}}_t \left[min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t \right) \right]. \quad (5)$$

In Equation 5, ε is a hyperparameter. π_θ is the current estimate of policy probability distribution, while $\pi_{\theta_{old}}$, is the previous prediction. $\hat{\mathbb{E}}$ corresponds to the empirical mean operator, and $\hat{A}_t$, to the estimation of "advantage", a scalar defined as,

$$\hat{A}_t = r_t - \hat{b}_t, \quad (6)$$

where $\hat{b}_t$ is called baseline. The baseline corresponds to an estimate of the value of the current prediction. The baseline may be estimated through various methods. For instance, in [5] and [15], the authors have used an exponential moving average over past reward values, while in [16], the authors propose an additional neural network that shares parameters with the controller.

Moreover, as [16] suggests, the optimization of Equation 5 can be done through automatic differentiation, such as those implemented in deep learning frameworks.

3. Experiments and Results

In this section, the enhancement done through policies predicted by the *controller network* is explored. For that end, the neural network architecture used for denoising is the one described in Table 2.

The code for the experiments was written in Python[7], and is available on a public Github repository[8]. To run the code, a Dell Precision 5820 Desktop with RAM memory of 32GB, Hard Disk memory of 500GB, processor Intel Xeon W-2125 4.00GHz, GPU GeForce 1080Ti with RAM memory of 12GB, running Ubuntu 18.04 64 bits was used.

The section is divided as follows: in Section 3.1, the training of the *controller network* is described, and the sampled optimal policies are shown. In Section 3.2, denoising performance with augmentation policies is compared through different datasets.

[7] https://www.python.org/

[8] https://github.com/eddardd/Automatic-Data-Augmentation

3.1 Controller network training

In order to train the controller, the MNIST dataset was used to train the *child network* several times. The train, validation and test data was artificially corrupted with 25% AWGN.

To make the use of augmentations more critical, the amount of data available was severely reduced, limiting the training data to only 100 data samples taken randomly, before training time, out of a total of 50000.

In the experiments, each *child network* is trained through 500 iterations, and then, evaluated on a independent validation set. The validation PSNR is used as the controller's reward.

The controller is then trained following the PPO algorithm. It is trained using Stochastic Gradient Ascent, with learning rate parameter of 1×10^{-3}. The learning process was ran through a total of 500 epochs. Moreover, the baseline is estimated through a moving average of windows size 3, and the ε hyperparameter set to 0.2, as [16] suggests.

When comparing the training process with those found in the literature, specially with [5], it is noteworthy that such training length is rather insufficient to converge at an optimal policy. Therefore, the *controller network* behaves as a random search method.

Despite its inefficiency, the controller still predicts relevant policies that improve denoising models. Therefore, during the controller training, the five high scored policies are saved, so that their performance can be further tested. A summary of those policies can be found on Table 4, in the Appendix section.

3.2 Denoising performance

To effectively measure the policy improvement in a given problem, models are trained with, and without augmentation policies on artificially noised data. These experiments are done with two types of artificially noised data: one corrupted with AWGN, and other corrupted with Salt and Pepper noise.

To further study how data transformations influence on different datasets, these experiments are compared through three different dataset: the MNIST, EMNIST [17] and Challenge[9] datasets. All these databases were artificially noised.

The augmentation policy used to train these datasets is Policy π_5, from Table 4. Each dataset is restricted to 500 samples, where the

[9] https://www.figure-eight.com/dataset/open-images-annotated-with-bounding-boxes/

child network described in Section 2.2 is trained to perform denoising. To that end, a first model is trained with augmented train data. It is then evaluated on raw validation data. The result is compared to a baseline, which corresponds to the validation performance of the same model trained with non-augmented train data. In each run, the *child network* is trained through 500 epochs, except for the Challenge dataset, where the network was trained through 250 epochs. In all runs, a learning rate of 0.02 and a exponential decay of 0.1 were used.

In Table 3, a summary of denoising performance is shown. In Figure 4 the denoising results are shown for an image sample of each dataset, for each noise type.

MNIST Denoising				
Noise Type	**Noise Intensity**	**Performance**	**Baseline**	**Increase**
AWGN	25%	17.82dB	16.43dB	8.4%
S&P	20%	17.51dB	16.05dB	9.1%
EMNIST Denoising				
Noise Type	**Noise Intensity**	**Performance**	**Baseline**	**Increase**
AWGN	25%	18.32dB	17.47dB	4.9%
S&P	20%	18.39dB	16.88dB	8.9%
Challenge Denoising				
Noise Type	**Noise Intensity**	**Performance**	**Baseline**	**Increase**
AWGN	25%	23.51dB	23.22dB	1.2%

Table 3. Denoising summary for three datasets.

Notably, the augmentation policies have yielded better results in both MNIST and EMNIST datasets, having approximately 9% of improvement over the baselines. Such behavior was already expected, since the transformations in Table 1 were already know to improve MNIST dataset performance [11]. Moreover, since both datasets share geometrical structures, to transfer policies between these two datasets is an easier task.

Nevertheless, the application of policy π_5 on Challenge dataset was less effective than its application on MNIST and EMNIST. Such results are better understood by looking at the structure of these datasets: since the Challenge dataset has a more diverse input space than MNIST and EMNIST, the application of simple geometrical transformations is less likely to enrich the input space data distribution. A better approach would be to consider a larger set of available transformations, comprehending pixel and histogram transformations.

To further study how data availability impact on denoising performance, another experiment is done. By training models with a variable amount of available data samples, the performance with augmentations is compared to the one without augmentations. The result is displayed in Figure 5. By analysing those figures, the following conclusions are drawn:

- As these figures show, the models are more sensitive to augmentation policies in low data regime. This is due the fact that, when enough data is not available, deep neural networks tend to overfit the data.
- In the general case, using reasonable policies tends to enhance the generalization performance of denoising models. The reasons for this phenomena are that, first, augmentation policies populate the input space with new artificial samples, enriching the data distribution. Second, the transformed samples can induce data invariances in the model [5], that is, it becomes robust to rotation or translation, for instance.

4. Conclusion

The present paper was motivated by the reported success of data augmentation policies in image processing applications [11, 12, 13]. To explore how data augmentation can be used to enhance image denoising, the approach suggested by [5] was used under some adaptations. Effectively, instead of using validation accuracy as the reward of the *controller*, the validation PSNR was used.

To investigate how augmentation policies can enhance learned models, a denoising autoencoder was trained twice, one time with augmentation policies, and another without any transformations. By limiting the size of each dataset, the results have shown a best-case increase of 9.1% and 8.9% in the validation PSNR for MNIST and EMNIST datasets, respectively.

To further analyse how the predicted policy may be extended to other datasets, it was applied to the Challenge dataset, which is a dataset with more diverse image samples. The results have shown a performance increase of 1.2%, which was less significant than those for the latter mentioned datasets.

Finally, the importance of augmentations under a variable amount of available data was explored by comparing the performance of models trained with and without augmentation policies. As expected, when data is insufficient for the model to have a good generalization performance, augmentations play a key-role in preventing overfitting. As the number of available samples increase, the performance increase becomes more subtle.

Acknowledgments

This work was conducted during a scholarship supported by the International Cooperation Program CAPES/BRAFITEC at the Institut National de Sciences Appliquées de Rennes, Financed by CAPES – Brazilian Federal Agency for Support and Evaluation of Graduate Education within the Ministry of Education of Brazil.

References

[1] Larissa Cristina dos Santos Romualdo, Marcelo Andrade da Costa Vieira, and Homero Schiabel. Mammography images restoration by quantum noise reduction and inverse mtf filtering. In *2009 XXII Brazilian Symposium on Computer Graphics and Image Processing*, pages 180–185. IEEE, 2009.

[2] Elena Anisimova, Jan Bednar, and Petr Pata. Astronomical image denoising using curvelet and starlet transform. In *2013 23rd International Conference Radioelektronika (RADIOELEKTRONIKA)*, pages 255–260. IEEE, 2013.

[3] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[4] Alex Hernández-García and Peter König. Data augmentation instead of explicit regularization. *CoRR*, abs/1806.03852, 2018.

[5] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[6] Ming Zhang and Bahadir K Gunturk. Multiresolution bilateral filtering for image denoising. *IEEE Transactions on image processing*, 17(12):2324–2333, 2008.

[7] K Dabov, A Foi, V Katkovnik, and K Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. image processing, ieee transactions on 16 (8), pp. 2080-2095. 2007.

[8] Michal Aharon, Michael Elad, Alfred Bruckstein, et al. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311, 2006.

[9] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[10] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012.

[11] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3, 2003.

[12] Jin Yamanaka, Shigesumi Kuwashima, and Takio Kurita. Fast and accurate image super resolution by deep cnn with skip connection and network in network. In *International Conference on Neural Information Processing*, pages 217–225. Springer, 2017.

[13] Peng Liu and Ruogu Fang. Wide inference network for image denoising via learning pixel-distribution prior. *arXiv preprint arXiv:1707.05414*, 2017.

[14] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[15] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[17] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

5. Appendix

5.1 Predicted sub-policies

	Operation 1			Operation 2		
	Type	**Probability**	**Magnitude**	**Type**	**Probability**	**Magnitude**
π_1, **Validation PSNR**: 16.78						
Sub-policy 0	TranslateX	0.2	-5.0	TranslateY	0.0	-1.0
Sub-policy 1	FlipX	0.3	N.A.	TranslateX	0.0	-2.0
Sub-policy 2	TranslateY	0.1	0.0	TranslateX	0.4	2.0
Sub-policy 3	TranslateX	0.2	-2.0	Elastic Deform	0.2	-4.9
Sub-policy 4	FlipY	0.5	N.A.	TranslateY	0.0	-2.0
π_2, **Validation PSNR**: 16.84						
Sub-policy 0	Elastic Deformation	0.1	1.4	Shear	0.5	4.0
Sub-policy 1	FlipX	0.3	N.A.	TranslateX	0.3	-3.0
Sub-policy 2	FlipY	0.3	N.A.	TranslateX	0.8	-4.0
Sub-policy 3	TranslateY	0.0	-1.0	Shear	0.5	8.0
Sub-policy 4	FlipX	0.5	N.A.	Elastic Deform	0.7	2.8
π_3, **Validation PSNR**: 17.04						
Sub-policy 0	Shear	0.0	-20.0	TranslateX	0.0	-4.0
Sub-policy 1	FlipY	0.7	N.A.	TranslateY	0.6	0.0
Sub-policy 2	Elastic Deform	0.9	2.1	TranslateX	1.0	-3.0
Sub-policy 3	TranslateY	0.0	-1.0	FlipY	0.4	N.A.
Sub-policy 4	FlipY	0.2	N.A.	Elastic Deform	0.7	-2.8
π_4, **Validation PSNR**: 17.11						
Sub-policy 0	Elastic Deformation	0.1	6.3	TranslateX	0.2	-5.0
Sub-policy 1	TranslateX	1.0	-2.0	Shear	0.0	12.0
Sub-policy 2	FlipY	0.8	N.A.	TranslateY	0.1	1.0
Sub-policy 3	Elastic Deformation	0.0	1.4	Shear	0.0	16.0
Sub-policy 4	Shear	0.7	-8.0	Elastic Deform	0.0	2.8
π_5, **Validation PSNR**: 17.34						
Sub-policy 0	Elastic Deformation	0.4	3.5	TranslateX	0.0	-4.0
Sub-policy 1	FlipY	0.6	N.A.	TranslateY	0.0	-5.0
Sub-policy 2	Shear	0.9	-16.0	TranslateX	0.0	-2.0
Sub-policy 3	TranslateY	0.9	-3.0	Shear	0.1	-8.0
Sub-policy 4	FlipY	0.7	N.A.	TranslateY	0.1	-4.0

Table 4. Summary of best sampled policies.

5.2 Denoising summary

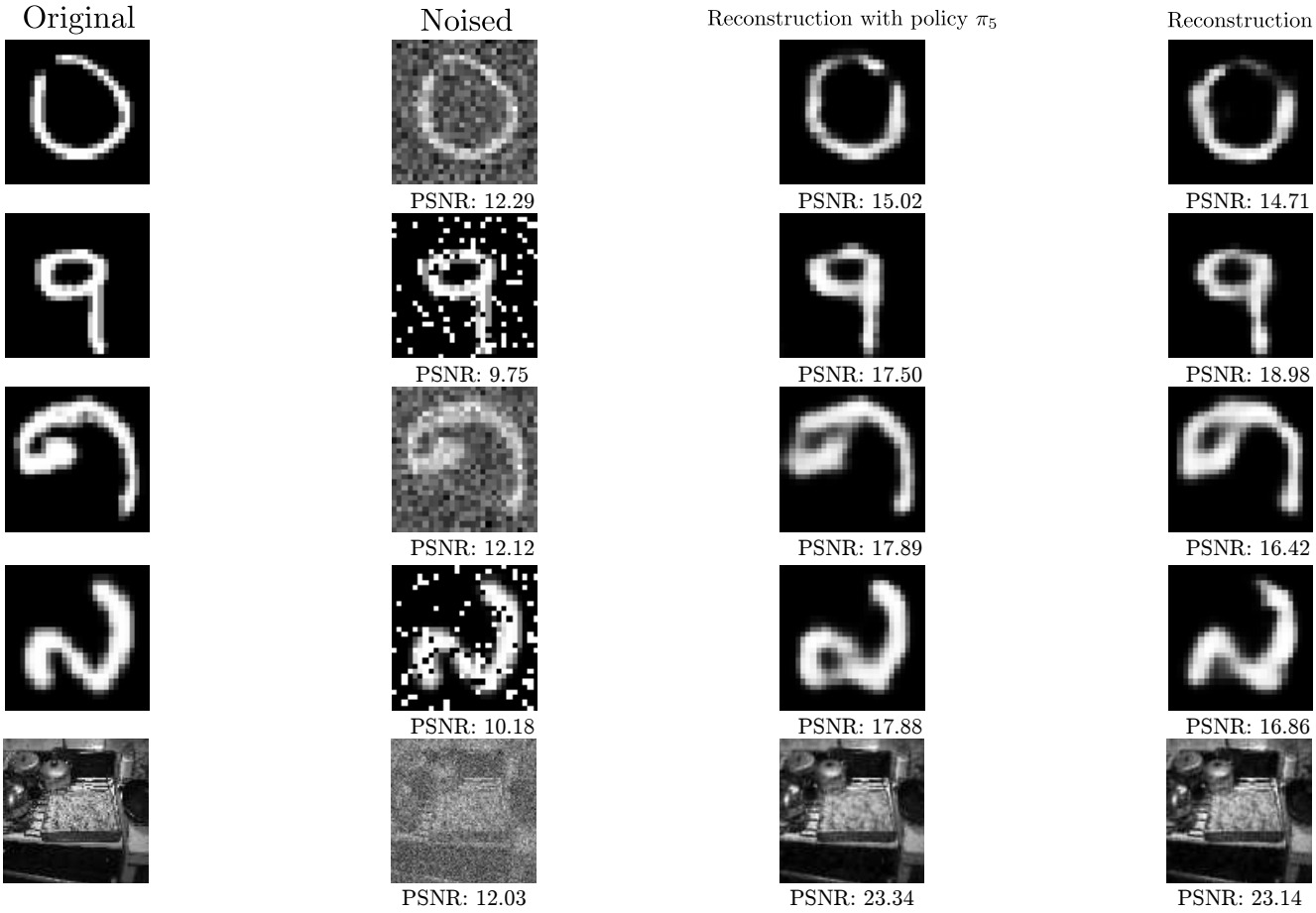

Figure 4. From top to bottom, a summary of denoising with and without the application of augmentation policy π_5 for the MNIST dataset (25% AWGN and 20% s&p), EMNIST dataset (25% AWGN and 20% s&p) and Challenge dataset (25% AWGN). Notice that in each case, the predictions made by models trained with augmentation policies were better in terms of PSNR.

5.3 Policy Performance with variable amounts data

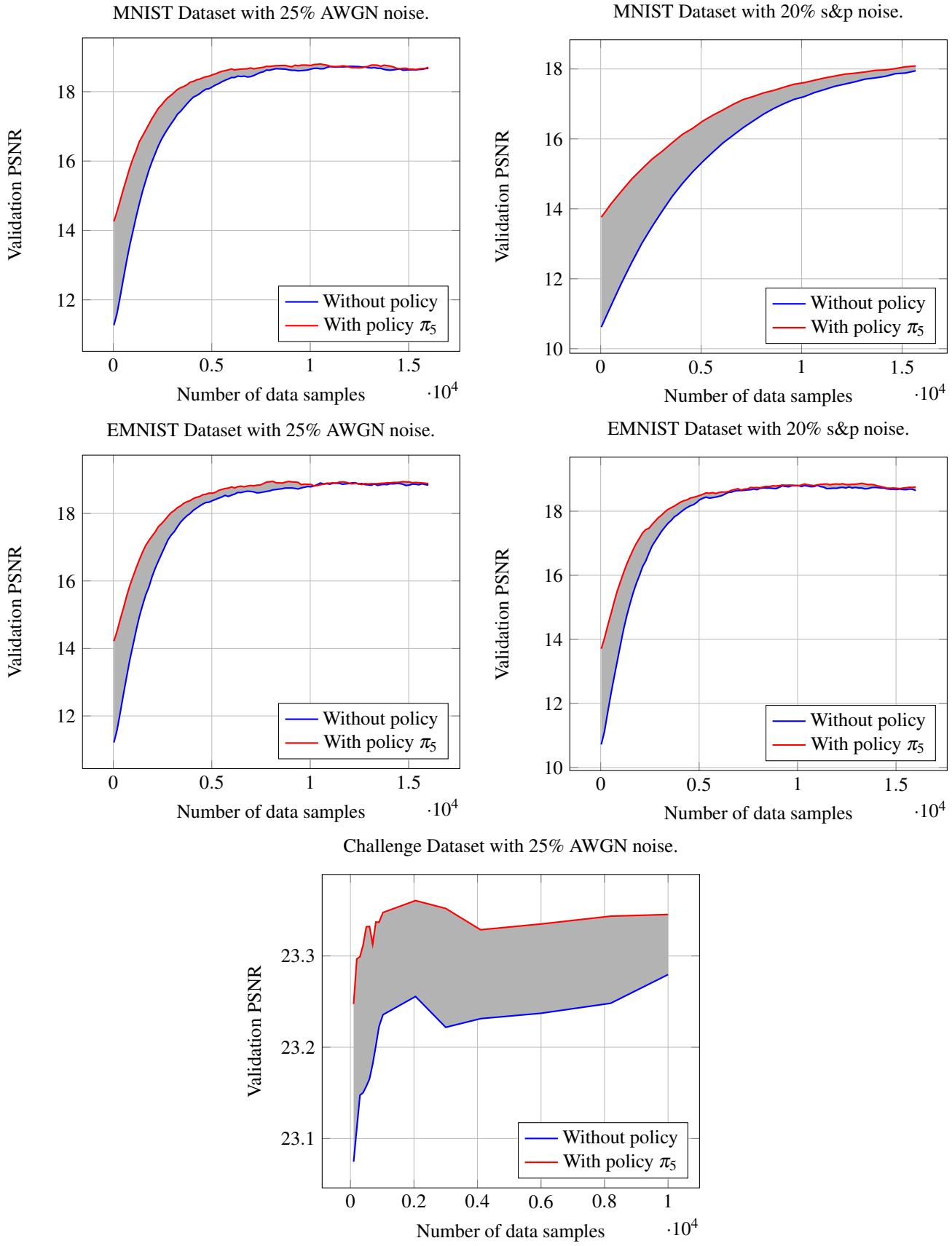

Figure 5. Performance comparison between models trained on augmentation data and raw data, under a variable amount of available samples.