MAP-GAN: UNSUPERVISED LEARNING OF INVERSE PROBLEMS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Brandon Campanella

December 2021

COMMITTEE MEMBERSHIP

TITLE:	MAP-GAN: Unsupervised Learning of Inverse Problems
AUTHOR:	Brandon Campanella
DATE SUBMITTED:	December 2021
COMMITTEE CHAIR:	Jonathan Ventura, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Foaad Khosmood, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Dongfeng Fang, Ph.D. Assistant Professor of Computer Science

ABSTRACT

MAP-GAN: Unsupervised Learning of Inverse Problems

Brandon Campanella

In this paper we outline a novel method for training a generative adversarial network based denoising model from an exclusively corrupted and unpaired dataset of images. Our model can learn without clean data or corrupted image pairs, and instead only requires that the noise distribution is able to be expressed analytically and that the noise at each pixel is independent. We utilize maximum a posteriori estimation as the underlying solution framework, optimizing over the analytically expressed noise generating distribution as the likelihood and employ the GAN as the prior. We then evaluate our method on several popular datasets of varying size and levels of corruption. Further we directly compare the numerical results of our experiments to that of the current state of the art unsupervised denoising model. While our proposed approach's experiments do not achieve a new state of the art, it provides an alternative method to unsupervised denoising and shows strong promise as an area for future research and untapped potential.

ACKNOWLEDGMENTS

Thanks to:

- My Mother and Father, whose hard work laid the foundation for me to pursue my dreams
- My advisor Dr. Ventura, for his advice, time, and patience with me on this thesis
- My committee members Dr. Khosmood and Dr. Fang, for their time and wisdom
- Dr. Sung-Jin Kim, Dr. Stephen Jenks, and Mr. Qian Zhang, who gave me my first chance
- The educators and mentors of my life, who continue to inspire me
- My innumerable friends, family, and neighbors who have all contributed to my well-being

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRO

It has long been a desire to present a distorted or noisy image to a computer, command it to enhance, and output a crisp and clear image. The benefits from such an advancement in technology are numerous and wide ranging, from simply increasing the aesthetics of an ill timed family photo, to allowing a CT scan to use less radiation shielding the patients from its potentially harmful effects. Cutting edge research in machine learning like deep convolutional neural networks have advanced the state of the art denosing capabilities under both traditional supervised learning and unsupervised learning problem scenarios [44, 45, 26]. This thesis aims to provide a novel method utilizing GANs to denoise images in a unsupervised scenario with a known noise distribution prior expressed analytically, and present and compare experimental results on a standardized set of images.

1.1 Motivation

The problem of image denoising has been the topic of study for decades intersecting many different disciplines like physics, statistics, mathematics, and computer science [35, 9]. While there are many techniques that attempt to solve this problem, an all encompassing solution to real world image noise has yet to be found. The causes of noise in images are just as plentiful as the solution space, with some causes being imperfect digital sensors, sensitive data transmission media/signals, and discrete color space just to name a few [35].

In almost every space where there exists image noise there is a desire to remove it. Some example applications where denoising can be beneficial are photo restoration for aesthetic purposes, low light photography, removal of noisy artifacts from telescopic images, and cleaning up medical images like CAT scans in order to see biological structures more clearly, to name a few. Historically this problem has fallen mostly in the mathematical/statistical realm working with bespoke sliding kernels, commonly Gaussian in design, that attempt to average surrounding pixels resulting in a pixelwise smoother image but with the undesirable effect of blurring or softness [34]. Notably "popular neural network technology has been applied in the field of image noise reduction in recent years" leading to new techniques that attempt to model the noise signal and remove it from data with state of the art accuracy [31, 52]. In the machine learning space, denoising was originally studied through a process where the model compares a reconstructed clean image of a noisy instance of the data, to a true clean image allowing for the model to learn its mistakes directly from the data. This implied that the data set contains pairs of noisy and clean images.

However previously mentioned machine learning solutions do not address the problem of how to restore and denoise images when presented with an exclusively noisy data set. Recent inventions like the generative adversarial network (GAN) and blind spot networks have made it theoretically possible to reconstruct images corrupted with a specific noise distribution, through a model that only has access to a domain of noisy images with only one image per target. This problem set of restoring images while only having access to single noisy images can be classified as unsupervised learning.

1.2 Challenges

A significant challenge with denoising in general is that the process frequently has undesirable side effects on the images. It is not uncommon to see that "noise removal introduces artifacts and causes blurring of the images" [35].

As stated previously machine learning and deep neural networks (DNN) have shown great promise and produce state of the art results on some denoising tasks but this too is not without its drawbacks. Deep neural networks and their derivatives like convolutional neural networks are notoriously difficult to train often having difficulty with gradient propagation and achieving a stable loss minimum [13]. More so the difficulty is exaggerated when the model only has noisy data to work with, presenting the scenario where the model does not have a way to directly back propagate on and therefore learn from its data.

Figure 1.1: Showing the discrepancy of measurement methods and human perception. Both the left and right photos have the same MSE compared to the original.

It is also important to note that the way in which we programatically measure the reconstructed images similarity with clean images is an imperfect science, and that

the way humans perceive images could be at odds with the mesurement methods. What exactly constitutes a better image and how to measure that is an active area of research with applications in image compression alongside denoising [48]. For example if using the popular mean squared error formula, a slight deviation in color over the entirety of the image almost unnoticeable to the human eye could cause the same score as a glaringly obvious hole in the middle of the image. Figure 1.1 is an example of this measuring to perception discrepancy as the left and the right photos have approximately the same MSE, and yet most observers would point to the left photo as more severly distorted.

1.3 Use of Generative Adversarial Networks

With the invention of Generative Adversarial Networks (GAN) in 2014 by Goodfellow et al. new possibilities have opened up on the way a model can be trained and what data is considered acceptable for a model to be trained on [15]. GANs are classified as generative models, meaning that GANs can use their network to create an instance of a distribution. This is in contrast to discriminative models that decide whether an instance is from a given distribution or not. From a high level perspective GAN's consist of two networks, a generator and a discriminator which compete against each other in a zero sum game where the generator attempts to fool the discriminator and the discriminator attempts to distinguish real data from generated data. During training a GAN essentially uses their discriminator as as loss function for their generator network, and has shown incredible flexibility in different task domains ranging from creating deep fakes, to texture synthesis, to supervised denoising [23, 20]. Research has also shown that using GANs for denoising can produce crisper and more defined images then a classic denosing system like wavelet transform and kernel methods [20] . Furthermore recent advances in denoising model architecture

like AmbientGAN have shown GAN's generative nature allows for new advances in unsupervised denoising [5].

1.4 Contribution

The contribution that this thesis aims to provide is to present a novel architecture, called MAP-GAN, for unsupervised denoising that utilizes a generative adversarial network and an analytic description of the noise generating distribution. We provide the mathematical theories on which our architecture is based and proceed to test our architecture on progressively harder datasets. We also set up an environment to directly compare our networks capability against the current state of the art unsupervised denoising work. Lastly we present our quantitative and qualitative findings, alongside associated ablation studies to allow readers to judge the effectiveness of this proposed scheme.

Chapter 2

RELATED WORK

Our research is just the latest in the interesting and storied field of image restoration and noise removal, drawing upon a long and sometimes confusing history of methods to remove noise under various circumstances of image quality and data set availability. This section aims to give the reader a brief taste of the research states throughout the years, the different techniques applied to try and find a solution to the denoising problem, and a history of the tools used in MAP-GAN.

2.1 Intro to Noise Removal

Image denoising has been a topic of research for decades with scientists from different fields like physics, math and computer science all approaching this problem through their own lens [35]. In early digital research prior to the wide adoption of deep learning, a popular approach to denoising was the use of filters over the images' pixels in the form of a kernel function to reduce the intensity of noise and attempt to smooth out the image. However these filters tended to have a blurring effect on the denoised images along with "Reconstruction artifacts, e.g., "ringing" effects or color speckles, [which] are inevitable because of high frequency loss in the blurred image" [51]. Kernels also tended to be hand crafted leading to extensive development times and narrow use cases. With these faults put together researchers were left dissatisfied with the results and actively searched for a better, more comprehensive solution.

With the advent of deep learning and convolutional neural networks, a new previously unreachable quality of denoising was able to be accomplished in a supervised learning

setting [6, 54]. Deep neural nets became a tool where a noise distribution could be learned from the image distribution and relatively effectively removed from a noisy image, if the original clean data was available. Open questions remained in the setting where only noisy images were in the available dataset, but with the the introduction of techniques like blind spot reconstruction and generative adversarial networks further advances could be made in the realm of unsupervised image denoising.

2.2 Classical

For the purposes of this paper we will consider classical denoising to be all of the methods and theories about denoising used prior to the advent and commonplace usage of deep neural networks. Classical techniques tended to focus on identifying specific characteristics of the noise in question and crafting a bespoke tool or strategy in order to acquire a representative clean image. Some popular techniques that proved effective and were well documented in literature are image filtering via a kernel function and wavelet analysis, both of which will be explored in further detail in this section [42, 35, 8].

Kernel filtering describes a wide swath of techniques that attempt to manage the tradeoff between removing noise and retaining the original signal of the image. Arguably the simplest kernel filter is a Gaussian kernel that effectively identifies the signal in each channel and takes the average of all the surrounding pixels. While effective at removing signal spikes, kernels of this class tend to produce a "blurred and smoothed image with poor feature localization and incomplete noise suppression" especially noticeable in places of sharp contrast like hard lines [2].

Nonconstant filters also exist such as the the Median Filtering Algorithm which proposes taking the median value of an $M \times N$ area of an image in order to attempt to

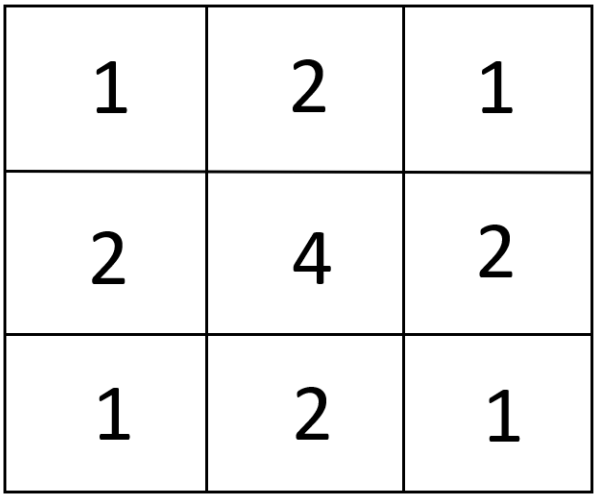


Figure 2.1: Example of a 3x3 gaussian kernel

denoise while retaining line sharpness and reducing the blurring effect seen in other filters [18].

The previously mentioned kernel techniques are some of the simplest to apply and conceptualize, but other authors have modified and used more advanced variations of kernel filtering to varying success. An example would be the former state of the art BM3D algorithm which first finds similar patches in an image, stacks the patches to form a 3 dimensional array and computes a weighted average along with threshold and Wiener filtering and returns a final estimate [10].

The wavelet category of classical denoising has also shown great effectiveness and popularity among the image processing community. According to Zhang and Gunturk, wavelet thresholding is when "a signal is decomposed into its approximation (low-frequency) and detail (high-frequency) subbands" [53]. Wavelet denoising can be as simple as multiresolution thresholding of the wavelet coefficients as seen in [11], or more complicated statistical analysis of the wavelets as seen in [43, 38].

It is also important to note that these techniques are not mutually exclusive and can often be seen alongside each other. An example of this is kernel filtering alongside

multiresolution filters where images are expanded and shrunk in order to highlight the difference between the noise signal and the original image signal, such as in the paper "Multiresolution Bilateral Filtering for Image Denoising" [53].

2.3 Deep

Before diving into denoising in the deep learning space it is important to first define what exactly deep learning means. One such definition of deep learning systems proposed by some of the top researchers in the field is "representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level" [27]. Further information on the underlying mechanisms and math behind deep neural networks, autoencoders, and generative adversarial networks can be found in the background section of this thesis.

Deep learning has revolutionized a large number of fields from natural language processing, to computer vision, to stock market prediction, and signal denoising. In the field of computer vision, deep learning made an unquestionable impact when in 2012 Krizhevsky, Sutskever and Hinton's deep convolutional network architecture placed first by a wide margin in the ImageNet challenge achieving never before seen image categorization success rates [25]. Innovations in novel network architecture, and new faster hardware to support them allowed for great advances in computer vision in the early 2010s. But it is important to note that the much of the deep learning image research going on in this time frame was primarily concerned with computer vision and discriminative models such as image classification and object segmentation. It was not until 2015 with the use of deep autoencoders that deep learning and denoising

became intertwined tasks, and was used very successfully as in Liang and Liu's Stack Denoising Autoencoder [30].

According to M. A. Kramer, an autoencoder "operates by training a feed forward neural network to perform the identity mapping, where the network inputs are reproduced at the output layer" after passing though a lower dimensional bottleneck layer [24]. Over time both autoencoders and the more advanced sibling variational autoencoders have shown to be good tools when it comes to denoising, used in many different methods and architectures like standalone, stacked, or used as a pretraining tool for convolutional neural networks. However much like the kernel filtering technique, a disadvantage of autoencoders and variational autoencoders is that its objective function tends to create a blurry output[20].

In 2014 Ian Goodfellow created an adversarial method of training deep generative models called the generative adversarial network (GAN) that provided a new approach to identify and sample the distributions of image sets[15].

Within the last 5 years research into GANs used for denoising applications has been increasing as state of the art results have been produced using this architecture. Some early works have utilized GAN's to replicate a noise distribution to enlarge a dataset so a more traditional CNN model can train on many simulated clean/noisy image pairs [7]. Other methods rely on the GAN to remove the noise itself from a dataset of clean and noisy paired images in a supervised learning setup [50].

2.4 Unsupervised Noise Removal

Recently research interest has increased in the problem posed when there exist only unpaired noisy images in the training dataset distribution. Training a machine learn-

ing model under this scenario is described as self-supervised learning. There have been several notable attempts at this poised problem, with most successful methods employing a U-net style deep neural network with a traditional loss function such as the l_2 loss in an attempt to find statistical properties to remove noise without an adversarial process [49, 29].

The paper Noise2Noise trains a model on pairs of noisy images of the same scene, each with unique sampling of the noise distribution. This allows the network to learn a mean of all plausible explanation images for its output, but still maintain relative sharpness. Their model is hinged on the fact that they can "in principle, corrupt the training targets of a neural network with zero-mean noise without changing what the network learns" [29]. While producing excellent results it is important to remember that their framework is based on having two independently realized noisy images which is not feasible in many cases, and arguably semi supervised.

Noisy-as-Clean takes a similar approach but instead of requiring two noisy images, they require only one and create a second noisy image by adding additional noise to the original noisy input. Noisy as clean hinges on the assumption that the expectation of the signal of the clean image is much greater than that of the additive noise, and therefore their simulated noisy image has similar expectation with the observed noisy image [49].

The authors of AmbientGAN have shown that it is possible for a GAN to learn the clean image distribution from only corrupted samples, but their model is a non-conditional generator and samples random examples from the clean image distribution [5]. It is also important to bring up the work of researchers from France in their paper "Unsupervised Adversarial Image Reconstruction" where they use a GAN to attempt to denoise a specific image but do not make any assumptions on the noise generating distribution and so the results leave something to be desired[36].

To the best of our knowledge the current state of the art in singe image unsupervised denoising is the paper from NVIDIA named "High Quality Self-Supervised Deep Image Denoising". They utilize a convolutional blind spot network in conjunction with a residual U-net architecture to achieve impressive results. More so they also utilize a known noise model in order to further induce their network to find the appropriate clean image [26].

Chapter 3

BACKGROUND

In this section we will aim to give the reader a quick refresher on the common causes of noise, the mathematical underpinnings of deep learning, and the architecture and math behind generative adversarial networks. The goal of this section is to not to give the reader a comprehensive overview of what is arguably a enormous section of math, computer science, and statistics literature but rather to provide just enough information at a high enough level so a reader with limited background in machine learning can conceptualize the method that we present in later chapters. We would encourage a reader new to this subject to find resources elsewhere to form a strong base such as "Deep Learning" [14]. We also assume a beginner level understanding in image representation and image processing.

3.1 What is Noise

This section aims to give a overview what of the common causes of noise in images are, and how they are mathematically modeled. According to Verma and Ali "Noise is a random variation of image intensity and visible as grains in the image" [46] and is mostly an undesirable quality that users and developers wish to be not present in the image. Most noise can be broken down into three characteristic types, impulse noise, multiplicative noise, and additive noise with additive noise in the form of a Gaussian distribution being the most common.

3.1.1 Impulse Noise

A commonly seen example of impulse noise is known as salt and pepper noise, where strong dots appear throughout the image. This type of noise can be caused by several reasons such as dust particles on the image sensor, or over heated faulty electrical components [46]. A further example of a process that can cause salt and pepper noise is when images are transmitted over noisy digital links, where each bit of a pixel has a certain probability of being flipped [4]. According to [47] salt and pepper noise can be approximately modeled by the distribution of

$$X(i,j) = \begin{cases} r_{max} & \text{with probability } a \\ r_{min} & \text{with probability } b \\ x(i,j) & \text{with probability } 1-p \end{cases} \tag{3.1}$$

such that $a + b = p \leq 1$, where r_{max} is the maximum luminance value a pixel can take, and r_{min} is the minimum luminance value, $x(i,j)$ is the noise free luminance value of the pixel at (i,j) and $X(i,j)$ is the luminance value at pixel (i,j).

3.1.2 Multiplicative Noise

Multiplicative noise is noise that can be decomposed as

$$f(\cdot) = g(\cdot)q(\cdot) \tag{3.2}$$

where $g(\cdot)$ is the clean image image, $q(\cdot)$ is the noise component and $f(\cdot)$ is the resultant noisy image [4]. An example of noise that can be considered multiplicative

is speckle. Speckle noise is seen in active radar and synthetic aperture radar, due to coherent processing of of back scattered signals from numerous down field points or when the size of the item being scanned is less than a radar's image processing unit [46, 37]. The negative exponential is used to model speckle noise and would take the place of $q(\cdot)$ in the example above. The negative exponential distribution can be modeled as

$$p(x) = \lambda e^{-\lambda x} \tag{3.3}$$

3.1.3 Additive Noise

Similarly additive noise is noise that can be decomposed as

$$f(\cdot) = g(\cdot) + q(\cdot) \tag{3.4}$$

Where $f(\cdot), g(\cdot), q(\cdot)$ maintain the same meaning as in the multiplicative noise section. Gaussian noise is an example of additive noise, and accurately represents commonly seen noise in nature. Gaussian noise is often found in traditional film photography and can be attributed to imperfections in the sensor and low light conditions. A normal distributions pdf would be

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)} \tag{3.5}$$

In this case $p(\cdot)$ would take the place of $q(\cdot)$ in the additive noise equation above.

3.2 Background of Deep Learning

This section aims to give a very brief background on the definition of deep learning, and the conceptual principles of it. Aimed to be a refresher for those who have learned about it at one point in the past.

3.2.1 What is Deep Learning

Deep learning is arguably one of the hottest computer science topics currently in media and popular science. But what is deep learning? According to Guo et al. "Deep learning algorithms are a subset of the machine learning algorithms, which aim at discovering multiple levels of distributed representations" [16]. Deep learning consists of multiple, possibly many layers of nonlinear functions that are composed together in a network that when viewed as a graph have a deep representation. Deep learning is usually supported by artificial neural networks, usually shorthanded to neural networks (NN).

Neural networks can come in several different architecture configurations and styles like the recursive neural net, but four our purpose we will be primarily concerned with feed forward neural networks. Feed forward means that the flow of data throughout the network is unidirectional with each layer getting activated only once while processing an instance data. Neural networks consist of many layers of simple nonlinear functions called neurons that receive a weighted input from previous neurons, compute some form of calculation based on its internal parameters, and then output a weighted real value to one or more child neurons. Each neuron usually consists of a linear portion which weights and computes a summation of the inputs from the previous layer of neurons and a nonlinear function, such as a tanh, sigmoidal function,

or rectified linear unit among many other possibilities. The exception to this is the first and last layers of the neural network, with the first layer being called the input layer and receiving data from an external source, like an image, audio, table, or any other form of information. The final layer is the output layer that produces an item in the target space, weather that be a binary output, a series of binary outputs, a real valued number or a complex structure like an image. Lastly the middle layers will often be called hidden layers as they have no contact with and so are hidden from the external environment.

A network is trained in a supervised fashion when it has access to the correct outputs of a function for a given input, i.e. if a network has access to the labeled data (x, y) for some function $f(x) = y$. A network is considered unsupervised if it only has access to the inputs and not to the outputs, or if it has access to exclusively unpaired data.

A neural network is trained via a method called backpropagation, which finds the gradient of each weight in the network based on an input and its loss, and updates the weight with the goal of finding minimum loss. It essentially ammounts to performing the chain rule over a stack of layers up until the target weight, starting from the output layer and finishing at the input layer. This process is an efficient procedure because previous layers derivatives do not need to be recalculated, and for each layer only the current layer's derivatives need to be calculated.

3.2.2 Math of Deep Learning and Backpropigation

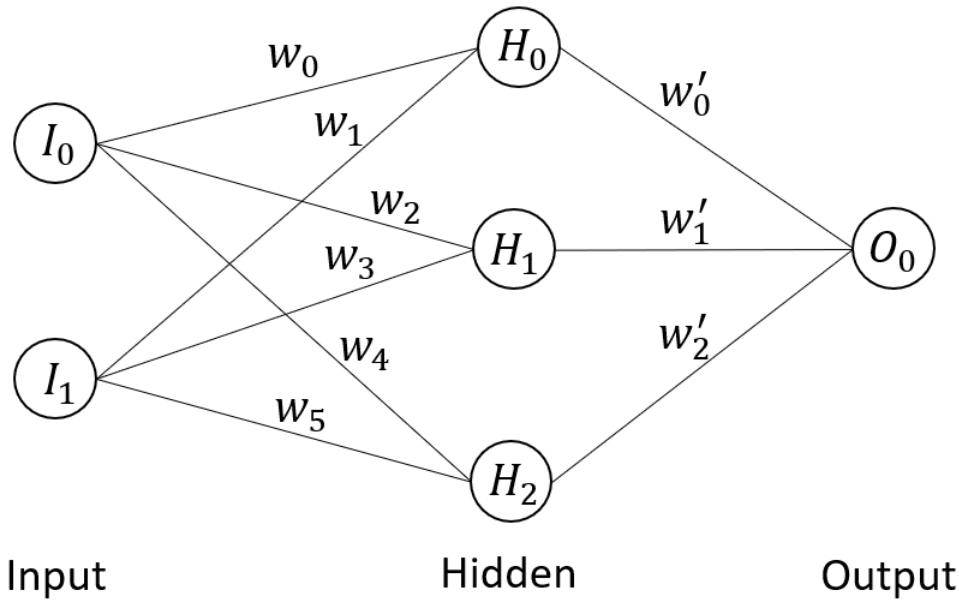


Figure 3.1: Example of a neural network with 2 input neurons, one hidden layer of 3 neurons, and one output neuron

In this section we will take a look at one forward and backward iteration of the target neuron H_0 in Figure 3.1. In a forward iteration the steps are

1. An input is given to the input layer I_0 and I_1
2. H_0 uses its weights to take a linear combination of its inputs resulting in $h_{intermediate} = I_0 \cdot w_0 + I_1 \cdot w_1$
3. A nonlinear function $f(\cdot)$ is applied to the output of the previous summation resulting in $H_{0\,\text{out}} = f(h_{intermediate})$
4. $H_{0\,\text{out}}$ is then passed to the output neuron O_0

Continuing with the other process in the full network, H_1 and H_2 follow steps 2, 3 and 4 with their own respective weights. O_0 then follows step 2 with its weights

$w_0^{'}, w_1^{'}, w_2^{'}$, and optionally follows step 3 and applies a nonlinear function resulting in $O_{0\text{ out}}$. Finally $O_{0\text{ out}}$ is either passed to a loss function during training, or used as a final prediction during testing.

In the backpropagation stage for H_0 we want to find the gradients of w_0 and w_1 for a given input value and loss function. We first assume that we have the upstream derivative U', which is found in the same manner we are about to lay out for H_0. We then know that if we look at H_0 in isolation we have the function of $U(F(G(i_0, i_1)))$ where F is the nonlinear function and G is the linear portion $I_0 \cdot w_0 + I_1 \cdot w_1$. In order to get the gradient of w_0 we want to find

$$\frac{\partial U}{\partial w_0} = \frac{\partial U}{\partial F} \cdot \frac{\partial F}{\partial G} \cdot \frac{\partial G}{\partial w_0} \tag{3.6}$$

With U' known it becomes a simple mater to find $\frac{\partial F}{\partial G}$ and $\frac{\partial G}{\partial w_0}$, which amounts to just finding the derivative of this layers nonlinearity and the derivative of a linear function.

With ∇w_0 found we can now perform an update on the weight in order to lessen the loss. This update is found by

$$w_0^{new} = w_0^{old} - a \cdot \nabla w_0 \tag{3.7}$$

where a is a value known as the learning rate and allows the user to control the speed at which the weight changes. The same process is applied to all other weights in the network to complete the backpropigation step.

3.2.3 Convolutional Neural Networks

Convolutional neural networks take advantage of the convolution function in layers of the networks. In deep learning a convolution is a mathematical operation that utilizes sliding a discrete weighted kernel over an instance of discrete data, and measures the interaction of the kernel and the data at each point. Convolutional neural networks when used with images provide numerous benefits. The first benefit is quicker training in high dimensional space; a convolutional kernel size is generally only a small fraction of the size of the input and so when calculating the gradients it is significantly faster using much less memory due to the decrease in the number of neurons.

Figure 3.2: Example of a convolution operation. No padding and step size of 1

A powerful property that allow CNNs to be effective is the spatial locality of images, meaning that pixels values are not independent of their neighbors values. Convolutional kernels take advantage of this by using their shared weights to learn patterns within an image in one location, and activating more strongly if that pattern is present in a separate location.

Another advantage of convolutions is that fully convolutional neural networks allow for varying input size of images, the sliding nature and reusable kernels allow for a much more flexible interface than a non-fully convolutional network. Further in practice it has been shown that convolutional neural networks tend to be a much more

stable to train then fully connected alternatives, and the lower number of weights tend to help prevent over fitting on the training data.

When taking a convolution, the convolutional kernel is slid across the image in varying step sizes. At each location a summation of the weights multiplied by the image pixel values is taken. Next like a fully connected network, a non linearity is applied to the summation value. If an image is color and contains a multichannel z-axis then the kernel is also multi channel having a depth the same size as the z-axis, and includes all the z-channels in the total summation. Often there are many convolutions applied at each step leading to an output that has a depth dimension equal to the number of different convolution kernels. These multiple convolutions can be done in parallel and are performed very efficiently.

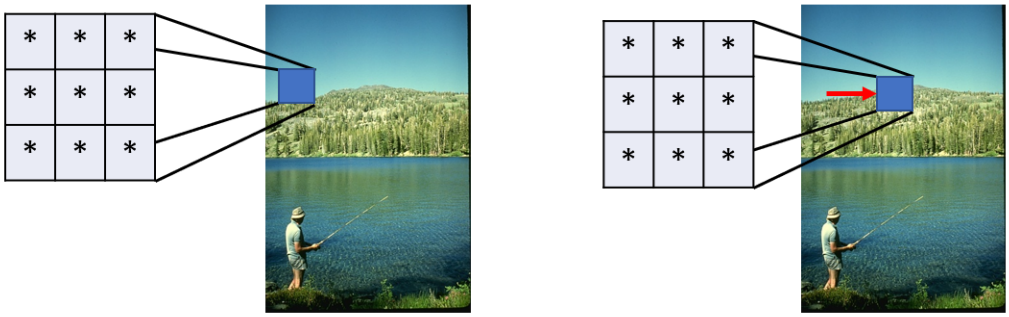


Figure 3.3: Sliding a convolution kernel over an image

After the forward pass, the backpropagation of the convolutional kernels is essentially the same as previously described with the gradient for each kernel being calculated at each convolutional usage, not just once per image. In practice, convolutions and traditional fully connected neural networks often seen working in tandem, taking advantage of the benefits of each network architecture style. This is commonly seen in image classification networks, where the first several layers are exclusively convolutions and the last layers are a fully connected network.

3.3 GAN Background

Created in 2014 by Ian Goodfellow, GANs are considered to be generative models meaning that they can create an item from a distribution given an input, as compared to discriminative models that decide if an item is in a distribution or not. The method of training a GAN is novel and represents a large independent advancement, but GANs do not exist in a vacuum and have several precursors leading to their creation.

Autoencoders are also considered to be a generative neural network style and are generally seen as an example of a precursor to GANs. Today it is common to see network architectures used in autoencoders, utilized by GANs, specifically the choke point style architecture characteristic of modern autoencoder networks.

3.3.1 Autoencoder

The history of autoencoders is a bit convoluted but is generally attributed to first being created in the 1980s by Rumelhart et al. [40, 3]. Autoencoders are a generative machine learning algorithm that utilizes a neural network with a choke point to express input in a compressed distribution, and inversely express the input distribution from the compressed distribution. Ideally points drawn from the compressed distribution not represented by an item from the data distribution should be able to generate a novel item from the input distribuion. While applicable to general input we will be looking at the image domain specifically.

During the training process autoencoders take some distribution of images as input and use a network to then represent the image in a lower dimensional space as an intermediate distribution. From that lower dimensional distribution space it then uses a second network to attempt to recreate the input image in the original dimensional

space. This recreated output is then measured directly against the input, and a loss like a mean squared error is applied.

At testing time, the first half of the network is removed and the second half of the network, and the second half is used as a generating function. From here the generating portion is given either random input in the lower dimensional space, or a point found by interpolating between two known images lower dimensional representations.

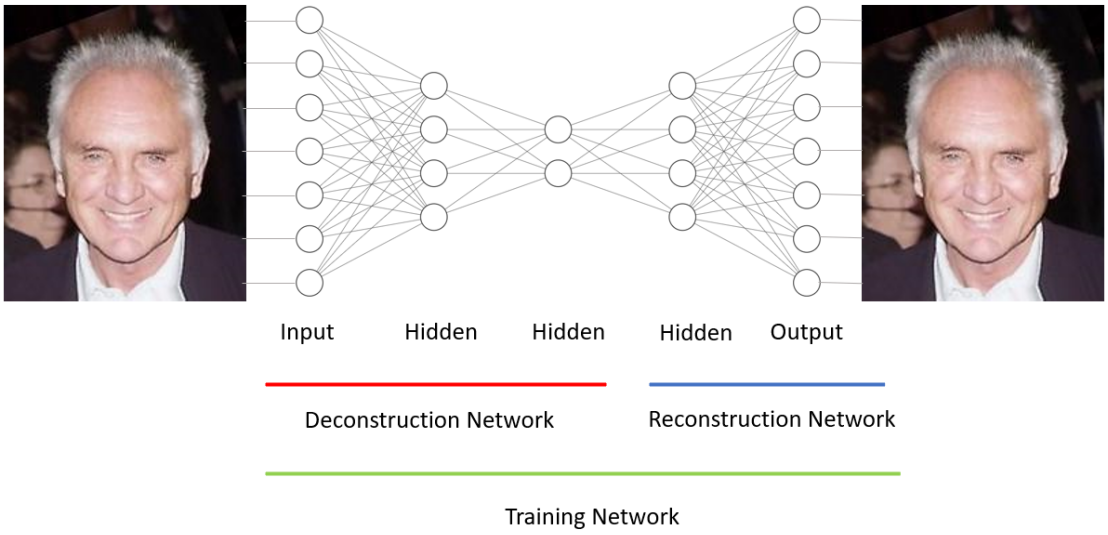


Figure 3.4: Example of an autoencoder's architecture

This is the simplest version of autoencoders, with more more complicated versions like the varaiational autoencoder attempting to solve issues found in the original formulation, primarily that a large portion of the lower dimensional space maps to junk outputs [22]. While autoencoers and their variants have seen much success, their outputs all still have critical limitations in some regards, most notably that the outputs are often blurry, dont represent fine details well, and that despite best efforts there remains a large portion of the lower dimensional distribution creates unintelligible output.

3.3.2 GAN Architecture

A GAN is a class of learning architecture that consists of two separate networks, a generator and a discriminator that compete in a type of 0 sum game with each other, learning from the others mistakes and successes. In the simplest configuration a generator is given a seed input from a random distribution and uses its network to attempt to produce an item from the target output distribution. At this point the output of the generator is passed to the discriminator.

The discriminator takes input from the real data distribution and from the output of the generator. Its goal is to discriminate between the generated output and the real data identifying which is real which is fake.

The discriminator's loss is calculated in the traditional way, via some sort of binary cross entropy or other equivalent mechanism. The generator on the other hand uses the discriminator as its loss function, essentially attempting to fool the discriminator into believing that the output it produced was from the real data distribution.

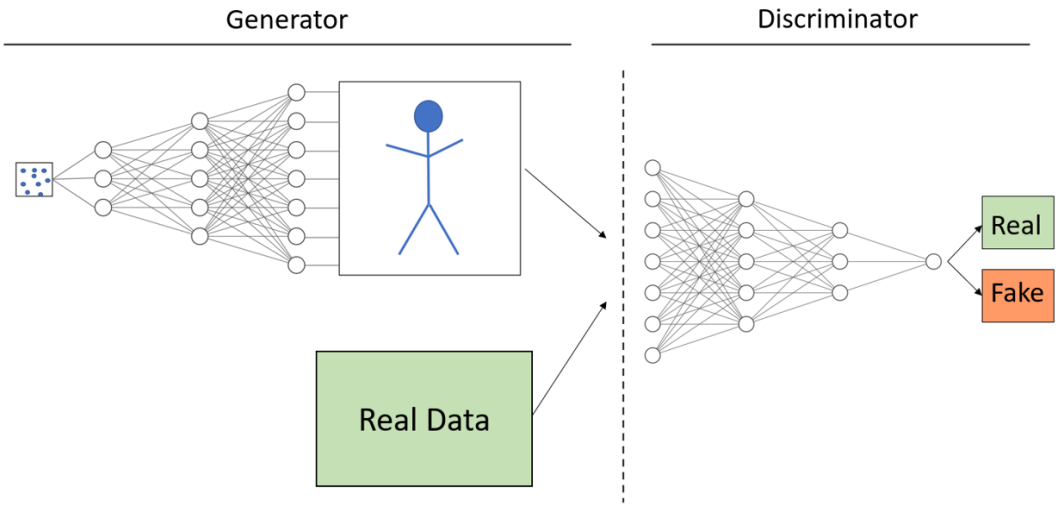


Figure 3.5: High level overview of a GAN.

During training, the generator and the discriminator take turns trying to fool each other; after each turn they update their respective networks based on the losses incurred. The discriminator updates itself like any other network via back propagation, however the generator performs backpropagation but through the entirety of the discriminator. The generator therefore finds its gradients over each pixel of the output from the discriminators response. From that point the generator can back propagate through its own network accordingly. Because the discriminator and generator are competing against each other, when one gets better the other would incur a larger loss assuming the ladder stays the same. This can be interpreted as a min max game where,

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{real}}}[\log D(x)] + \mathbb{E}_{z\sim p_{\text{fake}}}[\log(1 - D(G(z))] \tag{3.8}$$

It can be shown that under ideal training circumstances a Nash equilibrium between the generator and discriminator is found, where they both achieve a 50% success rate among each other. It can also be shown via Ian Goodfellow's paper [15] that the generators minimum loss is found when the generated distribution is equal to the true data distribution. The discriminators minimum found when it discriminates all it's data correctly.

Chapter 4

METHODS

In this section we will discuss the problem scenario and the theory behind the MAP-GAN method of denoising images.

4.1 Problem Scenario

Our problem scenario is, as expected, very similar to that proposed in [36]. Let $X \sim P_X$ be a signal that we want to recover, and assume further that we can only access this signal through a noisy or otherwise erroneous measurement. Let that erroneous measurement be $Y \sim P_Y$. Let F be a probabilistic corruption/measurement function that accepts items from X and outputs their measurements in Y. Lastly assume that we only have access to a subset of the noisy distribution $\hat{Y} \subset Y$.

In other words we can say $Y = \{F(x)|x \in X\}$ and given a set of noisy images $\hat{Y}$ and the measurement function $F(\cdot)$, for any $y \in Y$ we would like to produce $\hat{x}$ s.t. $y = F(\hat{x})$.

While our approach is applicable to a wide array of inverse problems, for the rest of this explanation we will assume that items from X and Y are digital images.

4.2 Approach

It can be seen that this problem nicely presents itself in a Bayseian framework. We would like to find

$$\hat{x} = \arg\max_{x} p_{X|Y}(x|y). \tag{4.1}$$

So applying Bayes rule we can formulate our problem as

$$\hat{x} = \arg\max_{x} \frac{p_{Y|X}(y|x) \cdot p_X(x)}{p_Y(y)} = \arg\max_{x} p_{Y|X}(y|x) \cdot p_X(x) \tag{4.2}$$

$p_{Y|X}(y|x)$ is known as the likelihood while $p_X(x)$ is known as the prior, and $p_{Y|X}(y|x)p_X(x)$ is known as the posterior. Finding the x that maximizes the posterior is known as finding the Maximum A Posteriori (MAP) estimate. The likelihood represents the certainty that y is the result of a corruption measurement on x while the prior represents the certainty that x is part of the true image distribution.

Our solution utilizes a Generative Adversarial Network to identify the correct image, similar to [36] we devise the problem of finding the a generator $G : Y \to X$ such that for each input y the generated output is its associated MAP estimate $\hat{x}$. However unlike previous work we use an analytic form of the likelihood as will be shown in upcoming sections. We can then describe our objective as finding

$$\hat{G} = \arg\max_{G} \mathbb{E}_{p_Y}\{\log p_{Y|X}(y|G(y)) + \log p_X(G(y))\} \tag{4.3}$$

The likelihood is now $\log p_{Y|X}(y|G(y))$ and the prior is $\log p_X(G(y))$. To arrive at equation 4.3 from equation 4.2, we is replaced x with $G(y)$, and took the log of the equation.

4.2.1 Expressing the Likelihood

In our method we assume that the noise is *i.i.d.* pixel wise and drawn from a distribution that can be expressed analytically. We can then write the likelihood $p_{Y|X}(y|x)$ in its analytic form as $p_{Y|X}(y|x) = p(y_0|x_0) \cdot ... \cdot p(y_n|x_n)$ and are therefore able to maximize over it.

For example if we have additive zero mean Gaussian noise with standard deviation σ, we would have

$$p(y_i|x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-||y_i - x_i||^2}{2\sigma^2}} \tag{4.4}$$

and so if we once again substitute G for x we get

$$p(y_i|G(y)_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-||y_i - G(y)_i||^2}{2\sigma^2}} \tag{4.5}$$

which can then be used to express

$$p_{Y|X}(y|G(y)) = p(y_0|G(y)_0) \cdot ... \cdot p(y_n|G(y)_n) \tag{4.6}$$

as the analytic form of the likelihood that we can optimize from. Note that equation 4.6 would force G to produce outputs similar to the input of y and not just some random instance from the distribution of X. This can then be substituted into the likelihood term of equation 4.3.

4.2.2 Expressing the Prior

The prior on the other hand is intractable to formulate explicitly and thus we must turn to machine learning for the solution. As the literature has shown an unconditional GAN can be used to find a prior distribution P_X from a (traditionally random) input distribution P_R. So given $r \in R$ a GAN should be able to find $G(r) = x$ where the probability of x being drawn from P_X is great. This is essentially the approach taken by AmbientGAN to express the prior.

While MAP-GAN is similar to AmbientGAN in some respects, its import to remember that we are opting to denoise specific images not randomly ascertain an example from the target distribution X. The enforcement that the the output of $G : Y \to X$ is similar to its input is taken care of by the likelihood, so for the prior portion of equation 4.3 we want to optimize that the output of G is an item from the distribution of X.

At this point it is worth talking about the GANs discriminator D and how it relates to the loss function of the generator. We are trying to find the discriminator

$$\hat{D} = \arg \max_{D} \mathbb{E}_{p_Y} \{\log D(y) + \log(1 - D(F(G(y))))\} \tag{4.7}$$

where $\hat{D}$ is the optimal discriminator. This allows us to roughly apply lemma 5.1 and 5.2 with modifications from AmbientGAN [5] to show that by optimizing the discriminator we will achieve an optimal generator within our solution architecture. For our purposes an optimal generator means that it can provably recover the original clean distribution.

Theorem 1. *Let P_X be the clean distribution and let P_Y be the measured noisy distribution over X, and let F_Θ be the measurement distribution parameterized by Θ.*

Further assume that for the given noise generating distribution parameterized by Θ *there is a unique probability distribution,* P_X *that induces the distribution* P_Y *from* F_Θ*. Then if the discriminator* D *is optimal, the generator* G *is optimal if iff* $P_{G(Y)} = P_X$*.*

Proof of Theorem 1. The original GAN paper [15] states that if the discriminator is optimal then

$$\hat{D}(y) = \frac{P_Y(y)}{P_Y(y) + P_{F(G(Y))}(y)} \tag{4.8}$$

which therefore implies that the Generator is optimal iff $P_{F(G(Y))} = P_Y$ (see [15] Theorem 1 for details). Further because we are assuming there is a unique probability distribution, P_X,that can induce the noisy distribution P_Y from F_Θ, it can be deduced that $P_{G(Y)} = P_X$.

□

The latter portion now involves showing that F_θ produces a unique probability distribution from its input. This must be taken on a case by case basis for the noise producing distribution. If F is Gaussian this can be shown by theorem 5.2 from Bora [5] and so we direct the reader to look there for further confirmation.

4.2.3 All Together

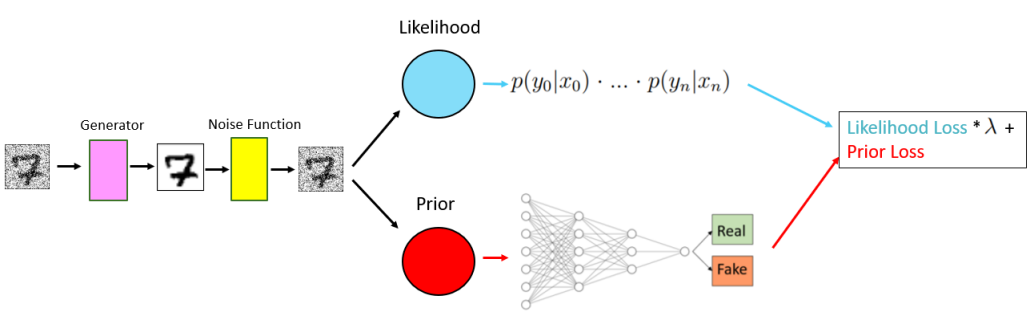


Figure 4.1: MAP-GAN scheme

In section 4.2.1 we demonstrated that it is possible to find an expression likelihood that can be optimized inside of equation 4.3. Similarly in section 4.2.2 we demonstrated that it is feasible to encourage a generator to produce examples from a clean distribution while only having access to noisy data.

In creating a loss to enforce the likelihood term we must examine the unique analytic expression of the noise distribution. For Gaussian noise we can see that the loss is expressed in the form of

$$Loss_{Gaussian\_Likelihood}(G) = \mathbb{E}_Y\{||y - G(y)||^2\} \tag{4.9}$$

For the prior generator loss, remembering from equation 4.6 that $\hat{D}$ is the optimal discriminator we see that

$$Loss_{prior}(G) = \mathbb{E}_Y\{1 - \hat{D}(F(G(y)))\} \tag{4.10}$$

With both penalties equation 4.10 and equation 4.9 put together our total objective function can finally be expressed as

$$Loss_{total}(G) = \lambda \cdot Loss_{Gaussian\_Likelihood}(G) + Loss_{prior}(G) \tag{4.11}$$

The algorithm of the MAP-GAN training procedure is described below.

Algorithm 1: MAP-GAN Training Procedure

Initialize weights of G and D ;
for i *from* 1 *to* n_{epochs} **do**
  for j *from* 1 *to* $n_{batches}$ **do**
    for k *from* 1 *to* $n_{D\ steps}$ **do**
      $Y_{real} \leftarrow$ sample_batch(Y);
      $Y_{fake} \leftarrow f(G(\text{sample\_batch}(Y)))$;
      update_discriminator(Y_{real}, Y_{fake});
    end
    $Y_{fake} \leftarrow f(G(\text{sample\_batch}(Y)))$;
    update_discriminator(Y_{fake})
  end
end

Chapter 5

DATASET

5.1 Collections

In this section I will briefly go over the main datasets that were used to both develop and to evaluate the efficacy of this network. The three training datasets used are the MNIST data set, the CelebA dataset, and the ImageNet data set. The datatsets are ordered in increasing complexity of both image size and content, and allows us to get a clear picture of the ability of our method. The testing datasets used are KODAK and BSDS300. All datasets mentioned will be explored in greater detail in this chapter.

5.1.1 Mnist

The MNIST dataset is a collection of handwritten digits, numbers 0-9, represented as single channel greyscale images. MNIST images were collected in 2010 from the NIST Special database by LeCun et.al. [28]. Due to their relative simplicity these 28x28 pixel images are often used as the first dataset from which to test the feasibility of a machine learning project on. There are in total 60,000 training images used.

5.1.2 CelebA

The intermediate difficulty dataset used is the CelebA collection of images. According to the official website and provider of this dataset, "CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset with more than 200K celebrity images,

Figure 5.1: MNIST examples

each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter" [32].The specific variant used for experimentation is the Aligned and Centered CelebA dataset which took the liberty of aligning and centering the celebraty faces. All images are 218x178 RGB in format.

Figure 5.2: CelebA examples

5.1.3 Imagenet

For our large and complex image dataset we chose to use the ImageNet validation dataset, as it is the standard used to train denoising models in papers like [26, 29], and allows us to accurately rank our denoising capability. The creators of ImageNet describe it as an "image dataset organized according to the WordNet hierarchy", comprised of hand annotated image files from throughout the internet [41]. With over 100,000 synsets listed by Wordnet, and with ImageNet aiming to have 1000 images per synset, there are a total of millions of labeled images represented by the

full dataset. The validation dataset is comprised of approximately 50,000 images of various resolutions but all of which are represent by the RGB color space channel.

Figure 5.3: ImageNet examples

5.2 Testing Only

The datasets in the sections below are only for testing, and are commonly used testing standards in all sorts of applications from image compression, to colorization. More relevant to us is that they are used by peers to benchmark the capability of their self-supervised denoising networks.

5.2.1 Kodak

The KODAK dataset represents 24 photos of size 768x512 with 3 channel RGB colorspace [12].

Figure 5.4: Kodak examples

5.2.2 BSDS300

The BSDS300 dataset consists of 300 photos of size 481x321 with 3 channel RGB color space [33]. For our network simplicity we removed the rightmost column and the bottom most row to have resultant images of size 480x320.

Figure 5.5: BSDS300 examples

Chapter 6

EXPERIMENTS

In this section we will present the results of our experiments. We will start by presenting proof of concept results on the MNIST dataset, followed by results on the CelebA dataset. We will then examine the results of our model trained on the ImageNet validation dataset when attempting to denoise photos from the KODAK and BSDS300 dataset, and compare our results against the current state of the art models. Lastly we will perform an ablation test with our model to see if our network is reaching its full capability.

6.1 Measurement

In order to measure the results of this networks output we will be using the peak signal-to-noise ratio (PSNR). It is a metric that is commonly used to compare compression quality, and is well positioned in identifying the discrepancy in noisy and clean data. Further it is the standard measurement used by other papers in the denoising field and so along with using the correct image dataset and noise function, it allows us to directly compare our results to other literature in the field [29, 26]. Notably PSNR uses the decibel (dB) scale. PSNR is calculated as

$$PSNR = 10 * \log_{10}\left(\frac{R^2}{MSE}\right) \tag{6.1}$$

where R is the difference between the maximum and minimum value that a pixel may take, and MSE (mean squared error) is calculated as

$$MSE = \frac{\sum_{m,n=0}^{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N} \tag{6.2}$$

Where I_1, I_2 are the images being compared, and M and N are the dimensions of the images.

6.2 Preliminary Experiments

This section covers a progression of experiments showing the competency and possibility of the MAP-GAN architecture. The noise distribution used throughout the experiments is a *i.i.d.* pixelwise additive Gaussian with zero mean and a standard deviation that will be noted in each experiment. The GAN's generator network design that is used in all the experiments is a customized version of High-Quality Self-Supervised Deep Image Denoising network (HQSS) [26]. This network in turn is a modified version of the five-level U-Net [39] architecture used by Lehtinen et al. [29], as stated in HQSS [26]. The discriminator is a convolutional network of our own design. The exact network details will be outlined in the appendix of this thesis.

6.2.1 MNIST Experiment

We first apply our system on the MNIST dataset. For this evaluation we train two separate models on the full training dataset. One model has been given a dataset whose Gaussian corruption has standard deviation of $\sigma = 32$ and the other that has standard deviation of $\sigma = 64$, both with zero mean. The optimal hyper parameters were found via cross validation and both of these models were trained for 50 epoch each.

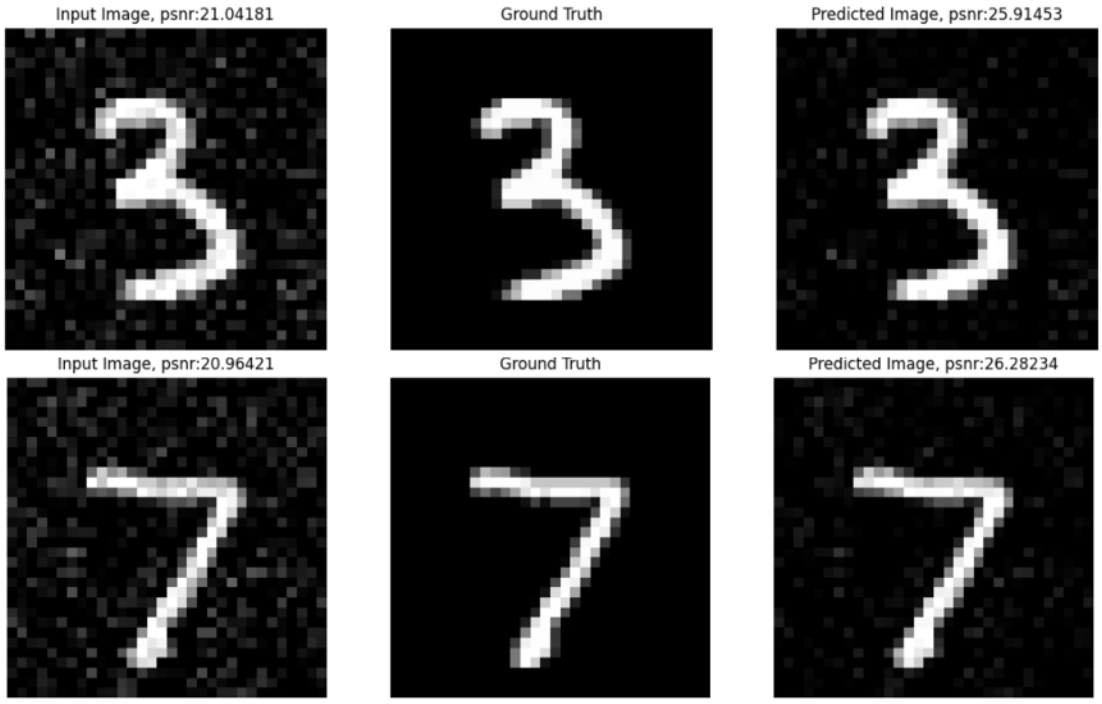


Figure 6.1: Results of network denoising MNIST where $\sigma = 32$

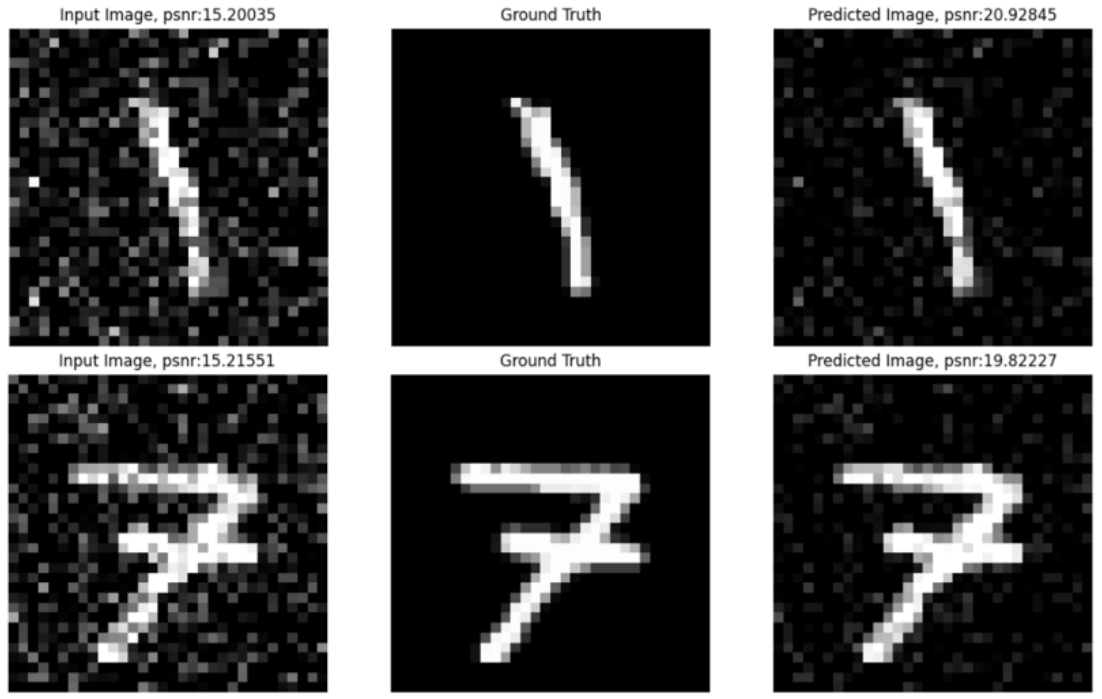


Figure 6.2: Results of network denoising MNIST where $\sigma = 64$

6.2.2 CelebA

The next portion of evaluation of the MAPGAN architecture is on the CelebA dataset, a significantly larger dataset both in terms of unique images and the size of the images dimensions. Further that images present a more complex image distribution and are in color. This then allows us to examine the architecture under a more complex image distribution with a nontrivial target space. This dataset was corrupted with Gaussian noise, with a standard deviation $\sigma = 25$.

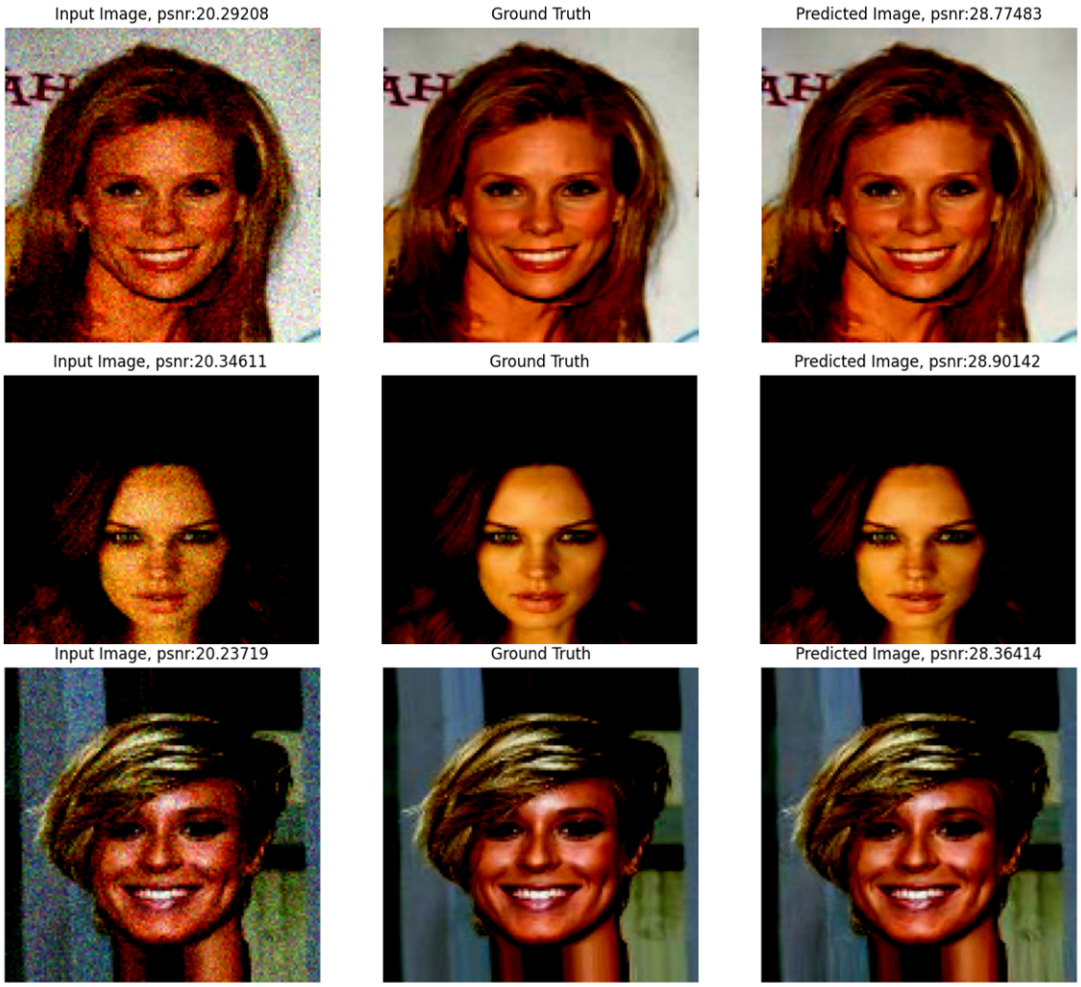


Figure 6.3: Results of denoising CelebA photos where $\sigma = 25$

The model was trained for 30 epoch before stopping, reaching its maximum denoising. Here we can see significant improvement over the original noisy images. Notice that facial features remain sharp after denoising has been applied.

6.3 Imagenet

Here we will train our model on a portion of the imagenet dataset and compare our results to the state of the art unsupervised denoising methods. The current state of the art unsupervised denoising method, to the best of our knowledge, is High-Quality Self-Supervised Deep Image Denoising (HQSS) by Samuli Laine et al. who use a blind-spot method in conjunction with a deep convolutional neural net.

Like them our model will be trained on the 50,000 images contained in the ILSVRC2012 (ImageNet) validation set. In order to get an accurate comparative metric, for our testing set we will use the popular KODAK and BSD300 datasets, the same as in HQSS. The number of images in these datasets is quite small so in order to get an accurate average PSNR we will expand the datasets by iterating over them for 10 rounds, with each round having an independently drawn noisy measurement.

Figure 6.4: Example denoising results of our network trained on ImageNet. Gaussian noise where $\sigma = 25$

All images in the training and testing datasets were corrupted with Gaussian noise with a standard deviation of $\sigma = 25$. Further the network was trained for 40 epoch, at which point results diminished. Optimal parameters were found through cross validation on a portion of the imagenet validation dataset.

Table 6.1: Results of Denoising Over Different Architectures. HQSS and Noise2Noise Results Taken From [26].

Method	Unsupervised	Kodak	BSDS300	Average
Noise2Noise	No	32.45	31.07	31.76
HQSS	Yes	32.45	31.03	31.74
MAP-GAN	Yes	29.54	29.75	29.65
MAP-GAN Unpaired	No	29.55	29.78	29.67
MAP-GAN Paired	No	29.33	29.62	29.48

6.4 Ablation Studies

To further examine the underlying properties of MAP-GAN we also conducted two ablation studies. Notably in both studies the underlying network architecture and training schema stayed the same as the ImageNet experiments unless otherwise noted.

In the first study we provided the network with unpaired clean and dirty data in a semi-supervised senario. In other words the architecture has access to the set of clean data X and the set of noisy data Y, but not pairs (x, y). In this scenario the discriminator will decide between the generated output (not renoised) and the clean data distribution.

In the second study the network has access to clean and dirty data pairs, (x, y). The modifications to the discriminator's input from the first study remain, it will decide between the generated output (not renoised) and the clean data distribution. In

addition the generator has an increased mean squared error loss of the clean data x upon the the generated output $G(y)$.

The quantitative results of these studies can be found in Table 6.1.

Chapter 7

RESULTS

In this section we will discuss the results and try to identify the strengths and weakness that can be extrapolated from the previous experiments.

7.1 Compared to Previous State of the Art

From graph 6.1 and the output figures we can see that our method, while able to quantitatively and qualitatively reduce the noise of the images a significant amount, was thus far unsuccessful in achieving state of the art results with our architecture. We notice that generally speaking we fell behind by approximately 2 DB. Notably this gap is not so insurmountable as to draw a conclusion about the future potential of the MAP-GAN system, but it does show that our current best effort network architecture is likely not capable of surpassing the state of the art with any hyper parameter combination.

While looking at the test image results like those provided in the appendix we can see significant improvement in overall texture noise, while seemingly maintaining a strong level of fine detail. We note that images composed of textures that have relatively high levels of contrast in smaller areas, like choppy water or blades of grass such as that seen in Figure 7.1, tend to have lower scores than that of uniform textures. We hypothesise that the MAP-GAN model had difficulties learning the likelihood in these scenarios and instead optimized for an average texture, that while looked reasonable on a macro scale, caused a relatively poor per pixel score.

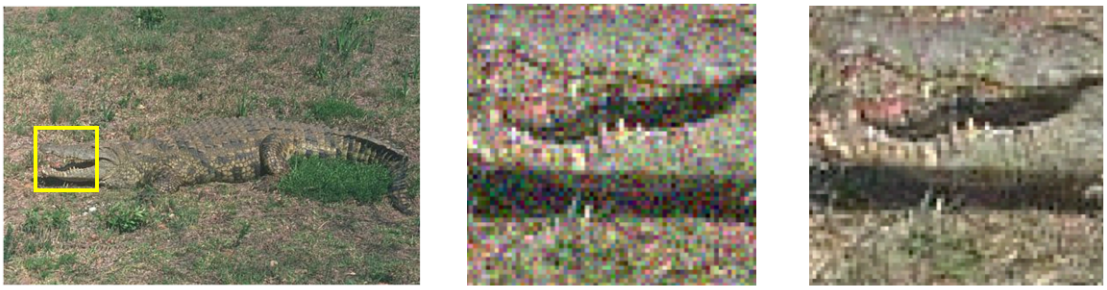


Figure 7.1: Example of poor MAP-GAN denoising over grassy image

7.2 Abolition Results

In order to test the effectiveness and the capacity of our network to learn in an unsupervised setting as compared to a semi-supervised and fully supervised setting we performed two abolition studies as presented in previous experiments. When examining the semi-supervised MAP-GAN's numerical results we can see that it achieved superior performance to the unsupervised version, but just barely. Interestingly the fully supervised version of our network failed to achieve strong results when compared to the semi-supervised and unsupervised versions, falling behind but only marginally so.

There are several possible conclusions that can be drawn from these results. The first is that the unsupervised MAP-GAN network has achieved full learning capacity; despite not having access to clean data the network was able to successfully learn the underlying clean distribution just as effectively as a semi-supervised or fully supervised GAN. This would tend to suggest that with the network, learning system, and architecture given, the best capable loss has been achieved.

Another conclusion that could be drawn is that somewhere within our current setup exists a flaw that impinges our network from learning an even more optimal clean

distribution. This fault could lie in the MAP-GAN system itself, the GAN, a poor network design choice, or incorrect hyperparameters. We know this because of the superior PSNR numbers produced under identical conditions in [26] show that there is further information that can be extracted from the noisy dataset. In order to attempt to identify the weakness of the system further experimentation is needed, primarily different network architectures under the MAP-GAN scheme.

It is worth noting that the paired training variant scored slightly lower than the other variants which is an unexpected result. Hyperparameters, specifically the learning weight, was adjusted to attempt to find the best outcome but consistently scored below the regular MAP-GAN and the unpaired MAP-GAN variants.

7.3 CelebA and MNIST

The purpose of the CelebA and MNIST tests were to see if the MAP-GAN architecture was feasible, study how it responded to increasingly complex images, and develop competent networks. Qualitative metrics provided the most utility during this stage in development, while quantitative metrics could be seen on a per-image basis. However without another network to compare with the PSNR scores for these tests provided only moderate utility. The CelebA results were overall visually satisfactory, particularly in managing to reconstruct fine texture like hair. Many human features are already soft, without hard lines and so lend themselves to be reconstructed well.

MNIST results on the other hand left left something to be desired, notably in the elimination of noise form the background of the images. While recognizing that both tests shown have a significantly higher noise standard deviation, $\sigma = 32, 64$ while other tests had $\sigma = 25$, we had expected that the MAP-GAN schema would be able to eliminate the noise from a solid background better than the visual results show. We

hypothesize the reasoning for this is 2 fold; the first reason being that the network used is too powerful for the given training images. The network used in these experiments was developed and targeted at images of size 256x256 with 3 color channels and 50,000 training examples. The size of the MNIST images are 32x32 single channel with an equivalent amount of training examples, so it is possible that the dataset does not have the image size or dataset size to fully train the network. Second, the MNIST photos represent large swaths of extreme black and white contrast, with the background having 0 lumen, the numbers having full 255 lumen, and very little in between. While in other image sets we have seen the network perform well on smooth areas of low contrast, it is possible that the extreme variance of this dataset is causing issues in estimating the likelihood.

Chapter 8

CONCLUSION

Our initial goal was to create a novel architecture that could compete or beat the state of the art unsupervised denoising scheme. Unfortunatley we were unable to create a network that was able to beat the current state of the art architecture, but we were able to conclusively show the potential of the MAP-GAN schema and produce respectable results.

We implemented our architecture utilizing a deep neural net validating the MAP-GAN schema of using a known noisy prior in a generative adversarial network for unsupervised denoising. We then trained our model over 3 popular datasets of progressing difficulty and variation, and extrapolated quantitative and qualitative data therefore confirming that our schema is capable of practical and effective learning. We also trained and tested our model in such a way that we could fairly compare our results to the state of the art architecture of Laine et al. from both a quantitative and qualitative perspective. Lastly we performed an abolition study to see the effects of our network under semi-supervised and fully supervised scenarios. The conclusion drawn is that our network scheme did not have the ability to supersede the state of the art network, and our abolition study showed that our unsupervised method was able to achieve very close to the results of a semi supervised and fully supervised training implying that it successfully recovered the clean distribution to the best of the network's ability.

While our MAP-GAN model may not have achieved state of the art performance compared to (HQSS) impressive results, we do not necessarily believe that this is

an inherit issue within the MAP-GAN scheme itself. Rather it is possible that an incorrect deep neural network architecture was used or a more targeted learning rate schedule could have been found.

8.1 Future Work

Within the direct scope of this project we can see multiple avenues for future work and more in depth research. The first opportunity is to continue to modify the deep network architecture used within our model along with different testing and training methods and separate learning rate schedulers. We feel that there is room for improvement within this area alone. Another excellent opportunity for future work is to check this model out on different noise generating processes to identify its strengths and limitations. Since our model requires that the noise generating process be expressed analytically, this may limit the types of noise methods available, but we believe that it is worth investigating available avenues such as salt and pepper noise.

BIBLIOGRAPHY

[1] Cal Poly Github. `http://www.github.com/CalPoly`.

[2] S. K. B. K. Image denoising based on non-local means filter and its method noise thresholding. *Signal, Image and Video Processing*, 7:1211–1227, 11 2013.

[3] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[4] C. Boncelet. Chapter 7 - image noise models. In A. Bovik, editor, *The Essential Guide to Image Processing*, pages 143–167. Academic Press, Boston, 2009.

[5] A. Bora, E. Price, and A. G. Dimakis. Ambientgan: Generative models from lossy measurements. In *International Conference on Learning Representations*, 2018.

[6] S. Cha and T. Moon. Fully convolutional pixel adaptive image denoiser. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 4159–4168. IEEE, 2019.

[7] J. Chen, J. Chen, H. Chao, and M. Yang. Image blind denoising with generative adversarial network based noise modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3155–3164, 2018.

[8] D. Cho. *Image denoising using wavelet transforms*. PhD thesis, Concordia University, 2004.

[9] E. Cohen, R. Heiman, M. Carmi, O. Hadar, and A. Cohen. When physics meets signal processing: Image and video denoising based on ising theory. *Signal Processing: Image Communication*, 34:14–21, 2015.

[10] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

[11] D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995.

[12] R. W. Franzen.

[13] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[14] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[16] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016. Recent Developments on Deep Big Vision.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[18] T. Huang, G. Yang, and G. Tang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(1):13–18, 1979.

[19] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[20] A. Jabbar, X. Li, and B. Omar. A survey on generative adversarial networks: Variants, applications, and training. *ACM Comput. Surv.*, 54(8), Oct. 2021.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[23] P. Korshunov and S. Marcel. Vulnerability assessment and detection of deepfake videos. In *2019 International Conference on Biometrics (ICB)*, pages 1–6, 2019.

[24] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[26] S. Laine, T. Karras, J. Lehtinen, and T. Aila. High-quality self-supervised deep image denoising. *Advances in Neural Information Processing Systems*, 32:6970–6980, 2019.

[27] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[28] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[29] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila. Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189*, 2018.

[30] J. Liang and R. Liu. Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network. In *2015 8th International Congress on Image and Signal Processing (CISP)*, pages 697–701, 2015.

[31] B. Liu and J. Liu. Overview of image denoising based on deep learning. *Journal of Physics: Conference Series*, 1176:022010, mar 2019.

[32] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[33] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[34] P. Milanfar. A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE Signal Processing Magazine*, 30(1):106–128, 2013.

[35] M. Motwani, M. Gadiya, R. Motwani, and F. Harris. Survey of image denoising techniques. 01 2004.

[36] A. Pajot, E. de Bezenac, and P. Gallinari. Unsupervised adversarial image reconstruction. In *International Conference on Learning Representations*, 2018.

[37] P. Patidar, M. Gupta, S. Srivastava, and A. K. Nagawat. Image de-noising by various filters for different noise. *International journal of computer applications*, 9(4):45–50, 2010.

[38] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Transactions on Image processing*, 12(11):1338–1351, 2003.

[39] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[40] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[41] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[42] C. Saxena and D. Kourav. Noises and image denoising techniques: A brief survey. *International journal of Emerging Technology and advanced Engineering*, 4(3):878–885, 2014.

[43] L. Sendur and I. W. Selesnick. Bivariate shrinkage functions for wavelet-based denoising exploiting interscale dependency. *IEEE Transactions on signal processing*, 50(11):2744–2756, 2002.

[44] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin. Deep learning on image denoising: An overview. *Neural Networks*, 2020.

[45] C. Tian, Y. Xu, L. Fei, and K. Yan. Deep learning for image denoising: a survey. In *International Conference on Genetic and Evolutionary Computing*, pages 563–572. Springer, 2018.

[46] R. Verma and J. Ali. A comparative study of various types of image noise and efficient noise removal techniques. *International Journal of advanced research in computer science and software engineering*, 3(10), 2013.

[47] X. Wang, S. Shen, G. Shi, Y. Xu, and P. Zhang. Iterative non-local means filter for salt and pepper noise removal. *Journal of Visual Communication and Image Representation*, 38:440–450, 2016.

[48] Z. Wang and A. C. Bovik. A universal image quality index. *IEEE signal processing letters*, 9(3):81–84, 2002.

[49] J. Xu, Y. Huang, M.-M. Cheng, L. Liu, F. Zhu, Z. Xu, and L. Shao. Noisy-as-clean: learning self-supervised denoising from corrupted image. *IEEE Transactions on Image Processing*, 29:9316–9329, 2020.

[50] Q. Yang, P. Yan, Y. Zhang, H. Yu, Y. Shi, X. Mou, M. K. Kalra, Y. Zhang, L. Sun, and G. Wang. Low-dose ct image denoising using a generative

adversarial network with wasserstein distance and perceptual loss. *IEEE transactions on medical imaging*, 37(6):1348–1357, 2018.

[51] L. Yuan, J. Sun, L. Quan, and H.-Y. Shum. Image deblurring with blurred/noisy image pairs. *ACM Trans. Graph.*, 26(3):1–es, July 2007.

[52] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

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

[54] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual dense network for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(7):2480–2495, 2021.

APPENDICES

Appendix A

IMPLEMENTATION

The layout specified below is specifically for the ImageNet implementation. The networks used in the other tests were similar, but had slight modifications to the architecture and the hyperparameters.

A.1 Training Inspirations

We took inspiration from the work of [26] for the network architecture and training style, seeing as how they currently hold the bar to beat.

We developed in a Python3 environment and utilized Tensorflow for their excellent neural network library. We initialized our weights using the He Normal initializer [17]. We used the Adam learning rate optimizer [21] with a learning rate of alpha $= 0.0004$ and beta $= 0.5$ for both the generator and the discriminator, along with a cosine decay of the learning rate that started at the half way training mark. The network was trained for 40 epoch.

While training over the image net validation dataset we resized the images color axis values to the range of $[0, 1]$ and used a batch size of 4. The training set was filtered to only contain images between 256x256 and 512x512, and training batches were composed of random 256x256 crops from the images to stay consistent with [26]. Test images were used as is, no padding was applied. All test image sets were

replicated 10 times, with each replication having an independent noisy measurement taken, to ensure a consistent average PSNR.

A.1.1 Network

The generator is roughly inspired by [26] network, which is roughly a u-net style with skip connections [39]. We used a bottleneck style fully convolutional network, with most convolutions having a size of 3x3, and utilized a stride of two in order to downsize the network. Similarly to first half of the generator network, in the second half we used deconvolutional operators of size 3x3 with a stride of 2. At the end of the generator we had several convolutions of size 1x1 with stride 1.

The discriminator that gave the best experimental results was a variant of the patch-GAN discriminator as seen in the paper "Image-to-image translation with conditional adversarial networks" [19].

Table A.1: Generator Network

Name	N_{out}	Function	Stride
Input	3		
Conv0	48	Convolution 3x3	1
Conv1	48	Convolution 3x3	2
Conv2	48	Convolution 3x3	2
Conv3	48	Convolution 3x3	2
Conv4	48	Convolution 3x3	2
Conv5	48	Convolution 3x3	2
Conv6	48	Convolution 3x3	1
Deconv0	96	Transposed Convolution 3x3	2
Concat	144	Concatenate output of Conv1	
Deconv0b	96	Convolution 3x3	1
Deconv1	96	Transposed Convolution 3x3	2
Concat	144	Concatenate output of Conv2	
Deconv1b	96	Convolution 3x3	1
Deconv2	96	Transposed Convolution 3x3	2
Concat	144	Concatenate output of Conv3	
Deconv2b	96	Convolution 3x3	1
Deconv3	96	Transposed Convolution 3x3	2
Concat	144	Concatenate output of Conv4	
Deconv3b	96	Convolution 3x3	1
Deconv4	96	Transposed Convolution 3x3	2
Concat	99	Concatenate Input	
Deconv4b	96	Convolution 3x3	1
Conv7	96	Convolution 1x1	1
Conv8	96	Convolution 1x1	1
Output	3	Convolution 1x1	1

Table A.2: Discriminator Network

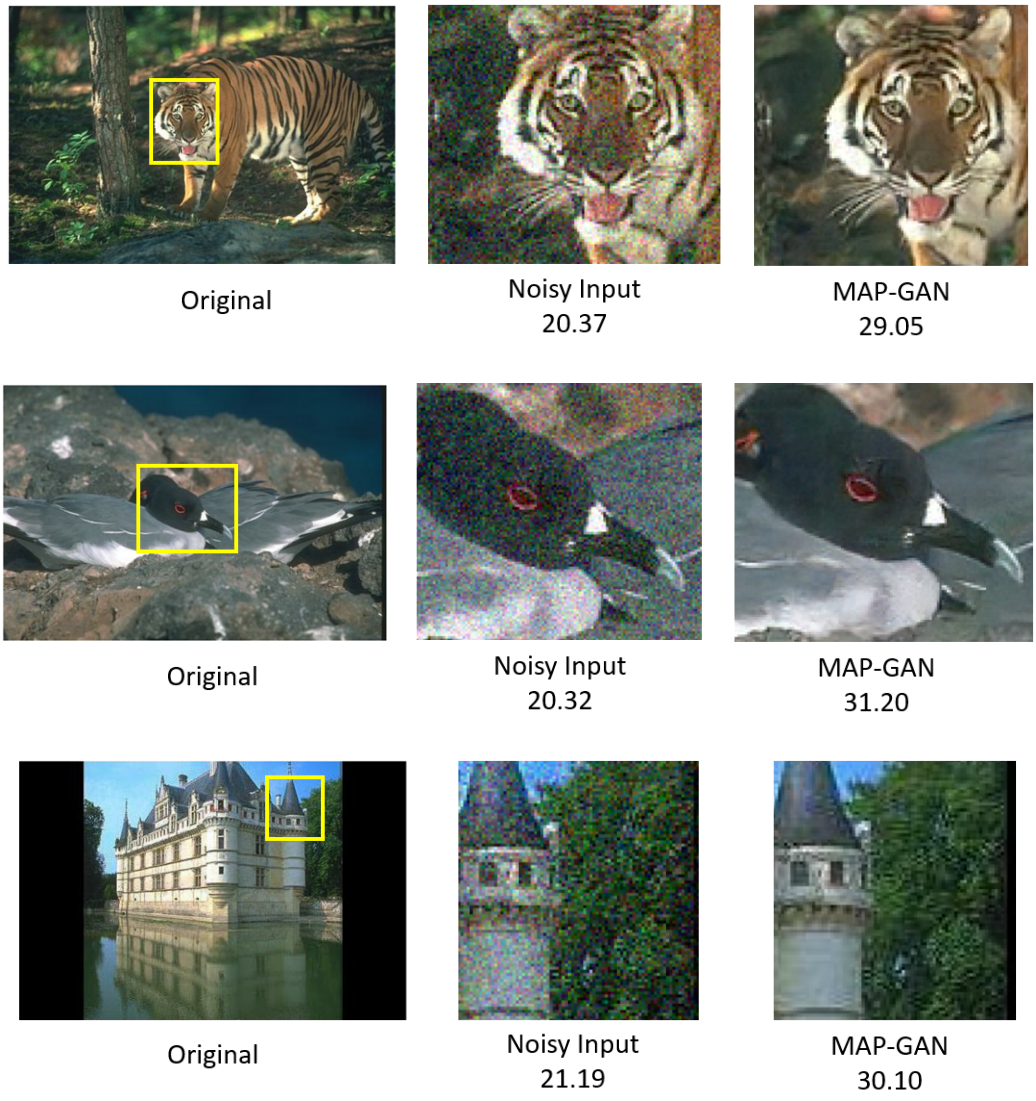
Name	N_{out}	Function	Stride
Input	3		
Conv0	96	Convolution 3x3	1
Conv1	96	Convolution 3x3	2
Conv2	96	Convolution 3x3	1
Conv3	96	Convolution 3x3	2
Conv4	96	Convolution 3x3	1
Conv5	96	Convolution 3x3	2
Conv6	96	Convolution 3x3	1
Conv7	48	Convolution 3x3	1
Conv8	24	Convolution 3x3	1
Conv9	1	Convolution 3x3	1

Appendix B

APPENDIX

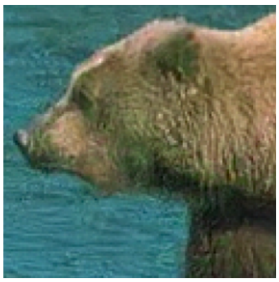
B.1 Additional Photos

Here we present additional photos from the testing dataset (Kodak and BSDS300) denoised by the MAP-GAN model trained on the ImageNet Validation dataset.

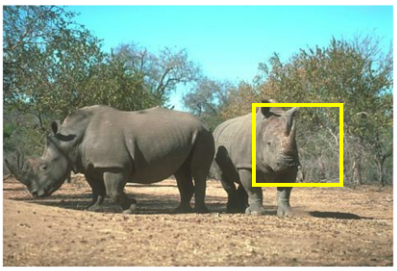


Original

Noisy Input
20.29

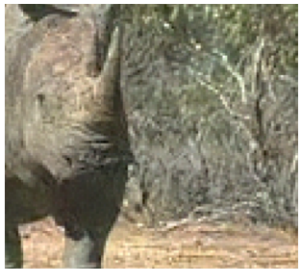


MAP-GAN
29.03

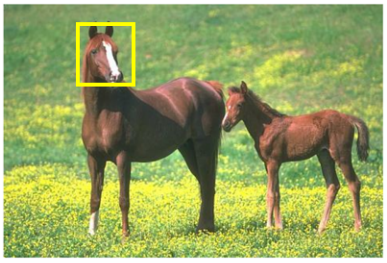


Original

Noisy Input
20.47

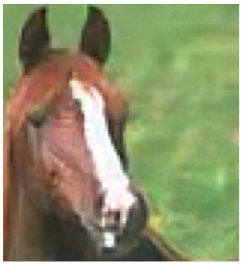


MAP-GAN
26.98



Original

Noisy Input
20.32



MAP-GAN
27.34

Original

Noisy Input
20.47

MAP-GAN
28.25

Original

Noisy Input
20.43

MAP-GAN
28.72