Deep Neural Network Approximation for Image Denoising

Bachelor's Thesis Project: Final Report
Bhumika Chopra, Silky Singh
Supervised by: Prof. Sivananthan Sampath

1 Abstract

Presence of noise in images can adversely affect tasks that require subsequent processing of images, for example, medical diagnosis. It is indeed very easy for an image to accumulate noise when being passed over a transmission channel, during compression or acquisition.

Image denoising is the process of removing noise from a noisy image so as to restore the original image. It is not an easy task to distinguish between the relevant and irrelevant high frequency components- noise, edges and texture. Eventually, it leads to loss of information.

Multiple noise removal techniques have been proposed in the past(ref. [1]) based on transformations, domain filtering and thresholding. In the recent times when deep neural networks and GAN-based models are producing state-of-the-art in many image processing tasks like image segmentation, image classification, image inpainting etc., the image denoising task has also seen better results from this shift.

2 Introduction

We start off by explaining deep neural networks and their property of being universal approximators. Next, we have shown a basic CNN architecture and the proof of Johnson-Lindenstrauss dimensionality reduction lemma which is essential for understanding kernel transformations in CNNs. After that we have given a brief introduction to GANs and generative modeling. We have also provided illustrations for both CNNs and GANs for the task of image denoising.

In sections 9 and 10, we briefly discuss the types of noise present in images and classical methods of image denoising. In section 11, we discuss and give proofs for the convergence of Stochastic Gradient Method applied on convex and strongly-convex non-smooth functions in a convex domain. In section 12, we discuss the convergence of GANs. Finally in section 13, we discuss an application of image denoising framework for the task of unmasking masked human face images.

3 What are deep neural networks?

Deep Neural Networks are interconnected group of neurons stacked together in layers to learn a mapping from the input space to the output space. It is characterised by an input layer, followed by hidden layer(s) and finally the output layer.
Neural networks have found applications in healthcare, autonomous driving, fraud detection, defence etc.

4 (Convolutional) Neural Networks are universal approximators

The universal approximation properties of Deep Neural networks have been widely studied. It has been shown in the literature([2]) that the neural networks can be used to approximate any continuous function to an arbitrary accuracy when the depth of the neural network is large enough. We wish to study these properties in detail in the future.

5 Dimensionality reduction in CNNs

Convolutional neural networks reduce the dimension of the input images to better learn the discriminative features of the input space. By reducing the dimensions, they are better able to capture the representative properties of the input space. Generally, a set of points in a higher dimensional space lie in a much lower dimensional manifold. By inherently reducing the dimension of the inputs, DNNs/CNNs also potentially reduce the compute resources that are needed.
The Johnson-Lindenstrauss lemma shows that there always exists a mapping from higher dimensional space to a lower dimensional space.

5.1 Proof of Johnson-Lindenstrauss Lemma [3]

Goal: Project a set of points $\{x_1, x_2, ..., x_n\}$ to a lower dimensional space such that the relative distances between the points are preserved.

Near Orthogonality: There could be as many as $e^{c\epsilon^2 d}$ unit vectors in R^d which are ϵ-orthogonal, i.e., whose mutual inner products all lie in $[-\epsilon, \epsilon]$.

Lemma: Let $\epsilon \in (0, 1)$ and $k \geq \frac{24}{3\epsilon^2 - 2\epsilon^3} \log n$. Then for any set of points $S = \{x_1, x_2, ..., x_n\}$ in R^d, $\exists$ a mapping

$$A : R^d \rightarrow R^k$$

such that
$\forall x_1, x_2 \in S$:

$$(1-\epsilon)\cdot ||x_1 - x_2||^2 \le ||Ax_1 - Ax_2||^2 \le (1+\epsilon)\cdot ||x_1 - x_2||^2$$

Construction of the map: Let M be a $k \times d$ matrix. The entries of M are drawn from a standard normal distribution $N(0,1)$ independently, hence they are i.i.d.

Then the map A is defined as: $A(x) = \frac{1}{\sqrt{k}}Mx$

Some properties of Gaussian distribution:
The density function of a normal distribution is given by:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\Big(-\frac{(x-\mu)^2}{2\sigma^2}\Big)$$

If $G_i \sim N(\mu_i, \sigma_i^2)$,

$$cG_i \sim N(c\mu_i, c^2\sigma_i^2)$$
$$G_1 + G_2 \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

Let $Y = Mx$. Each element of the vector Y can be written as:

$$Y_i = \sum_{j=1}^{d} M_{ij}x_j$$

by the properties above, $M_{ij}x_j \sim N(0, x_j^2)$ and therefore:

$$Y_i \sim N\Big(0, \sum_{j=1}^{d} x_j^2\Big), \quad i = 1, 2, ...k$$

for $x \in R^d$, consider Z:

$$Z = (Ax)^2 = \Big(\frac{1}{\sqrt{k}}Mx\Big)^2 = \frac{1}{k}Y^2 = \frac{1}{k}\sum_{j=1}^{k} Y_j^2$$

Note that, $\sum_{j=1}^{k} Y_j^2$ has a chi-square distribution(by definition) with mean k. Thus, $E[Z] = \frac{1}{k}\cdot k = 1$

We will prove for one tail, and a similar approach follows for the other tail.
Markov's inequality:

$$Pr[|X| \ge c] \le \frac{E[|X|]}{c}$$

$$\begin{aligned} Pr[Z \geq 1+\epsilon] &= Pr[e^{tkZ} \geq e^{tk(1+\epsilon)}] \\ &\leq \frac{E[e^{tkZ}]}{e^{tk(1+\epsilon)}} \\ &= \prod_{i=1}^{k} \frac{E[e^{tY_i^2}]}{e^{t(1+\epsilon)}} \end{aligned} \tag{1}$$

Now,

$$E[e^{tY^2}] = \int_{-\infty}^{\infty} e^{ty^2} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy = \frac{1}{\sqrt{1-2t}}, \quad 0 < t < \frac{1}{2}$$

Thus,

$$\begin{aligned} Pr[Z \geq 1+\epsilon] &\leq \Big(\frac{1}{\sqrt{1-2t}}\Big)^k \frac{1}{e^{tk(1+\epsilon)}} \\ &\text{set t} = \frac{\epsilon}{2(1+\epsilon)} \\ &\leq \Big((1+\epsilon)\cdot e^{-\epsilon}\Big)^{k/2} \\ &\text{using } log(1+\epsilon) < \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3}, \text{we get:} \\ &\leq \Big(e^{-\frac{\epsilon^2}{2}+\frac{\epsilon^3}{3}}\Big)^{k/2} \\ &\text{using } \ k \geq \frac{24}{3\epsilon^2 - 2\epsilon^3} \log n \\ Pr[Z \geq 1+\epsilon] &\leq \frac{1}{n^2} \end{aligned} \tag{2}$$

Similarly,

$$Pr[Z \leq 1-\epsilon] \leq \frac{1}{n^2}$$

Using the Bonferroni inequality, taking the disjoint set S consisting of all pairing of points(n points result in $\frac{n(n-1)}{2}$ pairings), the union bound for the probability of any pair of points falling out of the desired error is:

$$\begin{aligned} P(\cup E_i) &\leq \sum_{i=1}^{n} P(E_i) \\ &\leq \frac{n(n-1)}{2} \cdot \frac{2}{n^2} \\ &\leq 1 - \frac{1}{n} \end{aligned} \tag{3}$$

Thus, the probability that all distances between the points in S are preserved is: $\frac{1}{n}$, which proves the existence of such A.

6 CNN- Illustration

Recently, CNNs have been employed to the task of image denoising. One common architecure that is used is an auto-encoder network, which consists of an encoder and a decoder. The encoder transforms the input to a latent code, usually much smaller in dimension. The decoder part of the network is the mirror image of the encoder network and it undoes all the operations on the latent code to finally produce an output of the same size as the input. Auto-encoders are heavily used in image-to-image translation tasks, and are a natural choice for image denoising as well.

Refer to Figure 1 for a simple auto-encoder architecture.

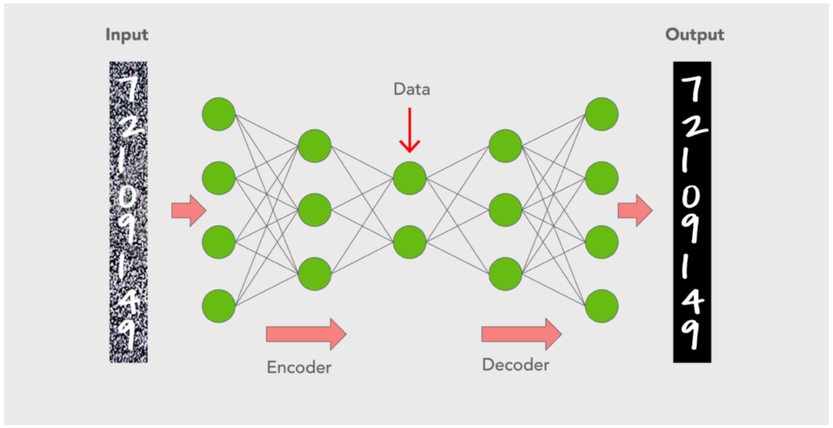


Figure 1: An auto-encoder architecture for image denoising

To illustrate the effectiveness of auto-encoders for image denoising, we built a simple PyTorch model consisting of an encoder and a decoder. The encoder takes in the noisy image and converts it to a latent code. The decoder part of the network takes in the latent code, and produces a denoised image. All the weights and biases in the network are learned during the training process.

For training, we produced pairs of noisy and denoised images from the MNIST dataset. [1]

Once the model is trained, we tested it on a sample of 10 images from the test set. To each of the test image, we added a Gaussian noise and then passed it as an input to our pre-trained model. We show the results for 10 test images in Figure 2:

[1]MNIST is a collection of 70,000 28x28 handwritten digits(60,000 for training, and 10,000 for testing), mainly used in image processing tasks.

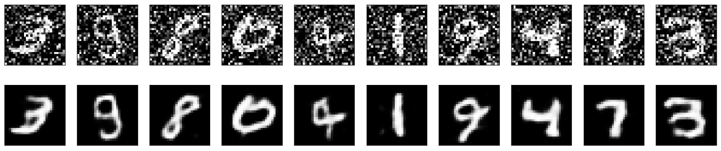


Figure 2: Test results for image denoising on the MNIST dataset
(Top row: noisy images, Bottom row: denoised images)

7 GANs

7.1 Introduction

Generative Adversarial Networks, or GANs for short, are an approach to generative modeling using deep learning methods, such as convolutional neural networks. GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real (from the domain) or fake (generated). The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

- Generator - Model that is used to generate new plausible examples from the problem domain.
- Discriminator - Model that is used to classify examples as real (from the domain) or fake (generated).

7.2 Generator

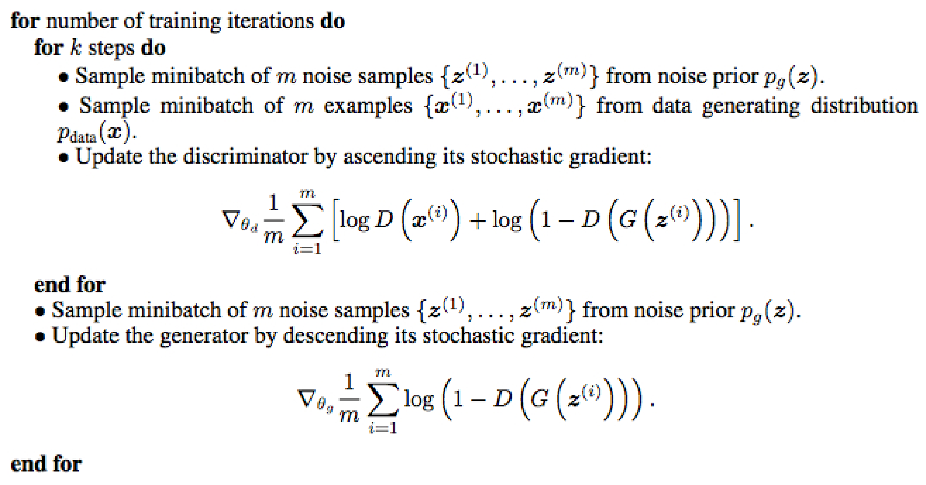
The generator model takes a fixed-length random vector as input and generates a sample in the domain. The vector is drawn randomly from a Gaussian distribution, and is used to seed the generative process. After training, points in this multidimensional vector space will correspond to points in the problem domain, forming a representation of the data distribution.

7.3 Discriminator

The discriminator model takes an example from the domain as input (real or generated) and predicts a binary class label of real or fake (generated). The real example comes from the training dataset. The generated examples are output by the generator model. The discriminator is a normal classification model.

7.4 Training GANs

Although generative modeling is an unsupervised learning problem, the GAN architecture is trained as if we are training a supervised learning model. This can also be formulated as a zero-sum game where the generator wins if it successfully fools the discriminator on all test inputs i.e. the discriminator is unable to distinguish between the real and generated images. The discriminator wins in the other case (when training we have to ensure that the adversary is a constant opponent)



```
for number of training iterations do
    for $k$ steps do
        • Sample minibatch of $m$ noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
        • Sample minibatch of $m$ examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
        • Update the discriminator by ascending its stochastic gradient:
            $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right].$
    end for
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Update the generator by descending its stochastic gradient:
        $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$
end for
```

Figure 3: Training

7.4.1 Minimax loss function

$$\min_G \max_D V(D, G)$$

$$V(D, G) = E_{x \sim p_{data(x)}}[\log(D(x))] + E_{z \sim p_z(z)}[\log(1 - D(G(z))]$$

7.4.2 Wasserstein Loss

The discriminator acts as a critic rather than a classifier.

$$criticloss = E_{x \sim p_{data(x)}}[D(x)] - E_{z \sim p_z(z)}[D(G(z)]$$

$$V(D, G) = E_{z \sim p_z(z)}[D(G(z)]$$

7.4.3 Difficulties

- Stability and convergence
- Early-stopping on seeing erratic loss functions

- Learning rate and hyperparameter selection
- Spectral Normalization
- Mode Collapse

8 GAN- Illustration

GANs are being used in all areas of machine learning particularly those that benefit form generative modeling. We implemented a GAN where the generator takes noisy images as input, with the aim of constructing as clear an image as possible. The discriminator model tries to effectively distinguish between the true and the generated image. The generator network model is based on U-Net while the discriminator is a simple CNN model (pre-trained on the dataset). We use cross entropy loss for training and follow the same steps in the above stated algorithm in figure 3.

The model is trained on the MNIST digits dataset and we have applied Gaussian random noise to generate the noisy images. After training we tested the model on 8 test images as shown in the figures 4-5.

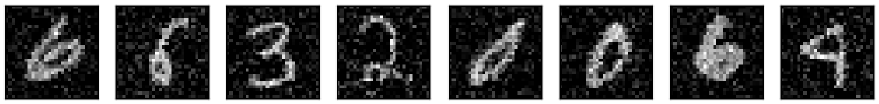


Figure 4: Testing set (noisy images)

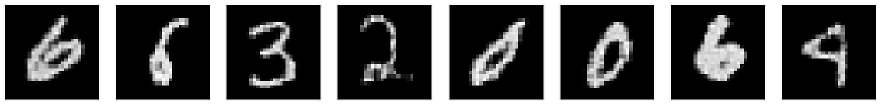


Figure 5: Denoised images

9 Noise Models

Noise is generally present in 2 forms- additive or multiplicative.

- Additive $\rightarrow w(x,y) = s(x,y) + n(x,y)$
- Multiplicative $\rightarrow w(x,y) = s(x,y) * n(x,y)$

Here, (x,y) is the pixel location (can be in 3D or 2D) in the image, $n(x,y)$ is the noise added and $s(x,y)$ is the original image intensity.

10 Classical methods

As given in [1] there are 3 basic approaches-

- **Spatial domain filtering** - mostly used to remove additive noise. Make use of low pass filtering to remove noise by assuming that noise lies in a higher frequency spectrum. E.g. average filter, median filter,
- **Transform domain filtering** - Involves transforming an image from one domain to another (generally lower dimensional) domain and applying spatial filtering methods. E.g. BM3D
- **Wavelet thresholding method** - Involves applying the wavelet transform to a signal and then computing a threshold to remove coefficients below it as they correspond to noise. E.g. hard thresholding, soft thresholding.

11 Convergence of Stochastic Gradient Descent

11.1 Subgradient Descent Algorithm

It is an algorithm for minimizing a non-differentiable convex function. A subgradient of a function $f : R^d \rightarrow R$ in $x \in R^d$ is a vector $g \in R^d$ that satisfies:

$$f(y) \geq f(x) + \langle g, y - x \rangle \quad \forall \quad y \in R^d$$

Intuitively, a subgradient of f in x is any vector g that allows us to construct a linear lower bound to f.

The set of subgradients of f is denoted by $\partial f(x)$.

The updates in the SD algorithm is defined as follows:

$$x_{t+1} = \prod_V \Big(x_t - \eta_t g_t \Big)$$

where g_t is the subgradient and $\prod_V$ denotes projection onto the feasible set V.

There are some differences between Subgradient Descent and Gradient Descent:

1. In SD, we do not assume that f is differentiable. We only have access to a subgradient g at any point x_0 in V, i.e., $g \in \partial f(x_0)$. Example- the hinge loss, and the ReLU activation functions are non-differentiable.

2. SD is not a descent method. It is possible that $f(x_{t+1}) > f(x_t)$

If SD is not a descent algorithm, then how does it work?

SD algorithm minimizes a local approximation of the original objective function. The subgradients give us the linear lower bound to the function f around x_0. However, the minimum of a linear function in unbounded domain can be $-\infty$. If we constrain the minimization of this lower bound in the neighbourhood of x_t, we will get the following formulation:

$$x_{t+1} = \text{ argmin }_{x \in V} f(x_t) + \langle g_t, x - x_t \rangle$$
$$s.t. \;\; ||x_t - x||^2 \leq h$$

This is also equivalent to the following unconstrained formulation, for some $\eta > 0$:

$$x_{t+1} = \text{ argmin }_{x \in V} f(x_t) + \langle g_t, x - x_t \rangle + \frac{1}{2\eta} ||x_t - x||_2^2$$

Simplifying the RHS:

$$x_{t+1} = \text{argmin}_{x \in V} \langle g_t, x - x_t \rangle + \frac{1}{2\eta_t} ||x_t - x||_2^2$$

$$x_{t+1} = \text{argmin}_{x \in V} 2\eta_t \langle g_t, x - x_t \rangle + ||x_t - x||_2^2$$

Since $||\eta_t g_t||^2$ is independent of x, we can add this term on the RHS:

$$x_{t+1} = \text{argmin}_{x \in V} ||\eta_t g_t||^2 - 2\eta_t \langle g_t, x_t - x \rangle + ||x_t - x||_2^2$$

$$x_{t+1} = \text{argmin}_{x \in V} ||x_t - \eta_t g_t - x||^2$$

$$x_{t+1} = \prod_V \Big(x_t - \eta_t g_t \Big)$$

$\prod_V$ is the Euclidean projection onto V, i.e., $\prod_V(x) = \text{argmin}_{y \in V} ||x - y||_2$

NOTE:

1. A decreasing learning rate must be used with non-differentiable functions. With constant learning rate, we can't guarantee convergence to the optimum value.

2. It is common in the literature to use the term Stochastic Gradient Descent instead of Subgradient Descent.

11.2 PROOF OF CONVERGENCE

[4] have presented a proof of convergence of SGD algorithm applied on non-smooth convex and strongly-convex functions in their 2013 paper. We discuss the details below.
Let us assume that F is a convex function over a closed convex domain W, which is a subset of some Hilbert space with an induced norm $||.||$

Let

$$\arg\min_{w} F = w^* \quad \in W$$

Using the result above, we can define the iterates as follows:

$$w_{t+1} = \prod_{W}(w_t - \eta_t \hat{g}_t)$$

where $g_t \in \partial F(w_t)$ and $\mathbb{E}\hat{g}_t = g_t$

Theorem 1: Suppose F is λ- strongly convex and that $\mathbb{E}[||\hat{g}_t||^2] \le G^2 \; \forall \; t$. Consider SGD with step sizes $\eta_t = \frac{1}{\lambda t}$. Then for any $T > 1$, it holds that

$$\mathbb{E}[F(w_T) - F(w^*)] \le \frac{17G^2(1 + log(T))}{\lambda T}$$

Proof: By convexity of W, for any $w \in W$:

$$\mathbb{E}[||w_{t+1} - w||^2] = \mathbb{E}[||\prod_{W}(w_t - \eta_t \hat{g}_t) - w||^2]$$

$$\le \mathbb{E}[||w_t - \eta_t \hat{g}_t - w||^2]$$

$$\le \mathbb{E}[||w_t - w||^2] - 2\eta_t \mathbb{E}[\langle g_t, w_t - w\rangle] + \eta_t^2 G^2$$

Let k be an arbitrary element in $\{1, 2, ..., \lfloor T/2 \rfloor\}$. Rearranging such that the inner product is on the left hand side, and summing over all $t = T-k, ..., T$, we get

$$\begin{aligned}\sum_{t=T-k}^{T} \mathbb{E}[\langle g_t, w_t - w\rangle] &\le \sum_{t=T-k}^{T} \frac{\mathbb{E}[||w_t - w||^2] - \mathbb{E}[||w_{t+1} - w||^2] + \eta_t^2 G^2}{2\eta_t} \\ &\le \sum_{t=T-k}^{T} \frac{\mathbb{E}[||w_t - w||^2]}{2\eta_t} - \sum_{t=T-k+1}^{T} \frac{\mathbb{E}[||w_t - w||^2]}{2\eta_{t-1}} + \frac{G^2}{2} \cdot \Big(\sum_{t=T-k}^{T} \eta_t\Big) \\ &\le \frac{\mathbb{E}[||w_{T-k} - w||^2]}{2\eta_{T-k}} + \sum_{t=T-k+1}^{T} \frac{\mathbb{E}[||w_t - w||^2]}{2}\Big(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\Big) + \frac{G^2}{2} \cdot \Big(\sum_{t=T-k}^{T} \eta_t\Big)\end{aligned} \tag{4}$$

By convexity of F, we can lower bound $\langle g_t, w_t - w\rangle$ by $F(w_t) - F(w)$ on the left hand side of the inequality. Also substituting $\eta_t = \frac{1}{\lambda t}$, we get:

$$\mathbb{E}\Big[\sum_{t=T-k}^{T}\Big(F(w_t)-F(w)\Big)\Big] \leq \frac{\lambda(T-k)}{2}\mathbb{E}[||w_{T-k}-w||^2]+\frac{\lambda}{2}\sum_{t=T-k+1}^{T}\mathbb{E}[||w_t-w||^2]+\frac{G^2}{2\lambda}\sum_{t=T-k}^{T}\frac{1}{t}$$

From Rakhlin et. al., we know that $\mathbb{E}[||w_t - w^*||^2] \leq \frac{4G^2}{\lambda^2 t}$. Thus, for any $t \geq T-k$,

$$\begin{aligned}
\mathbb{E}[||w_t - w_{T-k}||^2] &\leq 2\mathbb{E}[||w_t - w^*||^2 + ||w_{T-k} - w^*||^2] \\
&\leq 2\Big(\frac{4G^2}{\lambda^2 t} + \frac{4G^2}{\lambda^2(T-k)}\Big) \quad \text{(using Rakhlin et. al. result)} \\
&\leq \frac{16G^2}{\lambda^2(T-k)} \leq \frac{32G^2}{\lambda^2 T}
\end{aligned} \tag{5}$$

Using the above result, and putting $w = w_{T-k}$, we obtain the following:

$$\mathbb{E}\Big[\sum_{t=T-k}^{T}\Big(F(w_t) - F(w_{T-k})\Big)\Big] \leq \frac{\lambda}{2}\Big(\frac{32G^2}{\lambda^2 T}\Big)\cdot k + \frac{G^2}{2\lambda}\sum_{t=T-k}^{T}\frac{1}{t}$$

Let S_k be defined as the expected average value of the last $k+1$ iterates,

$$S_k = \frac{1}{k+1}\sum_{t=T-k}^{T}\mathbb{E}[F(w_t)]$$

Using the definition of S_k, we can rewrite the above inequality as follows:

$$(k+1)S_k - \mathbb{E}[F(w_{T-k})]\cdot(k+1) \leq \frac{16G^2 k}{\lambda T} + \frac{G^2}{2\lambda}\sum_{t=T-k}^{T}\frac{1}{t}$$

$$-\mathbb{E}[F(w_{T-k})] \leq -\mathbb{E}[S_k] + \frac{16G^2 k}{\lambda T(k+1)} + \frac{G^2}{2\lambda(k+1)}\sum_{t=T-k}^{T}\frac{1}{t}$$

$$-\mathbb{E}[F(w_{T-k})] \leq -\mathbb{E}[S_k] + \frac{G^2}{2\lambda}\Big(\frac{32}{T} + \sum_{t=T-k}^{T}\frac{1}{(k+1)t}\Big)$$

Using the fact that $(k+1)S_k - kS_{k-1} = \mathbb{E}[F(w_{T-k})]$ and since S_k's are constants, we have $\mathbb{E}[S_k] = S_k$. Thus,

$$k\mathbb{E}[S_{k-1}] = (k+1)\mathbb{E}[S_k] - \mathbb{E}[F(w_{T-k})]$$

$$\begin{aligned}
k\mathbb{E}[S_{k-1}] &\leq (k+1)\mathbb{E}[S_k] - \mathbb{E}[S_k] + \frac{G^2}{2\lambda}\Big(\frac{32}{T} + \sum_{t=T-k}^{T}\frac{1}{(k+1)\cdot t}\Big) \\
\mathbb{E}[S_{k-1}] &\leq \mathbb{E}[S_k] + \frac{G^2}{2\lambda}\Big(\frac{32}{kT} + \sum_{t=T-k}^{T}\frac{1}{k(k+1)t}\Big)
\end{aligned} \tag{6}$$

Summing the above inequality from $k = 1$ to $k = \lfloor \frac{T}{2} \rfloor$, we obtain:

$$\mathbb{E}[F(w_T)] = \mathbb{E}[S_0] \leq \mathbb{E}[S_{\lfloor \frac{T}{2} \rfloor}] + \frac{16G^2}{\lambda T} \sum_{k=1}^{\lfloor \frac{T}{2} \rfloor} \frac{1}{k} + \frac{G^2}{2\lambda} \sum_{k=1}^{\lfloor \frac{T}{2} \rfloor} \sum_{t=T-k}^{T} \frac{1}{k(k+1)t}$$

The expected average of the last $\lfloor T/2 \rfloor$ iterates, $\mathbb{E}[S_{T/2}]$ was analyzed in Rakhlin et. al. (2011). For $T > 1$,

$$\mathbb{E}[S_{\lfloor T/2 \rfloor}] \leq F(w^*) + \frac{10G^2}{\lambda T}$$

Upper bounding the terms on the RHS of the inequality (6):

$$\sum_{k=1}^{\lfloor T/2 \rfloor} \frac{1}{k} \leq 1 + log(\frac{T}{2})$$

$$\begin{aligned} \sum_{k=1}^{\lfloor T/2 \rfloor} \sum_{t=T-k}^{T} \frac{1}{k(k+1)t} &\leq \sum_{k=1}^{\lfloor T/2 \rfloor} \frac{1}{k(T-k)} \\ &= \frac{1}{T} \sum_{k=1}^{\lfloor T/2 \rfloor} \Big(\frac{1}{k} + \frac{1}{T-k} \Big) \\ &\leq \frac{1 + log(T)}{T} \end{aligned} \tag{7}$$

Substituting the above bounds in the inequality (6), we get the following:

$$\begin{aligned} \mathbb{E}[F(w_T)] &\leq F(w^*) + \frac{10G^2}{\lambda T} + \frac{16G^2}{\lambda T}\Big(log(T/2) + 1\Big) + \frac{G^2}{2\lambda}\Big(\frac{1 + \log T}{T}\Big) \\ \mathbb{E}[F(w_T) - F(w^*)] &\leq \frac{17G^2(1 + log(T))}{\lambda T} \end{aligned} \tag{8}$$

Hence proved.

Theorem 2. Suppose that F is convex, and that for some constants D, G, it holds that $\mathbb{E}[\hat{g}_t] \leq G^2$ for all t, and $\sup_{w,w' \in W} ||w - w'|| \leq D$. Consider SGD with step sizes $\eta_t = \frac{c}{\sqrt{t}}$ for some constant $c > 0$. Then for any $T > 1$, it holds that

$$\mathbb{E}[F(w_T) - F(w^*)] \leq \Big(\frac{D^2}{c} + cG^2\Big)\frac{2 + log(T)}{\sqrt{T}}$$

PROOF:
Let k be an element in $\{1, 2, 3, ..., T-1\}$. The proof is similar to Theorem 1, where instead of $\eta_t = \frac{1}{\lambda t}$, we put $\eta_t = \frac{c}{\sqrt{t}}$, and sum over $t = T-k, ..., T$. Also, since we have assumed that $||w - w'|| \leq D \ \ \forall w, w'$, implies that $||w_t - w_{T-k}||^2 \leq D^2$, implies that $\mathbb{E}[||w_t - w_{T-k}||^2] \leq D^2$.

$$\mathbb{E}[\langle g_t, w_t - w_{T-k}\rangle] \le \frac{D^2}{2c}(\sqrt{T} - \sqrt{T-k}) + \frac{G^2}{2}\sum_{t=T-k}^{T}\frac{c}{\sqrt{t}}$$

From the laws of integration, we can obtain:

$$\sum_{t=T-k}^{T}\frac{1}{\sqrt{t}} \le 2(\sqrt{T} - \sqrt{T-k-1})$$

Also using the fact that F is convex, we obtain:

$$\begin{aligned}\mathbb{E}\Big[\sum_{t=T-k}^{T} F(w_t) - F(w_{T-k})\Big] &\le \Big(\frac{D^2}{2c} + cG^2\Big)(\sqrt{T} - \sqrt{T-k-1}) \\ &= \Big(\frac{D^2}{2c} + cG^2\Big)\frac{k+1}{\sqrt{T} + \sqrt{T-k-1}} \\ &\le \Big(\frac{D^2}{2c} + cG^2\Big)\frac{k+1}{\sqrt{T}}\end{aligned} \tag{9}$$

Now, consider S_k defined earlier. $S_k = \frac{1}{k+1}\sum_{t=T-k}^{T}\mathbb{E}[F(w_t)]$

$$\begin{aligned}(k+1)\cdot S_k - (k+1)\mathbb{E}[F(w_{T-k})] &\le \Big(\frac{D^2}{2c} + cG^2\Big)\cdot\frac{k+1}{\sqrt{T}} \\ -\mathbb{E}[F(w_{T-k})] &\le \Big(\frac{D^2}{2c} + cG^2\Big)\cdot\frac{k+1}{\sqrt{T}} - S_k\end{aligned} \tag{10}$$

Plugging the above inequality in $k\cdot\mathbb{E}[S_{k-1}] = (k+1)\mathbb{E}[S_k] - \mathbb{E}[F(w_{T-k})]$, we get:

$$\begin{aligned}k\cdot\mathbb{E}[S_{k-1}] &\le (k+1)\mathbb{E}[S_k] + \Big(\frac{D^2}{2c} + cG^2\Big)\cdot\frac{1}{\sqrt{T}} - \mathbb{E}[S_k] \\ &\le k\mathbb{E}[S_k] + \Big(\frac{D^2}{2c} + cG^2\Big)\cdot\frac{1}{\sqrt{T}} \\ \mathbb{E}[S_{k-1}] &\le \mathbb{E}[S_k] + \Big(\frac{D^2}{2c} + cG^2\Big)\cdot\frac{1}{k\sqrt{T}}\end{aligned} \tag{11}$$

Sum the above inequality over $k = 1, 2, ..., T-1$, we get:

$$\mathbb{E}[F(w_T)] = \mathbb{E}[S_0] \le \frac{1}{\sqrt{T}}\Big(\frac{D^2}{2c} + cG^2\Big)\sum_{k=1}^{T-1}\frac{1}{k} + \mathbb{E}[S_{T-1}] \tag{12}$$

And we know that $\sum_{k=1}^{T-1}\frac{1}{k} \le (1 + log(T))$. To further bound the terms on the RHS, we take Eq. (4) with $k = T-1$ and $w = w^*$

$$\mathbb{E}[S_{T-1}] - F(w^*) = \frac{1}{T}\mathbb{E}\Big[\sum_{t=1}^{T}\mathbb{E}[F(w_t) - F(w^*)]\Big]$$

$$\Rightarrow \mathbb{E}[S_{T-1}] - F(w^*) \leq \Big(\frac{D^2}{c} + cG^2\Big)\frac{1}{\sqrt{T}}$$

Plugging these bounds in Eq. (12), we get the following:

$$\begin{aligned}\mathbb{E}[F(w_T) - F(w^*)] &\leq \Big(\frac{D^2}{2c} + cG^2\Big)\frac{1}{\sqrt{T}}(1 + log(T)) + \Big(\frac{D^2}{c} + cG^2\Big)\cdot\frac{1}{\sqrt{T}} \\ &\leq \Big(\frac{D^2}{c} + cG^2\Big)\cdot\frac{2 + log(T)}{\sqrt{T}}\end{aligned} \tag{13}$$

Hence proved.

12 Convergence of GANs

Using the minimax loss function with cross-entropy loss [5] proved that, 'when given a model with infinite training capacity' the generator will be able to produce a good estimate of the actual data distribution. We define,

$$V(D,G) = E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_z(z)}[\log(1 - D(G(z)))]$$

Let the true data distribution is - p_{data} and the generator estimated distribution is - p_g.

Proposition 1 - Given the generator, G the optimal discriminator is given by-

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

Proof

Given a generator, G, the discriminator will try to maximize, V(D,G).

$$V(D,G) = \int_x p_{data}(x)\log(D(x))dx + \int_z p_z(z)\log(1 - D(G(z)))dz$$

$$V(D,G) = \int_x p_{data}(x)\log(D(x)) + p_g(x)\log(1 - D(x))dx$$

We have to find the point at which the function y $\rightarrow a\log(y) + b\log(1-y)$ attains its maximum value on the interval $[0,1]$, here $a, b \in R$. On differentiating and equating to 0 we get, $\frac{a}{y} - \frac{b}{1-y} = 0 \rightarrow y = \frac{a}{a+b}$. Hence, we get that $V(G,D)$ is maximum for $D^*_G(x)$ which is given by

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

We define a virtual training criterion -

$$C(G) = \max_D V(D, G)$$

Training the GAN can now be thought of as finding the generator, G, which minimizes

$$C(G) = E_{x\sim p_{data}}[\log(D^*_G(x))] + E_{x\sim p_g}[\log(1 - D^*_G(x))]$$

$$C(G) = E_{x\sim p_{data}}[\log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}] + E_{x\sim p_g}[\log \frac{p_g(x)}{p_{data}(x) + p_g(x)}]$$

Theorem 1 The global minimum of the virtual training criterion $C(G)$ is achieved only when $p_g = p_{data}$ and the minimum value is $-\log 4$.

Proof
At $p_g = p_{data}$ we get $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$. To see that this is the least possible value of $C(G)$, reached only for $p_g = p_{data}$, observe that

$$C(G) = -\log 4 + D_{KL}(p_{data}||\frac{p_{data} + p_g}{2}) + D_{KL}(p_g||\frac{p_{data} + p_g}{2})$$

Here, D_{KL} is the Kullback–Leibler divergence, it is a measure of how one probability distribution Q is different from another reference distribution P. We have the result,

$$D_{KL}(P||Q) = \sum_{x\sim X} P(x) \log \frac{P(x)}{Q(x)}$$

if P and Q are discrete distributions, or

$$D_{KL}(P||Q) = \int_{-\inf}^{inf} P(x) \log \frac{P(x)}{Q(x)} dx$$

if P and Q are continuous distributions. The Jensen-Shannon Divergence, given by,

$$JSD(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M)$$

here

$$M = \frac{1}{2}(P + Q)$$

is another method of measuring the similarity between 2 probability distributions. Substituting this in C(G), we get

$$C(G) = -\log 4 + 2JSD(p_{data}||p_g)$$

Proposition Jensen-Shannon divergence is always non-negative.
Proof

We note that $\ln a \leq a - 1\ \forall a \geq 0$. Now, we will prove that $D_{KL}(P||Q) \geq 0$. WLOG assume that the distributions P and Q are discrete.

$$-D_{KL}(P||Q) = -\sum_{x \sim X} P(x) \log \frac{P(x)}{Q(x)} = \sum_{x \sim X} P(x) \log \frac{Q(x)}{P(x)}$$

$$-D_{KL}(P||Q) \leq \sum_{x \sim X} P(x)(\frac{Q(x)}{P(x)} - 1) = \sum_{x \sim X} Q(x) - \sum_{x \sim X} P(x) = 1 - 1 = 0$$

$$D_{KL}(P||Q) \geq 0$$

Hence, $JSD(P||Q) \geq 0$.

If $JSD(P||Q) = 0$ then $D_{KL}(P||M) = 0$ where $M = \frac{P+Q}{2}$. This gives that $P = M$ which implies both distributions P and Q are equal.

Since the Jensen–Shannon divergence between two distributions is always non-negative and zero only when they are equal, we have shown that $C(G)$ =-$\log 4$ is the global minimum of C(G) and is attained only when is $p_g = p_{data}$, i.e., the generative model perfectly replicates the data generating process.

Theorem 2- Convergence of GANs. If G and D have enough capacity and at each step of the algorithm, D is allowed to reach its optimum and p_g is updated so as to minimize $C(G)$ then p_g converges to p_{data}.

$$C(G) = E_{x \sim p_{data}}[\log(D^*_G(x))] + E_{x \sim p_g}[\log(1 - D^*_G(x))]$$

Proof

The function $F(x) = a \log x + b \log(1 - x)$ where $a, b \in [0, 1]$ is convex. We get that, $C(G)$ is convex in p_g.

The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. This implies that if $f(x) = sup_{\alpha \in A} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x) \in \partial f$ if $\beta = argsup_{\alpha \in A} f_\alpha(x)$. This is equivalent to computing the gradient descent update for p_g at at the optimal D given the generator, G. We have proven that $C(G)$ is convex and attains a unique global minimum when $p_g = p_{data}$. Hence, with sufficiently small updates to p_g it will converge to p_{data}. This concludes the proof.

In very specific conditions with constraints on the model architectures and assumption that the discriminator is able to achieve it's optimal given the generator, the GAN converges. [6]

13 Application of Image Denoising

We now present an example of how image denoising framework can be used to unmask masked human images. Given an image of a person with a mask, our goal is to remove the mask and return an output image of the same person without the mask. In the times of Covid-19, it is certainly useful for identity verification at public places like airports without the person removing his/her mask. It can also be used as a built-in feature in the face recognition models to aid the verification process.

This type of image reconstruction task comes with its own challenges. It requires experimenting with GAN models with various architectures, loss functions and datasets to produce optimal results.

13.1 UNet architecture [7]

UNet evolved from the traditional CNNs and was specifically designed for processing biomedical images. In biomedical images we need to not only identify the possibility of an infection but also localize the area. UNets achieve this by performing classification at each pixel in the image. It has a U shape architecture (hence, the name) with 2 majors halves. The left part is the contracting path, normal convolutions, and the right part is the expansive one, transpose convolutions. The network makes use of 'skip connections', which propagates a convoluted image block to the deconvoluted one. Reference figure 6.

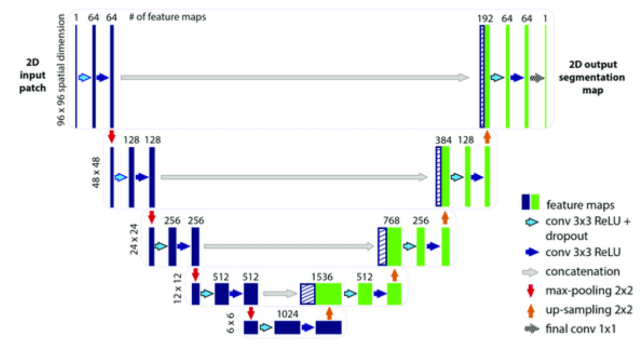


Figure 6: U-Net

13.2 Loss function

SSIM(Structural Similarity Index) measures the similarity between two images. It compares luminance, contrast and structure between the two images and returns a weighted combination of the three values between 0 and 1. SSIM values closer to 1 indicate high similarity. In contrast, loss functions like MSE(Mean Squared Error) or MAE(Mean Absolute Error) measure pixel-wise difference between the ground truth image and the generated image. The model needs to re-create the nose and mouth behind the mask. For this purpose, we chose SSIM as the loss function in our model, since pixel-level reconstruction is not necessary.

13.3 Training details

We trained a UNet+RESNET based model(ref. [8], [9]) on the CelebA dataset. We separated 1000 images for testing and did a 80-20% split of the rest of the dataset for training and validation. On fine-tuning the parameters, we obtained the best results when the input image has size 256x256x3, kernel size is 3x3, number of epochs is 20, batch size is 12 with ADAM optimizer.

13.4 Results

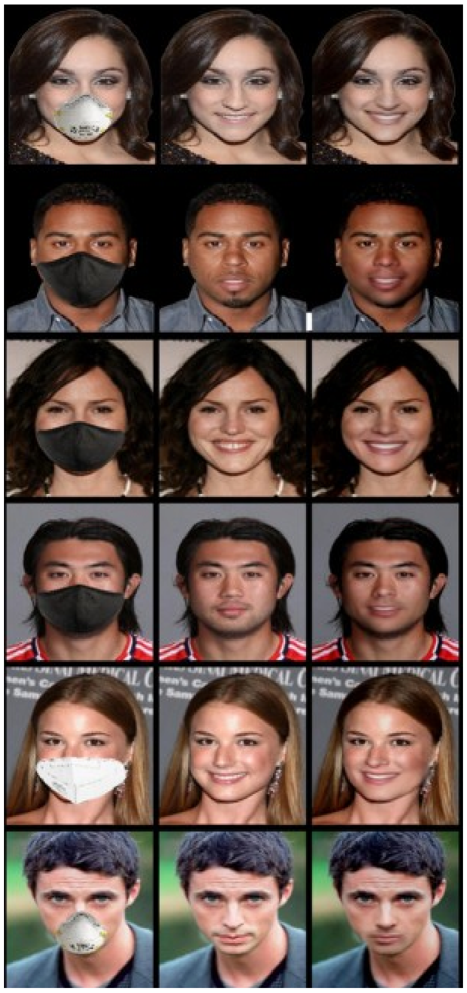


Figure 7: Test results: (masked input image | ground truth | generated output)

13.5 Improvements

1. The generated images under the mask are not as sharp and refined as normal human face images. We can train an additional a refinement network to resolve this.

2. The model heavily relies on the facemask being properly applied to the human face, therefore improvements in human face points detection (dlib) and application on face can drastically improve results.

3. Unmasking masked human faces has wide applications especially in security and privacy. Hence, constructing a model that does not rely on a large number of parameters, thereby is portable on mobile phones and other handheld devices is also worth mentioning.

References

[1] Fan L et al. “Brief review of image denoising techniques”. In: *Vis. Comput. Ind. Biomed. Art 2, 7* (2019).

[2] Ding-Xuan Zhou. “Universality of Deep Convolutional Neural Networks”. In: *Applied and Computational Harmonic Analysis* 48 (2018), pp. 787–794.

[3] CMU scribe notes. *Proof of Johnson Lindenstrauss Lemma.* eprint: `https://www.cs.cmu.edu/afs/cs/academic/class/15750-s18/ScribeNotes/lecture23.pdf`.

[4] Ohad Shamir and Tong Zhang. “Stochastic Gradient Descent for Non-smooth Optimization: Convergence Results and Optimal Averaging Schemes”. In: *International Conference on Machine Learning - ICML* (2013).

[5] Ian J. Goodfellow et al. “Generative Adversarial Networks”. In: *Advances in Neural Information Processing Systems* (2014).

[6] Asuman Ozdaglar Farzan Farnia. “Do GANs always have Nash equilibria?” In: *Proceedings of the 37th International Conference on Machine Learning* (2020).

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI* (2015), pp. 234–241.

[8] https://github.com/strvcom/strv-ml-mask2face.

[9] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Computer Vision and Pattern Recognition - CVPR* (2015).