Beyond CNNs: Encoded Context for Image Inpainting with LSTMs and Pixel CNNs

Taneem Ullah Jan, Ayesha Noor
Computer Science & IT University of Engineering and Technology Peshawar, Pakistan
***Correspondence**: taneemishere@gmail.com, ayesha.noor2324@gmail.com.

Citation | Jan. T. U, Noor. A, "Beyond CNNs: Encoded Context for Image Inpainting with LSTMs and Pixel CNNs", IJIST, Special Issue pp 165-179, May 2024
Received| May 04, 2024 **Revised|** May 15, 2024 **Accepted|** May 20, 2024 **Published|** May 26, 2024.

Our paper presents some creative advancements in the image in-painting techniques for small, simple images for example from the CIFAR10 dataset. This study primarily targeted on improving the performance of the context encoders through the utilization of several major training methods on Generative Adversarial Networks (GANs). To achieve this, we upscaled the network Wasserstein GAN (WGAN) and compared the discriminators and encoders with the current state-of-the-art models, alongside standard Convolutional Neural Network (CNN) architectures. Side by side to this, we also explored methods of Latent Variable Models and developed several different models, namely Pixel CNN, Row Long Short Term Memory (LSTM), and Diagonal Bidirectional Long Short-Term Memory (BiLSTM). Moreover, we proposed a model based on the Pixel CNN architectures and developed a faster yet easy approach called Row-wise Flat Pixel LSTM. Our experiments demonstrate that the proposed models generate high-quality images on CIFAR10 while conforming the L2 loss and visual quality measurement.

Keywords: Index Terms—Image Inpainting; GAN; Pixel CNN; LSTM

Introduction:

Image in-painting involves reconstructing damaged or missing parts within an image, commonly applied in the restoration of old photos or paintings and image editing tasks. Notably, tools like Photoshop feature a robust completion tool that doubles as a removal tool. Despite Convolutional Neural Networks (CNNs) surpassing human classification accuracy on ImageNet [1], in-painting outcomes still fall short of human predictions. The challenge lies in the vast number of possible ways to fill an $8 \times 8 \times 3$ section, around $50{,}000$ possible ways, while ImageNet has only $32{,}000$ classes. Interestingly, humans effortlessly mentally reconstruct missing image sections by comparing context with their knowledge of the world, enabling scene and object recognition, as well as extrapolating missing elements from memory. Artificial and computer-based methods leverage similar principles in their approach.

There exist two categories of approaches: local methods [2][3] solely rely on contextual information, such as color or texture, and aim to extend and blend these details seamlessly. These techniques demand minimal previous perception and training. For instance, if there are no eyes on the head part, a local method might replace it with a patch of skin-textured pixels. However, these methods face limitations in scenarios where larger patches are absent, excelling primarily in tasks like watermark removal. On the other hand, more advanced methods adopt a global, context-based, and semantic approach [4][5][6]. These methods identify patterns within images, like a door or a cabin, which leverage this understanding to infuse the empty spots. Unlike local techniques, they hold the importance of specific elements, such as the necessity of a nose in a particular facial position, constructing a fitting replacement based on their broader knowledge of the context.

An attractive aspect of such problems lies in the effortless generation of extensive datasets for training. Some datasets of images like ImageNet and CIFAR10 [7], which we used for our convenience, can be readily pre-processed by introducing alterations to the images. This approach allows the generation of hundreds of millions of training examples, enabling the training of larger deep networks.

The Problem:

Every image is divided into two segments: the portion that is absent and under reconstruction, and the contextual part. To enhance simplicity, we suppose that the section which is missing, is a square of dimension $n \times n$. However, it is worth mentioning that the functionality of the network remains consistent even when dealing with arbitrary removals. See figure 1.

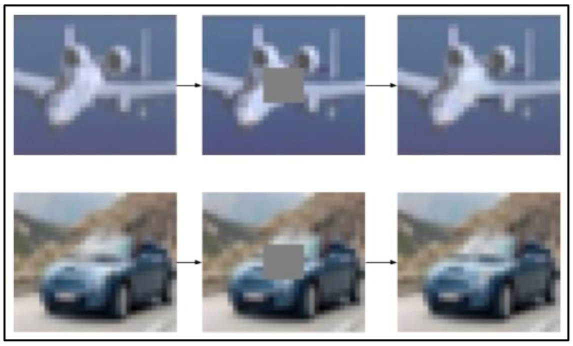

Figure 1: Image In-Painting

The concept of image in-painting is commonly presented as a constrained image generation challenge. The network is tasked with receiving a contextual input and generating an image with identical dimensions as the absent patch. The ultimate assessment hinges on the average element-wise L2 [8], the distance between the original missing section $Y \in R^{n \times n \times 3}$ and the predicted counterpart $\widehat{Y} \in R^{n \times n \times 3}$. In our illustrative instance using CIFAR10, $n = 8$. For a given sample i, the loss is calculated as follows:

$$L = \frac{1}{n^2} \sum_{p,q,r} \left(Y^{(i)}{}_{p,q,r} - \widehat{Y}^{(i)}{}_{p,q,r} \right)^2$$

We introduce an evaluation metric termed approximate exact-match (AEM), which is solely used for assessment purposes. We observe that a slight move of one or two elements in the value of the pixel channel has minimal visible impression, the effect can be seen in figure 2. Therefore, if a calculated pixel value falls from the accurate image value in the range of ± 5 at each channel, this qualifies as the same and or equal. We present the mean-AEM, denoted as MAEM, where a value of 100% implies that the visual impression of the image is almost identical to the original image, a simpler version of Generated Image Quality Assessment (GIQA) [9].

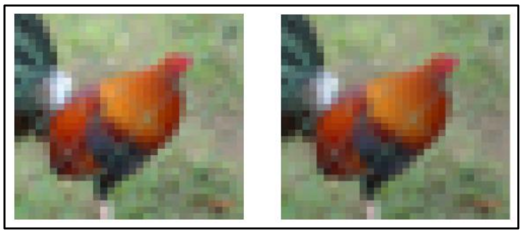

Figure 2: On the left is ground truth example and on the right is a ±5 randomly added to each pixel channel

In-painting comes in two forms: blind [10][11], where the network lacks information about the position and shape of the missing area. On the other hand, in the non-blind [12], such details are provided within the inputs. Extensive research indicates that blind in-painting poses a hard challenge. While non-blind in-painting is more extensively documented, there remains considerable scope for enhancement. Consequently, our emphasis is placed on the latter, reflecting a deliberate choice to concentrate efforts on non-blind in-painting, recognizing its potential for further advancements.

Our primary goal is to make use of the latest computer vision methodologies to develop a resilient and well run in-painter. Here we aimed to attain satisfactory outputs in the form of mean square error or $L2$ loss, benchmarked against current models. Initial results and the existing methods indicate minimal empirical distinctions between utilizing a square-shaped mask and employing randomly selected rectangular shape masks in the middle of given images. Therefore, for implementation simplicity, our focus primarily revolves around centered square-shaped cover ups of a consistent size. Specifically, on dataset like CIFAR10, this involves the removal of a patch from every image at the center of size 8×8.

Related Work:

Various researchers have investigated a diverse range of methods to tackle such challenges. A notable work by Pathak et al. [13], from where our initial inspiration took root. They adapted the conventional Generative Adversarial Network (GAN) [14] model by incorporating contextual information of image, rather than using stochastic noise, for predicting the incomplete section. Highlighting the importance of Leaky ReLU as detailed in [15] within the discriminator, and exclusion of pooling layers, they implemented compression and decompression operations with strides differing from 1. Their model training involved a combination of $L2$ loss and adversarial loss, measuring the success of the generator in deceiving the discriminator. However, our study revealed that this increased the risk of overfitting by utilizing fully connected layers in several instances. On the other hand, they have used relatively simple CNN model architecture as for encoders and decoders. In our work, our aim was to not only explore this but also investigate modern methodologies renowned for state-of-the-art results in domains like VGG [16], Inception [17], and others similar.

In a publication [18], their proposed approach revolves around modeling images as the conditional probability product distribution. Here the objective was to calculate the image pixels in sequential order, such as to the bottom-right point from the top-left. These functions which calculate conditional probability are shaped to either capture the contextual information of pixels within the upper rows with the help of recurrent networks or by employing CNNs to operate on local pixels (Pixel CNN [18]). Although originally designed for image generation, this method proves adaptable for our objective of optimizing the probability of the reconstructed image by providing pixel data.

Yang et al. [19] present a technique aimed at addressing the in-painting of large sections within extensive images. Traditional models encounter challenges in producing sharp results for such tasks, often resulting in blurry outputs with noticeable edges between the contextual information and the reconstructed region. The authors reduce this issue by incorporating hierarchical approaches to introduce fine-grained features above the regenerated spots, enhancing the resolution of in-paintings. Their approach involves training two distinct networks; first is a feature extractor that is assessed using a content-based loss same as in [13], second is texture-based network which employs minimizing the texture-based local loss. The incorporation of perceptual equivalence ensures that the patterns within the generated spots closely align with the context of local texture. This strategy significantly improves visual sharpness, it may be considered unnecessary in our case, given the context of working with smaller images.

The structure and process of our work represent a substantial adaptation of the CIFAR10 classification pipeline, originally derived from resources. The only retained elements are the queuing system and monitoring session. It is noteworthy that the autoencoders, GANs, and Wasserstein GANs (WGAN) [20] utilized in our study are also publicly available through standard sources. Moreover, there are adaptations of [13] in existence, their inefficiency and non-functionality led us to avoid their use. In the exploration of advanced models such as VGG, Inception, and ResNet [21], we adapted them from model repositories, and made adjustments to tailor them to CIFAR10 specifications.

Methods:

Dataset:

Our primary dataset is CIFAR10, comprising images of dimensions $32 \times 32 \ \times \ 32$. It consists of around 50 thousand training and about 10 thousand test samples. Notably, the CIFAR10 has been taken from the Tiny Images dataset [22]. When our model achieves stability, we extend our training efforts to leverage this larger dataset. The inclusion of a more extensive variety of images from the larger dataset proves helpful, particularly given that CIFAR10 is limited to only 10 classes. The substantial number of samples in the larger dataset serves as an efficient countermeasure against overfitting. Our model facilitates seamless scaling for training on this expanded dataset, resulting in high training performance.

Additionally, in the middle we also incorporate data augmentation steps on our CIFAR10 that helps in amplifying the dataset further enhancing the adaptability of this model for minor variations effectively. This involves introducing small, random adjustments to hue, saturation, contrast, and applying random Gaussian blur to the images. Also we put zeros in place of the middle $8 \times 8 \times 3$ crop.

Autoencoders:

In-painting constitutes a subset of a broader category of image generation problems involving the creation or modification of pixels. Tasks like deblurring, denoising, and small-scale blind in-painting, such as text removal, are commonly addressed using autoencoders. Autoencoders typically consist of two main components, an encoder and a decoder. Initially, an image is encoded in a latent feature space, or embedding. On the other hand, the decoder reconstructs the original image based on this embedding. The training process involves jointly optimizing both networks to decrease the input-output disparity. This architectural configuration requires the encoder to maximize the information encoded into the embedding. The encoder must learn the abstract intelligent features to compress the information in least possible loss, that too within the limited size of this latent space. In CIFAR10 experiments, where input images consist of vectors with a length of 3072, it has been found that employing a bottleneck size of 512 and or 1024 produces satisfactory results. Larger bottlenecks do not provide any attraction to autoencoders to discover accurate representations but keeping raw sections of pixels proves to be sufficient. On the other hand, using smaller bottlenecks lack sufficient capacity to encode the input information. This can be seen in figure 3.

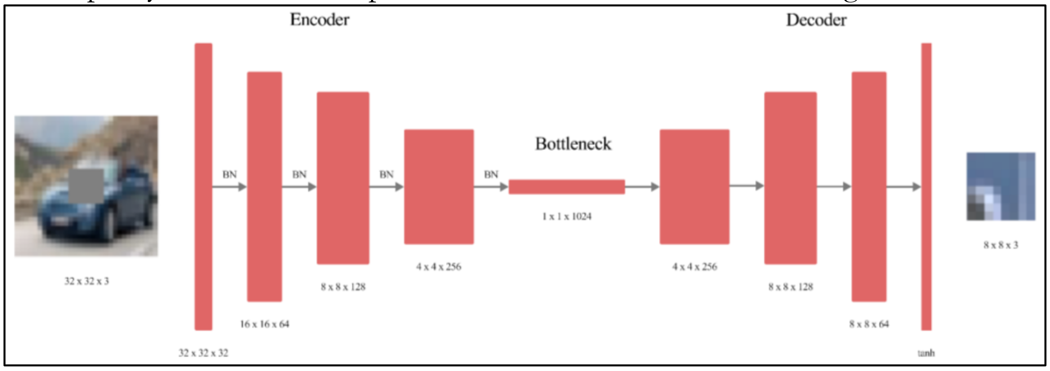

Figure 3: Autoencoder architecture of our model

The typical ratio between the image and latent space dimension in the existing literature often exceeds than what we employ, ranging from 3 to 6. Empirically, our choice stems from working with small images, introducing a challenge in identifying unified entities and components in the target image. Achieving improved encoding ratios, as seen in ImageNet; 12 or more, for instance, is more feasible because it is comparatively easier to isolate high-level features in larger images like those in ImageNet.

To establish a baseline, we initially constructed a straightforward CNN architecture. The input dimension is $32 \times 32 \times 3$, and the output dimension is $8 \times 8 \times 3$. Guidance from the literature advocates for the use of several small filters rather than larger ones. The rationale behind this approach lies in the ability to achieve an equivalent receptive field with deeper networks. Throughout, we utilized filters of size 3 exclusively. While filters of size 5 and 7 were tested on multiple occasions, their performance consistently lagged behind that of size 3 filters.

Our experimentation involved a $[\text{Conv} - \text{Conv} - \text{Pool2}] \times 3$ architecture, followed by either 2 fully connected layers or 2 convolutional layers. This architecture closely resembles a well-performing design on CIFAR10, featuring smooth dimension reduction coupled with an increment in the number of filters, initiating at 32 or 64 and doubling at each pooling step. Subsequently, we opted to eliminate max pooling by entirely substituting it with convolutional layers of stride 2, maintaining the same filter progression. Transitioning from the $4 \times 4 \times 256$ convolutions to the bottleneck can be conceptualized as a stride 4 operation with 1024 filters. Across all scenarios, ReLU activation consistently outperformed alternatives. The incorporation of Batch Normalization on each layer given substantial performance improvements, approximately around $\pm 15\%$, with comparable execution times.

In terms of qualitative assessment, as previously mentioned, a prominent issue is blurriness: while colors are generally accurate, details and textures tend to be lost, resulting in predicted sections that often resemble indistinct dots, failing to seamlessly blend into the image. To address this challenge, our objective is to diminish visible continuity errors between the predicted section and the context. We aim to achieve this by predicting a patch that slightly overlaps with the context and imposing a robust penalty on the loss specifically within this overlap region.

Generative Adversarial Networks (GANs):

The first figured-out method demonstrated notable L2 loss. Nevertheless, the generated output images consistently exhibited blurriness and a lack of intricate details, as illustrated in figure 4. The inherent nature of L2 loss encourages the network to adopt a risk-averse approach, generating safe predictions characterized by a lack of sharp shapes and substantial changes across the patch. The inclination to generate blurred, average-color images derived from context minimizes the occurrence of substantial errors. Consequently, while the model excels in terms of the L2 norm, the generated outputs fall short of realism when evaluated by a human eye.

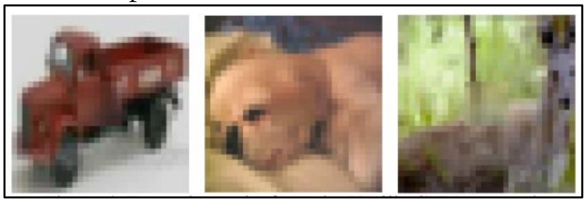

Figure 4: Results from normal CNN architecture

Deep Convolutional GANs (DCGANs):

To encourage our network to take more risks and generate realistic outputs, we opted to explore GANs. Here the underlying objective is the emulation of visual and perceptual evaluation and assessment. For instance, if a model predicts a human head without eyes or mouth, this is considered superior to generating an image with a black spot at the center. The rationale is that an averagely blurry image will never appear realistic to a human observer.

In the realm of GANs, the key concept involves training a discriminator network (D) concurrently with a generator network (G). The discriminator learns to assess the authenticity of an image, distinguishing between real and or generated spots and dots. Normally they are on both real and generated examples with distinct labels for each. The generator faces a penalty with an increased loss if the output image is viewed as generated by the discriminator. To outsmart the discriminator, our generator aims to create natural looking and visually clear images. As the discriminator improves, both networks benefit from the feedback loop, driving mutual enhancement.

The introduction of adversarial networks may not necessarily lead to an improvement in $L2$ loss, as the generator could throw significant errors. Although, the primary objective is to enhance the realism of the generated images. In the end, the crucial aspect is whether the predicted human eye, for example, appears authentic within the context of the image. Even if the $L2$ loss indicates a substantial difference from the ground truth; it constitutes an acceptable result, if the predicted sections are admissible in the context of the human head. Here the focus shifts from minimizing pixel-wise differences to creating outputs that are visually convincing and contextually appropriate.

Now, loss $L = \alpha L_{rec} + (1-\alpha) L_{dics}$, here L_{rec} is the same mean square error, while L_{disc} known as the probabilistic output from the discriminator for the generated images. This also can be called sigmoid cross entropy. Given that α changes the scale of the loss, using the loss itself for the optimization of this crucial hyperparameter is impractical. This challenge is to intricately linked to the observation that a lower $L2$ loss should not automatically correlate with script and visually better output. The choice of α is aimed at optimizing the results of our output samples and ensuring. Overall, this can be calculated as:

$$L_{disc} = -\sum_i \quad log(p_i) = -\sum_i \quad log\left(D\left(\hat{Y}^{(i)}\right)\right)$$

The discriminator, in our setup, is a network consisting of six convolutional layers, having a stride value of 2. After this, there is a final convolutional layer that reduces the depth value to 1, resulting in a normal value output. Conceptually, a typical classification task is addressed by our discriminator. Therefore, various recent methods can allow them to function as a typical generator network. We leveraged pre-trained models, including VGG, Inception, and ResNet, to enhance our discriminator. Since these models are designed for larger inputs like ImageNet, we attempted to pad our images, but the results were unsatisfactory. As a solution, we adjusted these techniques to operate on our smaller inputs. This involved removing the first layers with dimensions higher than 32×32 and injecting our input into the initial smaller layers. To prevent overfitting and improve execution time, we also reduced the depth and the number of filters in consideration of the fact that our input contains less information. The same adaptation strategy was applied to the encoder, where we experimented with a variety of high-performing models.

Learning Tricks:

As usual we encountered challenges in effectively training a GAN. Visual inspection of the predicted images revealed the presence of colored artifacts, as illustrated in figure 5. Upon closer inspection, it became apparent that shapes and color gradients aligned with the context, but the colors were distorted, particularly noticeable in the case of the plane. Due to difficulties in achieving satisfactory convergence with GANs, the $L2$ loss was higher compared to vanilla autoencoders. The current architecture exhibits a sensitivity to randomness, where the exact same method may result in convergence or divergence under different circumstances.

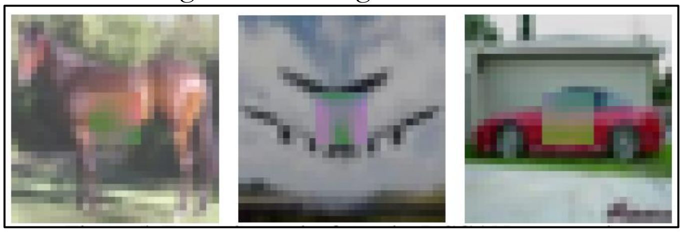

Figure 5: Results from DCGAN architecture

Training GANs poses inherent challenges as they are often unstable and highly sensitive to network architectures and parameters setting. Finding the right hyperparameters and architecture details can be a tedious task. Another common issue encountered is the imbalance between the discriminator and the generator, where one may exceed and overwhelm the other. As an example, if out network's discriminator becomes excessively robust, then our generator will struggle to deceive it, leading to a scenario where the adversarial loss sharply rises at high values, increasing the $L2$ loss. Conversely, if the generator becomes too dominant, it can successfully outsmart the discriminator, causing it to halt learning and assign identical probabilistic values to both fake and real samples. Then, it is symmetrical to have no adversarial loss. To address these challenges, we experimented with a few learning tricks aimed at mitigating these issues.

Separate Batches:

We provided two distinct batches: one containing solely real samples and the other composed entirely of fake samples. This approach facilitates clearer and more direct updates for each batch, allowing the network to focus distinctly on improving its understanding of real examples and enhancing the detection of fake ones.

Soft and Noisy Labels:

Soft and noisy labels were employed following the approach outlined in [23]. True images were assigned random labels between 0.9 and 1.1, while fake images received labels ranging from 0 to 0.2. Additionally, labels were occasionally flipped randomly between classes.

These strategies introduce noise and enhance the robustness of the discriminator, contributing to improved training stability.

Maximizing log D Instead of Minimizing log (1 − D):

Rather than minimizing $log(1-D)$, we chose to maximize $logD$. Here D represents the discriminator's output. While these formulations are equivalent in this context, the latter avoids the issue of vanishing gradients early in the training process, enhancing the stability and effectiveness of the learning procedure.

WGANs:

To address the challenges in training GANs, as discussed above, WGANs [20] have also been explored as an alternative to traditional GANs, relying on the Earth-Mover distance [24] for learning distributions. WGANs minimize the Earth-Mover distance, allowing for the estimation of this metric during training, which correlates well with visual quality. WGANs also eliminate the need for a realness probability from the discriminator, using an unbounded score, and enforce Lipschitz continuity through gradient clipping. Importantly, WGANs do not require a delicate balance between the generator (G) and discriminator (D), allowing for training the discriminator to convergence at each step. In practice, we train the discriminator 10 times at each iteration for stability and convergence.

Reducing Continuity Errors:

The adoption of WGANs marked a significant improvement in training a robust adversarial network. However, a noticeable issue persisted; the distinct visibility of the border between the context and the reconstructed region. Despite the challenge in visually explaining this clear border appearance, a detailed analysis revealed the absence of an organized color correction with pixels was highly noticeable, but a random error tended to blend more seamlessly, refer to figure 2. This border problem remained even in instances with otherwise accurate predictions.

To address this issue, we implemented a clever trick: predicting a 16×16 center square that encompasses the original 8×8 target along with a an each side overlap of 4-pixels. While we penalized our reconstruction loss more than 20 times, it is relatively simple for the model to accurately calculate and predict it. Here this strategic approach significantly enhances the quality of border merging. By ensuring that there are no major discontinuities within the output, this effectively reduced gaps between the input and prediction. It is important to note that, for visualization purposes, we retain and display only the 8×8 target in the final results.

Prior to using this strategic trick, feeding the entire image to the discriminator posed challenges, as the obvious border served as a key indicator for determining the authenticity of the image. However, with the introduction of the overlap and the subsequent reduction in the visibility of borders, we successfully transitioned to providing the complete image to the discriminator. In this improved setting, the discriminator could assess the entire image as a cohesive unit. Notably, an 8×8 square, either from a real or fake image, typically lacks sufficient meaningful information on its own, making it challenging for the discriminator to make a decisive meaning. In contrast, evaluating the entire image facilitates a clear and more straightforward return for the discriminator. Overall the results can be seen in figure 6.

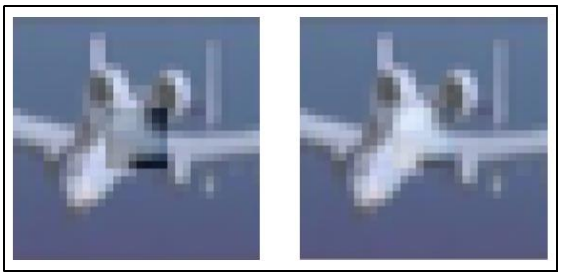

Figure 6: On the left is the result without overlap trick and on the right is with overlap

Density Based Models:

The Idea:

We opted to implement unsupervised models alongside our supervised methods for filling the contextual portion in the center of an image. In this approach, our objective is to approximate the function of Probability Mass (PMF), p of all images by multiplying conditional probabilities. Therefore, for a given image x of dimension $n \times n$, where the pixels are denoted as $\{x1, x2, \dots, xn^2\}$, the PMF is expressed as:

$$p(x) = \prod_{i=1}^{n^2} p(x \mid x_i, \dots, x_{i-1})$$

Where $p(x \mid x_i, \dots, x_{i-1})$ denotes the probability of the i^{th} pixel or x_i assuming the values $x1, \dots, x_{i-1}$ have been observed in the previous pixels.

Each pixel is represented by a triplet of integers as: $(R, G, B) \in \{0, 256\}^3$. Now using this notation $x_i = (x_{i,R}, x_{i,G}, x_{i,B})$ and $x < i = (x1, \dots, x_{i-1})$, the conditional probabilities can be written as:

$$p(x_i \mid x < i) = p(x_{i,R} \mid x < i) \times p(x_{i,G} \mid x < i, x_{i,R}) \times p(x_{i,B} \mid x < i, x_{i,R}, x_{i,G})$$

Therefore, the final probability $p(x)$ is obtained by multiplying $3n^2$ terms. Consequently, while in-painting images, our aim is to populate the space with the pixels by maximizing this probability.

The approach we take is greedy, which means that assuming we have learned the characteristics of the function p and are given an image with m missing pixels at positions $i_1 < i_2 < \dots < i_m$, we will first set the value $x_{i1,R}$ given $x < i_1$, then the value of $x_{i1,G}$ given $(x < i_1, x_{i1,R})$, and eventually we set $x_{im,B}$ given $(x < i_m, x_{im,R}, x_{im,G})$. Hence, as this approach is easy to implement and is computationally efficient, there is no assurance that our reconstructed image maximizes the likelihood $p(x)$.

An intuitive approach is to establish an order on the pixels, progressing from the top-left corner to the bottom-right corner, scanning rows consecutively. Once the order is defined, the next step involves selecting the type of conditional probability to be learned. We will introduce two distinct methods, drawing inspiration from [18]. The first method involves a Pixel CNN, where the conditional probability of a pixel is determined through a convolutional neural network operating on pixels in the surrounding neighborhood. The second method is a flattened row Long Short-Term Memory (LSTM), changing significantly from the one proposed in [18]. In this alternative approach, the probability is computed based on pixels from preceding rows, which are then fed into an LSTM network.

Pixel-CNN:

In order to calculate the conditional probability $p(x_i \mid x < i)$, a convolutional neural network is implemented. Various architectures were explored, incorporating combinations of convolutional and leaky ReLU layers. The model concludes with three fully connected layers, each followed by a softmax layer corresponding to the three-color channels (R, G, B). Our model applies softmax loss function, computing the estimated probabilities for the true pixel values $\hat{p}_{i,true}$ in an image with n^2, are computed as:

$$L_{softmax} = -\sum_{i=1}^{n^2} log(\hat{p}_{i,true})$$

An alternative perspective argues that the SoftMax loss may not be directly applicable, considering that the task at hand is not a classification one. In this context, calculating the value of 117 rather than 118 could not be as crucial as predicting 117 instead of 245. While it is obvious that a low probability $\hat{p}(x_{i,R} = 120)$ may not pose a significant issue. If the emphasis on values within the range (112, 124) is sufficiently enough, the softmax loss function has shown interesting results, even given the extensive training examples compared to the 256 possible values for each channel. The model we came up with finally incorporates a 5x5 section within the image for the initial convolutional layer, positioned at the top-left corner of the pixel grid. See figure 7.

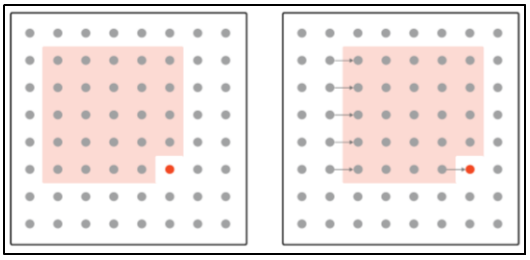

Figure 7: The red pixels are surrounded by gray dots; these are used for predicting the conditional distribution which form the neighborhood. Subsequently, to estimate the distribution of the next pixel we the same network, shifted one step to the right

Our method offers the advantage of utilizing a similar model, not only this but provides a similar set of learning parameters to compute the final conditional probabilities for each. The complexity remains persistent, irrespective of the image size, as we consistently use the same number of pixels; 24 to calculate the distribution of the next pixel. However, in the case of resolutions, like 128×128 images, this may be necessary to employ a greater relative section for more accurate predictions.

The outcomes obtained with our Pixel CNN model exhibit a slightly lower performance compared to GANs when evaluating the $L2$ loss. Two factors contribute to this disparity:

- Our Pixel CNN was trained using a SoftMax loss, which explains the higher $L2$ loss value of 6.98, as opposed to the results that came from our GANs.
- Iterating through the image from left to right, the Pixel CNN determines the value of each pixel based on its 24 neighbors located at the top-left corner. Consequently, during the gap-filling process the pixels at the bottom of the image remain unused. See figure 8.

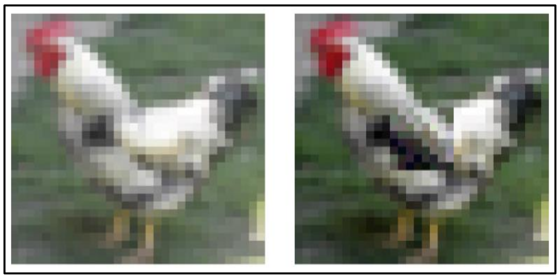

Figure 8: On the left is ground truth example and on the right is a reconstructed image from the Pixel CNN model

At the bottom and right portions of the image, clearly, the reconstruction of the bird reveals a neglect that is predominantly characterized with colors like light gray and white. This omission is particularly noticeable in the reconstructed square, where the discontinuity is more apparent at the bottom and right sides. Additionally, the limited number of dark feathers initially situated at the top-left corner of the absent section have now expanded to encompass a significant portion of the reconstructed square.

We also recognize the presence of the Row LSTM and Diagonal BiLSTM [25] from the same article. We successfully implemented and executed these methods, which yielded satisfactory results. While we refrain from providing an exhaustive report on these techniques due to their complexity in terms of implementation and performance, we attempted to propose a significantly simpler architecture. Despite its simplicity, this alternative architecture demonstrates the capability to deliver relatively good performance on CIFAR10, enabling insightful analyses.

Flattened Row LSTM:

Finally, we present a model that incorporates data from all preceding rows of the image to compute the probability distribution of each pixel. Termed as the Row Flattened LSTM, the architecture of this model can be seen in figure 9.

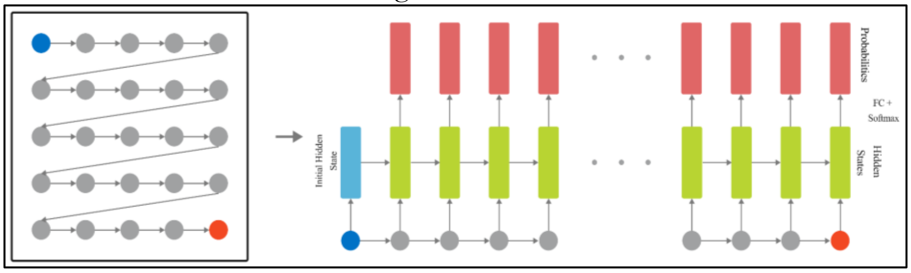

Figure 9: Row Flattened LSTM Architecture

The entire image is flattened, executing to the bottom-right from the top-left corner. Supplied to the LSTM are pixel channels, represented as 256-dimensional one-hot vectors, in the prescribed RGB sequence. With a set of 64 hidden dimensions, we proceed to convert these hidden vectors into output vectors from a fully connected softmax layer, each comprising 256 dimensions. Our Flattened Row LSTM reconstructed image example can be seen in figure 10.

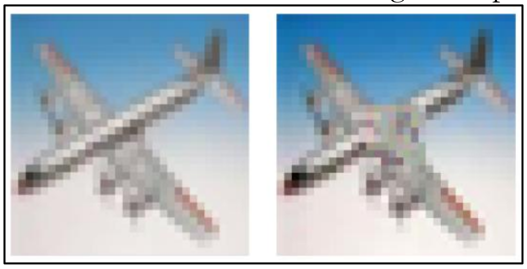

Figure 10: On the left is ground truth example and on the right is a reconstructed image from our Row-wise Flat Pixel LSTM

Our obtained result exhibits certain limitations compared to the Pixel CNN model. Several factors may contribute to these observations:

- The model assigns more influence to the pixels immediately on the left of the reconstructed pixel the way the LSTM scans. This differs from the Pixel CNN approach, where we employed a concentrated region of 24 neighboring pixels located at the top-left corner of our target pixel.
- As our current LSTM model may be too simplistic to effectively interpret this information, its strength lies in capturing all information from previous rows before making predictions. Enhancements, such as incorporating multiple LSTM networks or increasing the hidden dimensions, could potentially lead to more accurate predictions.

Results:

Autoencoders:

We used the Adam optimizer for training and conducted cross-validation for the learning rate. Learning rate decay proved beneficial across all our tests, striking a balance between swift initial learning and a gradual reduction in loss during later stages. However, we encountered challenges in cross-validating the decay rate and decay factor due to the intricacies involved in this process. Furthermore, we implemented dropout at a rate of 0.5 for every activation, and our experiments revealed no substantial changes in accuracy within the reasonable range of values for the probability drop from 40% to 80%. Also, we investigated the impact of the bottleneck size on our fundamental architecture, recognizing it as one of the most critical parameters. But it is worth noting that the best results, shown in table 1; did not emerge from the basic architecture.

Table 1: Train Vs. Val L2 Loss

Size of a Bottleneck	L2 Loss (Training)	L2 Loss (Validation)
256	7.02	7.96
512	6.36	7.41
1024	4.89	7.48
2048	4.23	8.14

It is apparent that overfitting poses a challenge. Significantly decreasing the size of bottleneck, and consequently it reduces the size of fully connected layer. This leads to a substantial decrease in the number of parameters, this is because this layer constitutes the majority of training parameters of the model. Overfitting diminishes notably from 2048 to 512, with 512 appearing to be the optimal choice, while 256 proves to be too small and potentially insufficiently expressive. Furthermore, to combat over-fitting more effectively, we plan to investigate strategies such as expanding the dataset and employing the data augmentation, integrating explicit L2 regularization, and experimenting with techniques like drop-connect, which involves randomly dropping connections in CNNs during training.

WGANs:

As previously mentioned, unlike our vanilla autoencoder, GANs do not exclusively optimize for $L2$ loss. Theoretically, $L2$ loss results should not be superior for GANs. However, we dedicated more time to refining GANs because our best $L2$ only architecture gave visually poor results; the optimized loss function and visual quality did not align. Consequently, our GANs exhibit improved results as a consequence of this experimental bias.

Due to challenges in training a Deep Convolutional GAN (DCGAN [26]), we exclusively present results for our implementation of WGAN, which has proven effective. The reported score in the paper was slightly inconsistent, representing a coefficient of 0 for the adversarial loss, making it a normal CNN. The most favorable outcomes were achieved with a VGG-like architecture. It is important to note that we do not argue that this is the optimal architecture, as our exploration of alternative options has been limited.

Density Methods:

Our conditional probability function in the Pixel CNN uses several convolutional layers with dropout. We opted not to include any pooling layers, as their addition did not appear to enhance the quality of our reconstructed images. Given that the parameters of the CNN are shared across all pixels, training the Pixel CNN essentially involves a classification task using a 24-pixel section; $5 \times 5 \times$ -1, as explained earlier, as input and an integer within the range $[0, 256]$ as the target. For optimization, we utilized the Adam Optimizer with cross-entropy loss, and the learning rate was cross-validated. The dropout rate was set at 50%.

Our Flattened Row LSTM model achieved a quadratic loss of 7.63 on the test set, employing the softmax cross-entropy loss. The hidden dimension 64 goes with cross-validation, although due to computational constraints, we could not test as many values as desired. With the incorporation of several LSTM networks, it is likely that we could have achieved a lower $L2$ loss.

Comparison:

The outcomes of the optimal run for each model type on the test set are depicted in table 2 below. Additionally, several outcomes from our top-performing run are illustrated in figure 11.

Table 2: Different models' L2 Loss values

Model	L2 Loss
Row Flattened LSTM CNN	7.63

Pixel CNN	6.98
Wasserstein GAN	4.26
CNN	7.49

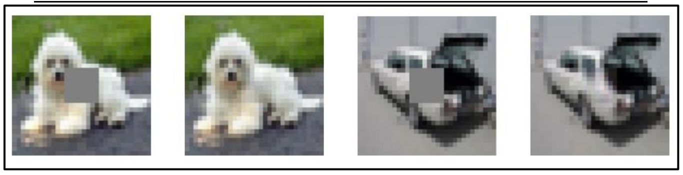
Figure 11: Results from Out-of-sample example

Conclusion:

Our unique CNN-based image in-painter with notable efficiency recognizes the significance of adversarial loss, as we incorporated a GAN into our model. Employing various tricks, we extended the approach from [13] to incorporate WGANs and utilized well-established architectures like VGG, Inception, and ResNet. Additionally, our overlap trick enhances border smoothing and also aids the training of discriminators. Our exploration not only delved into density-based methods, implementing Pixel CNNs based on [18], and introducing our model, Flattened Row LSTM. Through qualitative and quantitative comparisons, we looked to comprehend the limitations of these models. Overall, we are highly content with the outcomes achieved by our proposed architectures.

Future improvements involve adapting our models for larger images, offering exciting possibilities for handling more complex scenes with larger objects. Despite potential performance challenges, scaling up would open directions for addressing intricate scenarios. Additionally, refining our Flattened Row LSTM model to enhance symmetry and reinforce its generative capabilities stands as another goal for future improvement.

References:

[1] O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/S11263-015-0816-Y/FIGURES/16.

[2] T. Ružić and A. Pižurica, "Context-aware patch-based image inpainting using Markov random field modeling," IEEE Trans. Image Process., vol. 24, no. 1, pp. 444–456, Jan. 2015, doi: 10.1109/TIP.2014.2372479.

[3] K. H. Jin and J. C. Ye, "Annihilating Filter-Based Low-Rank Hankel Matrix Approach for Image Inpainting," IEEE Trans. Image Process., vol. 24, no. 11, pp. 3498–3511, Nov. 2015, doi: 10.1109/TIP.2015.2446943.

[4] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan, "Shift-Net: Image Inpainting via Deep Feature Rearrangement," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 11218 LNCS, pp. 3–19, Jan. 2018, doi: 10.1007/978-3-030-01264-9_1.

[5] C. S. Weerasekera, T. Dharmasiri, R. Garg, T. Drummond, and I. Reid, "Just-in-Time Reconstruction: Inpainting Sparse Maps using Single View Depth Predictors as Priors," Proc. - IEEE Int. Conf. Robot. Autom., pp. 4977–4984, May 2018, doi: 10.1109/ICRA.2018.8460549.

[6] J. Zhao, Z. Chen, L. Zhang, and X. Jin, "Unsupervised Learnable Sinogram Inpainting Network (SIN) for Limited Angle CT reconstruction," Nov. 2018, Accessed: May 06, 2024. [Online]. Available: https://arxiv.org/abs/1811.03911v1

[7] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009, [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[8] C. Cortes, M. Mohri, and A. Rostamizadeh, "L2 Regularization for Learning Kernels," Proc. 25th Conf. Uncertain. Artif. Intell. UAI 2009, pp. 109–116, May 2012, Accessed: May 06, 2024. [Online]. Available: https://arxiv.org/abs/1205.2653v1

[9] S. Gu, J. Bao, D. Chen, and F. Wen, "GIQA: Generated Image Quality Assessment," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 12356 LNCS, pp. 369–385, 2020, doi: 10.1007/978-3-030-58621-8_22.

[10] Y. Wang, Y. C. Chen, X. Tao, and J. Jia, "VCNet: A Robust Approach to Blind Image Inpainting," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 12370 LNCS, pp. 752–768, 2020, doi: 10.1007/978-3-030-58595-2_45.

[11] Y. Liu, J. Pan, and Z. Su, "Deep Blind Image Inpainting," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 11935 LNCS, pp. 128–141, 2019, doi: 10.1007/978-3-030-36189-1_11.

[12] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Scholkopf, "A machine learning approach for non-blind image deconvolution," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 1067–1074, 2013, doi: 10.1109/CVPR.2013.142.

[13] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context Encoders: Feature Learning by Inpainting," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-December, pp. 2536–2544, Dec. 2016, doi: 10.1109/CVPR.2016.278.

[14] I. J. Goodfellow et al., "Generative Adversarial Nets," Adv. Neural Inf. Process. Syst., vol. 27, 2014, Accessed: Oct. 02, 2023. [Online]. Available: http://www.github.com/goodfeli/adversarial

[15] B. Xu, N. Wang, H. Kong, T. Chen, and M. Li, "Empirical Evaluation of Rectified Activations in Convolution Network", Accessed: May 06, 2024. [Online]. Available: https://github.com/

[16] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., Sep. 2014, Accessed: May 06, 2024. [Online]. Available: https://arxiv.org/abs/1409.1556v6

[17] C. Szegedy et al., "Going deeper with convolutions," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 07-12-June-2015, pp. 1–9, Oct. 2015, doi: 10.1109/CVPR.2015.7298594.

[18] A. Van Den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional Image Generation with PixelCNN Decoders," Adv. Neural Inf. Process. Syst., pp. 4797–4805, Jun. 2016, Accessed: May 06, 2024. [Online]. Available: https://arxiv.org/abs/1606.05328v2

[19] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-January, pp. 4076–4084, Nov. 2017, doi: 10.1109/CVPR.2017.434.

[20] "Wasserstein generative adversarial networks | Proceedings of the 34th International Conference on Machine Learning - Volume 70." Accessed: May 06, 2024. [Online]. Available: https://dl.acm.org/doi/10.5555/3305381.3305404

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-December, pp. 770–778, Dec. 2016, doi: 10.1109/CVPR.2016.90.

[22] A. Birhane and V. U. Prabhu, "Large image datasets: A pyrrhic win for computer vision?," Proc. - 2021 IEEE Winter Conf. Appl. Comput. Vision, WACV 2021, pp. 1536–1546, Jun. 2020, doi: 10.1109/WACV48630.2021.00158.

[23] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," Adv. Neural Inf. Process. Syst., pp. 2234–

2242, Jun. 2016, Accessed: May 06, 2024. [Online]. Available: https://arxiv.org/abs/1606.03498v1

[24] "(PDF) The Earth Mover's Distance as a Metric for Image Retrieval." Accessed: May 06, 2024. [Online]. Available: https://www.researchgate.net/publication/220659330_The_Earth_Mover's_Distance_as_a_Metric_for_Image_Retrieval

[25] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel Recurrent Neural Networks." PMLR, pp. 1747–1756, Jun. 11, 2016. Accessed: May 06, 2024. [Online]. Available: https://proceedings.mlr.press/v48/oord16.html

[26] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," Int. Conf. Learn. Represent., 2015.