# ON RANDOM WEIGHTS FOR TEXTURE GENERATION IN ONE LAYER CNNS

Mihir Mongia,[1] Kundan Kumar,[2] Akram Erraqabi,[3] Yoshua Bengio[3]
[1]*Stanford University*, [2]*IIT Kanpur*, [3]*Montreal Institute for Learning Algorithms*
*MMONGIA@STANFORD.EDU*, *KUNDAN@IITK.AC.IN*, *ERRAQABI@GMAIL.COM*, *BENGIOY@IRO.UMONTREAL.CA*

***Abstract*— Recent work in the literature has shown experimentally that one can use the lower layers of a trained convolutional neural network (CNN) to model natural textures. More interestingly, it has also been experimentally shown that only one layer with random filters can also model textures although with less variability. In this paper we ask the question as to why one layer CNNs with random filters are so effective in generating textures? We theoretically show that one layer convolutional architectures (without a non-linearity) paired with the an energy function used in previous literature, can in fact preserve and modulate frequency coefficients in a manner so that random weights and pretrained weights will generate the same type of images. Based on the results of this analysis we question whether similar properties hold in the case where one uses one convolution layer with a non-linearity. We show that in the case of ReLu non-linearity there are situations where only one input will give the minimum possible energy whereas in the case of no nonlinearity, there are always infinite solutions that will give the minimum possible energy. Thus we can show that in certain situations adding a ReLu non-linearity generates less variable images.**

**Key Words - Texture Generation, CNN, Random Weights**

## I. INTRODUCTION

With the recent advancements in deep learning, many image processing tasks that were once thought impossible are now possible. One such task is that of generating completely random textures that visually look similar to a given sample texture. Recently Gatys et al. [1] experimentally showed that by utilizing a specific energy function that uses the first few layers of a pretrained CNN, natural textures can be effectively generated. Later Champandard [2] used a different energy function and random weights to generate textures. He et al. [3] used the same method as in Gatys et al. [1] with random weights, to generate nice textures. Around the same time, Ustyuzhaninov et al. [4] experimentally showed that less variable natural textures can be generated using just one layer of a CNN with random filters along with the same energy function. Although these methods work quite well there is no theoretical analysis why these methods (in particular the choice of energy functions and also use of random weights) would be helpful for texture generation even in the most basic settings such as in [4].

This leads us to ask a natural question. What theoretical properties lead to the fact that we can generate random textures just using a one layer CNN with random filters?

In section 3 with a slight adjustment to the one layer CNN architecture, we show rigorously why random weights in one layer CNNs without a nonlinearity, can be used to generate random textures with the same performance as with pre-trained weights, while also drawing connections to previous work in texture generation. In section 4, we show how the behavior of generated images changes when one adds a ReLu non-linearity.We show that in the case of a ReLu non-linearity there are often cases where there is only one solution that gives the minimum energy (the input itself), whereas in the case with no nonlinearity there are infinite inputs that give the minimum energy.

## II. PRELIMINARIES

We now review a subset of the models used by Ustyuzhaninov et al. [4] that we later analyze theoretically. In particular the subset corresponds to the models which use random weights.

The model employs a single-layer CNN with standard rectified linear units(ReLus) and convolutions with stride one, no bias and padding (f-1)/2 where f is the filter-size (f is always an odd number). This choice of padding will ensure that the spatial dimension of the output is the same as the spatial dimension of the input. In addition, 363 filters of dimensions $11 \times 11 \times 3$ (filter width, filter height, number of input channels) are randomly generated from a uniform distribution according to [5].

For an original texture image $x$, a matrix $G^x$ is formed,

$$G^x_{ij} = \sum_{m=1}^{m=M} F_{im} F_{jm} \tag{1}$$

where $F_{ij}$ corresponds to the output of the ith filter (after the nonlinearity) at location j.

To generate new textures corresponding to an original image $x$, an energy function $E$ is developed,

$$E(y) = ||G^x - G^y||_F \tag{2}$$

An image y is initialized randomly to values between 0 and 1. Then $y$ is changed according to gradient descent until $y$ is a local minimum with respect to the energy function. Note that the smallest this energy can be is zero.

## III. THEORY FOR USING RANDOM WEIGHTS

In this section we consider the same model as above with two modifications. We consider a model where there is no non-linearity. We also use circular convolution rather than valid convolution as done in Saxe et al. [6]. We study the behavior of the algorithm without a ReLu non-linearity to gain insight into what is happening or not happening in the non-linear case. In addition, if the random weights in the model of Ustyuzhaninov et al. [4] were to be positive, then having a ReLu nonlinearity would be equivalent to using no non-linearity. This is because an image signal comes in values between 0 and 255. Thus the convolution of positive filters and an image signal will be positive. We use circular convolution, as in [6], rather than valid convolution because we can more easily get a theoretical analysis. Intuitively doing circular convolution in a texture image is not too different than doing valid convolution because the statistics are usually uniform across texture images.

An important fact to note is that the output of a circular convolution of an original image and filter $F_i$ can be written as a matrix multiplication where the matrix is a function of the filter $F_i$. These matrices are called block circulant with circulant blocks (BCCB) matrices. These matrices can be diagonalized in the form

$$F_i = U D_i U^{\dagger} \tag{3}$$

where $U$ is the discrete Fourier basis in two dimensions [7].Thus the output of convolving an image $x$ with a filter $F_i$ is

$$F_i(x) = UD_iU^\dagger x \tag{4}$$

where $x$ has been appropriately vectorized, and an entry of $G^x$ can be re-expressed in the following way,

$$\begin{aligned} G^x_{ij} = x^T UD_i^\dagger U^\dagger UD_j U^\dagger x = x^T UD_i^\dagger D_j U^\dagger x = x^T UD_{ij} U^\dagger x \\ = \sum_k |\lambda^x_k|^2 D^k_{ij} \end{aligned} \tag{5}$$

where $\lambda^x_k$ is the is the kth Fourier coefficient corresponding to the kth basis vector in $U$ such that

$$x = \sum_k \lambda^x_k U(:,k) \tag{6}$$

Ideally we would like to generate all the images $y$ such that $G^x = G^y$. This would correspond to an energy of zero. From the equations above we can see that finding a $y$ such that $Energy(y) = 0$ would come down to solving a system of linear equations. To be clear the system of equations would be

$$\begin{aligned} G_{11} = \sum_k |\lambda^y_k|^2 D^k_{11},\ G_{12} = \sum_k |\lambda^y_k|^2 D^k_{12},\ G_{13} = \sum_k |\lambda^y_k|^2 D^k_{13} \\ \dots \quad G_{nn} = \sum_k |\lambda^y_k|^2 D^k_{nn} \end{aligned} \tag{7}$$

In matrix form the linear system of equations would look like

$$g = M|\lambda^y|^2 \tag{8}$$

or

$$\begin{pmatrix} G^x_{1,1} \\ G^x_{1,2} \\ \vdots \\ G^x_{2,1} \\ \vdots \\ G^x_{n,n} \end{pmatrix} = \begin{pmatrix} D^1_{1,1} & D^2_{1,1} & D^3_{1,1} & \dots & D^n_{1,1} \\ D^1_{1,2} & D^2_{1,2} & D^3_{1,2} & \dots & D^n_{1,2} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ D^1_{2,1} & D^2_{2,1} & D^3_{2,1} & \dots & D^n_{2,1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ D^1_{n,n} & D^2_{n,n} & D^3_{n,n} & \dots & D^n_{n,n} \end{pmatrix} |\lambda^y|^2 \tag{9}$$

First we note that this system of equations is solving for the magnitude of the fourier coefficients of a particular signal. Thus this system puts no constraints on the phase of each fourier coefficient and thus, even in the case of M being full rank, there are always infinite solutions to equation 9.

In the case that the system of equations is not full rank, then the choice of filters clearly effects what type of solutions will give energy zero. This is because the vector $|\lambda^y|^2$ is unique up to the rows of M. This has some nice implications. Let's define

$$d_{ij} = diag(D_i^\dagger D_j) \tag{10}$$

Since we have constraints of the form

$$G_{ii} = d_{ij}^T |\lambda^y|^2 \tag{11}$$

it implies the possible solutions of $|\lambda^y|^2$ must lie on the $n-1$ dimensional space where equation 11 holds.Thus $|\lambda^y|^2$ lies within an intersection of roughly $n^2/2$, $n-1$ dimensional spaces that have the same projection on to the frequency domain of the filters as does $|\lambda^x|$. Notice if this low rank system were due to random weights then the possible solutions would lie within a different intersection of $n-1$ dimensional spaces. Thus using random weights would yield distinctly different images than would using pre-trained weights in the case where the number of filters is not large enough to create a full rank matrix $M$.

Notice that even if this system of equations is full rank, the solution to the linear system only gives us the magnitude of each fourier coefficient. Also notice that the number of equations is approximately $n^2/2$. Thus if we have a relatively small number random filters compared to the dimension of $x$, the matrix M can become full rank. If M is full rank and M is created either by random weights or pretrained weights, the solutions $y$ that yield energy zero are characterized by images for which the fourier coefficient magnitudes are the same as the fourier coefficient magnitudes of the image $x$. This is because we already know the fourier coefficients of the image $x$ already satisfy the equations from 7 and thus $|\lambda^y_k| = |\lambda^y_k|$ for any $k$. Thus in the case that there are enough random weights and pre-trained weights , the solution $y$ to the system of equations is no different than the solutions $y$ to the system of equations generated by pretrained weights.

In Figure 1 we show textures generated by simply randomizing phase of fourier coefficients. one can notice that the ridges happen in completely different places. One issue with this however is that one can see vertical and horizontal lines appearing in the generated textures. This is due to the fact that the FFT assumes images are periodic and in fact may suggest the need for nonlinearities.

## IV. Unique Solutions in the ReLU case with Circular Convolution

In the case where there is no nonlinearity, we show that in all cases there are infinite solutions that will yield energy equal to zero (The results above also hold for 1-D vector signals by simply letting $U$ be discrete fourier transform for one dimensional signals). Since in [4] a ReLU is used, the results above beg the question, does the same behavior hold in the case where there is a ReLU non-linearity. We show below that for almost all 1-D vector signals(as opposed to 2-D image signals) if one samples enough random filters that there is only one solution that yields energy zero.

First we set some notation. For an $x \in R^n$, let $C(x)$ denote the circulant matrix corresponding to x. In other words the first row of $C(x)$ is x. The second row is x circularly shifted by one and so on. Let the set $P_i(x)$ be all the vectors $v$ such that in the matrix-vector multiplication $C(x)v$ has only the $ith$ element greater than zero.

*Theorem 1:* Let $c_1$, $c_2$, ..., $c_n \in R^n$ be linearly independent vectors. Then the intersection of the half planes decided by these n vectors has non-zero volume.

*Proof:* Denote the half plane restrictions by

$$Cx < 0 \tag{12}$$

where the rows of C correspond to the vectors setting the half planes. $C$ is full rank because the rows are linearly independent. Now let

$$q_k = C^{-1} e_k \tag{13}$$

where $e_k$ are the canonical basis vectors of a euclidean vector space. Notice that as a result the $q_k$ will be linearly independent and thus form a basis for $R^n$ because there are $n$ such vectors. Thus any $x$ of the form

$$x = \alpha_1 q_1 + \alpha_2 q_2 + ... \alpha_n q_n \tag{14}$$

where $\alpha_i > 0$, satisfies the half plane restrictions. Since the region of possible $x$ is a convex cone constructed from basis vectors that span the whole euclidean space, the region of vectors x that satisfy the half-plane restrictions has non-zero volume. ■

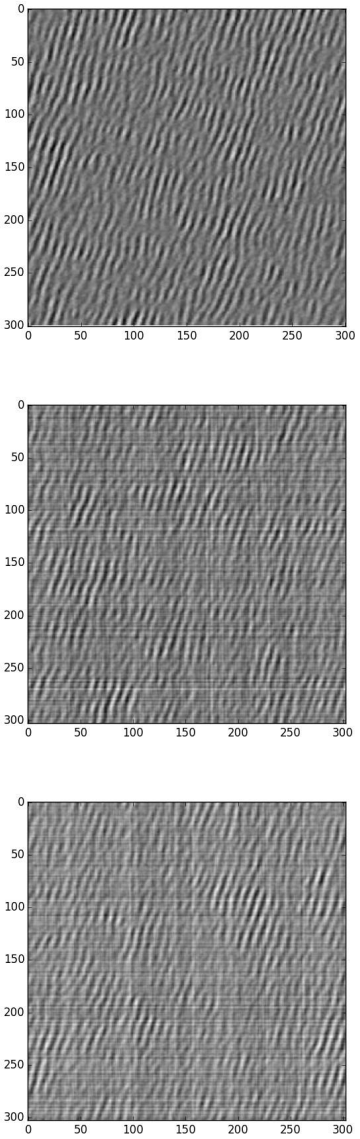

Fig. 1. This figure shows how randomly assigning phase to fourier coefficients can generate random textures. The top image is the original black and white texture. The bottom two images are generated textures.

Corollary: As long as $C(x)$ is full rank, $P_i(x)$ is a set that has nonzero volume in $R^n$ for any $i$. In other words, if you were to uniformly sample vectors from the unit ball, there is a finite probability that these vectors would be in the set $P_i(x)$ for any $i$.

*Theorem 2:* For any $G^x$ there are n vectors $y$ such that $G^x = G^y$. In particular, any y that is just a circularly shifted version of x will have a gram matrix $G^y = G^x$. This holds for any kind of non-linearity used in the network.

*Proof:* Consider $F^i$ to be the circulant matrix corresponding to i-th convolution filter. Then, output feature map corresponding to i-th convolution filter for input vector X can be written as $F^iX$. We'll use h(.) to represent the non-linearity that can be applied to any scalar or element-wise to any vector/matrix. Using this notation, we can say that

$$G^X_{ij} = (h(F^iX))^\top (h(F^jX)) = \sum_{k=1}^{n} h(<(F^i_{k,:}, X>).h(<F^j_{k,:}, X>) \tag{15}$$

. where $F^i_{k,:}$ is the $k^{th}$ row of $F^i$.

Now consider an operator Cs such that, Cs(x) circularly shifts x by one e.g. if $x = [x_1, x_2, x_3, ..., x_n]$, then $Cs(x) = [x_n, x_1, x_2, x_3, ..., x_{n-1}]$. Here, n is the length of vector x, $< a, b >$ is the dot product between vectors a and b. Then, $F^i$ being circulant implies that $Cs(F^i_{k,:}) = F^i_{k+1,:} \forall k \in [1, n-1]$ and $Cs(F^i_{n,:}) = F^i_{1,:}$. Also, it is straightforward to see $< Cs(a), Cs(b) > = < a, b > \forall a, b \in R^n$. Hence, we have

$$\begin{aligned} &< F^i_{k+1,:}, Cs(X) > = < Cs(F^i_{k,:}), Cs(X) > = < F^i_{k,:}, X > \\ \implies &h(< F^i_{k+1,:}, Cs(X) >) = h(< Cs(F^i_{k,:}), Cs(X) >) \quad \forall k \in [1, n-1] \end{aligned} \tag{16}$$

and

$$\begin{aligned} &< F^i_{1,:}, Cs(X) > = < F^i_{n,:}, X > \\ \implies &h(< F^i_{k+1,:}, Cs(X) >) = h(< Cs(F^i_{k,:}), Cs(X) >) \end{aligned} \tag{17}$$

This is true for all i in range [1,N] where N is the number of filters. Hence, from eq. 15 and eq. 16, 17, we have

$$\begin{aligned} G^{Cs(X)}_{ij} &= h(< F^i_{1,:}, Cs(X) >) * h(< F^j_{1,:}, Cs(X) >) \\ &+ \sum_{k=2}^{n} h(< F^i_{k,:}, Cs(X) >) * h(< F^j_{k,:}, Cs(X) >) \\ &= h(< F^i_{n,:}, X >) * h(< F^j_{n,:}, X >) \\ &+ \sum_{k=1}^{n-1} h(< F^i_{k,:}, X >) * h(< F^j_{k,:}, X >) = G^X_{ij} \end{aligned} \tag{18}$$

Hence, $G^X = G^{Cs(X)} = G^{Cs(Cs(X))} .. = G^(Cs^m(X))$. Hence, we get at least as many solutions as the possible circulations of x. ■

*Theorem 3:* Using the same notation set before Theorem 1, suppose we have $n$ sets of filters. Let each set $S_j$ contain $n$ filters (or equivalently vectors) each of which $\in P_i(x)$ for a unique $i$. In others words, if $v, z \in S_j$, then if $v \in P_i(x)$ for some $i$ then $z \notin P_i(x)$. Denote the set $S_{P_i}$ as the set of vectors $\in (\cup_{j=1}^{n} S_j) \cap P_i(x)$. Let the vectors in each $S_{P_i}$ be linearly independent. Under these conditions, if $G^x = G^y$ for some vector $y$, then $y$ is equal to $x$ or some circularly shifted version of $x$.

*Proof:* We denote the ReLU activation as $\mathscr{R}(x)$. Let's consider the set of $n^2$ filters in $\cup_{j=1}^{n} S_j$. For a given $x$, we want to characterize the solutions $y$ of the equation $G^y = G^x$. This equation is actually equivalent to a system of equations that correspond to the Gram matrix components

$$\forall i, j \in [1, \cdots, K] \qquad G^y_{ij} = G^x_{ij}$$

We can rewrite $G^x_{ij} = \mathscr{R}(F^ix)^\top \mathscr{R}(F^jx)$. Let's consider a diagonal $n \times n$ block of $G^x$ which corresponds to the filters coming from $S_i$ for some $i$. Without loss of generality suppose we have ordered the entries of $G$ such that the first diagonal block corresponds to filters in $S1$, the second diagonal block corresponds to the filters in $S2$ and so on. Without loss of generality let's consider the first block. We know that the corresponding filters in $S_1$ are defined such that only one component of $F^q_{S_1}x$ is positive, i.e $\forall q \quad \|\mathscr{R}(F^q_{S_1}x)\|_0 = 1$.Here $q$ is indexing the $n$ filters in $S_1$. Due to the conditions assumed in the theorem ($S_j$ contains $n$ filters each of which $\in P_i(x)$ for a unique $i$), we also have that $\mathscr{R}(F^{q1}_{S_1}x)^\top \mathscr{R}(F^{q2}_{S_1}x) = G^{S1}_{q1q2} * \delta_{q1q2}$ where $G^{S1}_{q1q2}$ corresponds to the appropriate entry in the first diagonal block of

$G^x$. Note that this implies the first diagonal block of $G^x$ is diagonal because of the $\delta_{q1q2}$ which is only non-zero when $q1 = q2$.

Given that we want $G^x = G^y$, it implies the first diagonal $n \times n$ block of $G^y$ must also be diagonal. This implies that for any $F^q_{S1} \in S_1$, $\|\mathscr{R}(F^q_{S_1} y)\|_0 = 1$ because if it were greater than 1, then the $n \times n$ block of $G^y$ will have non diagonal non-zero elements. If $\|\mathscr{R}(F^q_{S_1} y)\|_0 = 0$, then of course the $n \times n$ block would be the zero matrix.

This means that the only equations that characterize the solutions are those that correspond to the diagonal terms of the first block of $G^x$. These equations are $\|\mathscr{R}(F^q_{S1} y)\|_2^2 = G^{S1}_{qq}$. Since $\|\mathscr{R}(F^q_{S1} y)\|_0 = 1$, there are $n$ possible equations $F^q_{S1}[k,:]y = \sqrt{G^{S1}_{qq}}$ that could generate consistent vectors $y$, where $k$ corresponds to some row in the matrix (filter) $F^q_{S1}$.

Performing the same reasoning using the other sets of filters $S_j$, we end up with $n$ possible equations for each filter that characterize a solution y. $F^q_{S_j}[k,:]y = \sqrt{G^{S_j}_{qq}}$.

Note that the non-diagonal blocks of $G^x$ happen to be the inner product of the output of filters coming from $S_i$ *and* $S_j$ for some $j$ and $i$. These blocks contain also one non zero value per row for the same reason that make the diagonal blocks diagonal matrices. The value of knowing for which pairs of $F^{q1}_{S_i}$ and $F^{q2}_{S_j}$ produce a non-zero value is that it lets us know which matrices are aligned. In other words if the entry in $G^x$ corresponding to $F^{q1}_{S_i}$ and $S^{q2}_{S_j}$ is greater than zero, then if $F^{q2}_{S_j}[k,:]y = \sqrt{G^{S_j}_{q2q2}}$, then $F^{q1}_{S_i}[k,:]y = \sqrt{G^{S_i}_{q1q1}}$. Notice we used the same index $k$. Suppose we have aligned the above 2 filters. Then we can align those with $n-2$ other filters coming from other $S_j$, by looking in the appropriate non diagonal blocks of $G^x$.

Since we have aligned the filters appropriately, we now have n possible system of equations since we can choose n values of k. We note under the conditions specified in Theorem 3, that these system of equations is full rank, and can thus only have one solution. One can also see that each of these system of equations just gives a circularly shifted solution of $x$. This phenomenon corresponds to theorem 2. ∎

*Theorem 4:* Suppose we generate $nk$ random filters from the infinity norm 1 ball and suppose $C(X)$ is full rank. As $k$ approaches infinity, the probability that there is unique solution to $G^x = G^y$ (up to a circular shift) approaches one.

*Proof:* Let $\delta_1 ... \delta_n$ be the probability that a random vector lands in $P_1(x) ... P_n(x)$ respectively. We know $\delta_1 ... \delta_n$ is greater than zero because of theorem 1. Let $k = c * n$ where $c$ is some rational number. Let $count_i$ be the number of times we get filters in $P_i(x)$ in the $ith$ set of $k$ draws.

$$\begin{aligned} &P(count_i < n) = P(count_i < (\frac{1}{c\delta_i}) cn\delta_i) \\ &= P(count_i < (1 - \frac{c\delta_i - 1}{c\delta_i}) cn\delta_i) \\ &< exp(-\frac{c\delta_i - 1}{2c\delta_i} \frac{c\delta_i - 1}{c\delta_i} c\delta_i n) \end{aligned} \tag{19}$$

This follows from classical chernoff bound results. Notice the term $\frac{c\delta_i - 1}{c\delta_i}$ approaches 1 as $c$ approaches infinity.

Thus

$$\begin{aligned} &P(count_1 > n, count_2 > n, \dots count_n > n) \\ &= \prod_{i=1}^{n} (1 - exp(-\frac{c\delta_i - 1}{2c\delta_i} \frac{c\delta_i - 1}{c\delta_i} c\delta_i n)) \end{aligned} \tag{20}$$

which approaches 1 as c approaches infinity. In addition, given that we have sampled more than $n$ vectors that are in $P_i(x)$ for some $i$, if we choose n of those vectors they are going to be linearly independent with probability one since the set of linearly dependent vectors in $P_i(x)$ has measure zero.Thus as $c$ approaches infinity we have the conditions noted in theorem 3 with probability one. One can find the sets $S_1, S_2, \dots S_n$ through exhaustive search of the Gram Matrix and then execute the proof for theorem 3 and come to the conclusion that if $G^x = G^y$ for some vector $y$, then $y$ is equal to $x$ or some circularly shifted version of $x$. ∎

## V. Connections to Previous Work

Galerne et al. [8] show that an algorithm called Random Phase Noise (RPN) is effective at generating realistic textures of a certain class. In the most basic version of their algorithm, the authors take texture like images and generate new textures simply by randomizing the phase of the fourier coefficients.As long as there are enough filters to create a full rank system, the slightly modified model in section 3 is mathematically equivalent to randomizing phase. Thus we expect that even our modified model(with no nonlinearity) should produce good random textures, which is in fact what we observe in Figure 1. The authors note the same horizontal and vertical line artifacts with this algorithm as we do in our experiments and thus do some extra image processing to get textures that do not have line artifacts.

## VI. Discussion

We have shown rigorously that in a slightly simplified model compared to that of [4] (with circular convolution and no ReLU nonlinearity), there is no need for pretrained weights. To be precise, if the number of weights is small, then the images generated with pretrained weights and random weights will be different from each other. However, as the number of weights increases, random or pretrained weights make no difference in the images generated. We show that the images generated from a large number of weights are just modified versions of the original image. Frequency coefficient magnitudes are preserved, whereas frequency coefficient phases are randomized.

Using the lack of a nonlinearity as a starting point and the fact that in [4] a ReLU nonlinearity is used, we seek to see if having a ReLU would generate a set of signals with similar properties as in the no ReLu case. We show rigorously in the one-dimensional case that there are conditions under which the only signals $x$ that can be generated with the same gram matrix as the original signal $x^*$ are circularly shifted versions of $x^*$. This is distinctly different from the linear case where there would be an infinite number of solutions.

We view these results as giving a theoretical starting point to understand the contribution of nonlinearities, random weights, as well as the gram matrix energy function in texture generation. To be clear though, there is room for future work. Since the energy function is non-convex with respect to input images, input images may be caught in a local minima. We have no way to get a handle on what type of images these local minima might produce. Secondly we have not shown in full generality what happens in the Relu case. We have shown just sufficient conditions under which the solutions to a Relu model and linear model differ greatly. Thirdly our analysis does not yet account for multiple layers. On the other hand, the analysis presented above may provide guiding intuition for multiple layers since the output of each layer is just a linear convolution and ReLu of the previous layer.

## REFERENCES


[1] Leon A. Gatys, Alexander S. Ecker and Matthias Bethge. Texture Synthesis Using Convolutional Neural Networks, 2015; arXiv:1505.07376.

[2] Champandard, Alex. "Extreme Style Machines: Using Random Neural Networks To Generate Textures". nucl.ai. N.p., 2016. Web. 10 Aug. 2016.

[3] Kun He, Yan Wang and John Hopcroft. A Powerful Generative Model Using Random Weights for the Deep Image Representation, 2016; arXiv:1606.04801.

[4] Ivan Ustyuzhaninov, Wieland Brendel, Leon A. Gatys and Matthias Bethge. Texture Synthesis Using Shallow Convolutional Networks with Random Filters, 2016; arXiv:1606.00021.

[5] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In Aistats, vol. 9, pp. 249-256. 2010.

[6] Saxe, Andrew, Pang W. Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y. Ng. "On random weights and unsupervised feature learning." In Proceedings of the 28th international conference on machine learning (ICML-11), pp. 1089-1096. 2011.

[7] Hansen, Per Christian, James G. Nagy, and Dianne P. O'leary. Deblurring images: matrices, spectra, and filtering. Vol. 3. Siam, 2006.

[8] Galerne, Bruno, Yann Gousseau, and Jean-Michel Morel. "Random phase textures: Theory and synthesis." IEEE Transactions on image processing 20, no. 1 (2011): 257-267.

[9] Horn, Roger A., and Charles R. Johnson. Matrix analysis. Cambridge university press, 2012.