Islamic University of Technology (IUT)

Image to Image Translation with Multi-Scale Generator

Supervisor

Hasan Mahmud, Assistant Professor

Systems & Software Lab (SSL)

CSE, IUT

A thesis submitted in partial fulfilment of the requirements

for the degree of B. Sc. Engineering in Computer Science and Engineering

Academic Year: 2020-2021

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT)

A Subsidiary Organ of the Organization of Islamic Cooperation (OIC)

Dhaka, Bangladesh

March 28, 2021

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by the authors under the supervision of Hasan Mahmud, Assistant Professor, Department of CSE, IUT, Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Mashrur

Mashrur Mahmud Morshed

Student ID - 160041056

Tanvir

Hasan Tanvir Iqbal

Student ID - 1600410019

R. Shad

Mazharul Islam Rishad

Student ID - 160041053

Approved By:

Supervisor:

for, Md. Kamrul Hasan

Md. Kamrul Hasan, PhD
Professor
Dept. of Computer Science & Engineering.
Islamic University of Technology
Dhaka, Bangladesh

Hasan Mahmud, Assistant Professor

Systems and Software Lab (SSL)

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Abstract

Image to image translation is a highly generalized learning task, that can be applied to a wide number of Computer Vision application domains. Conditional Generative Adversarial Networks (cGANs) are used to perform image to image translation. The generator network typically used in the existing cGAN approach, Pix2Pix, adopts the U-Net architecture, consisting of encoding and decoding convolutional layers and skip-connections between layers of the same resolution. While effective and convenient, such an arrangement is also restrictive in some ways, as the feature reconstruction process in the decoder cannot utilize multi-scale features. In our work, we study a generator architecture where feature maps are propagated to the decoder from different resolution levels. We've experimentally shown improved performance on two different datasets — the NYU-V2 depth dataset and the Labels2Facades dataset.

Keywords: *Image to image translation, Computer Vision, Convolutional Neural Networks (CNN), Generative Adversarial Networks (GAN), Conditional Generative Adversarial Networks (cGAN)*

Contents

List of Figures

List of Tables

1 Introduction

When we think of the term *translation*, the most probable phenomenon to come to mind would perhaps be something pertaining to language. Even if two sentences from different languages are quite dissimilar to one another, they convey the same underlying *meaning* or *information*. And because of this, it is possible to translate between them.

Let us extend this simple yet powerful concept to an even more generalized context — it is possible to translate between any pair of information representations, given that they represent the same, or a similar, underlying meaning. Apart from language, images are also ubiquitous and effective conveyors of information. Needless to say, much like sentence to sentence language translation, it is possible to perform image to image translation (and even image to sentence and vice versa, for that matter).

Image to image translation is a highly useful and adaptable method. Many challenging and varied Computer Vision (CV) problems can in fact, be reduced down to image to image translations. Thus, many different problems can be reduced to a single, common solution method. For example, tasks like converting gray-scale to color (colorization), generating realistic photos/scenes from semantic maps, generating photos from sketches etc are all possible image to image translation applications.

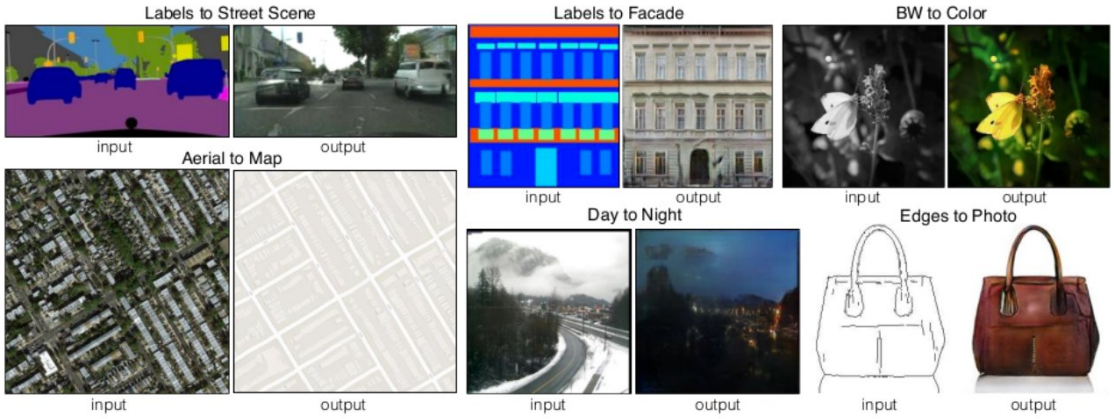

Figure 1: Examples of image to image translation, P. Isola et al. 2017 [2]

Like many other Computer Vision problems, image to image translation can be achieved with the aid of deep learning — to be specific, Generative Adversarial Networks [1]. Convolutional GANs excel at image synthesis. What is different in image to image translation is that artificial features are not simply being synthesized, but rather *reconstructed*. The GAN variant used here is more appropriately termed as a conditional GAN (cGAN) [16].

Rather than synthesizing a random artificial sample, image to image translation necessitates extracting useful features from the input, and then reconstructing the output using those extracted features. The explainability of the feature reconstruction process is an area of active study, and our work sheds further light on to this process.

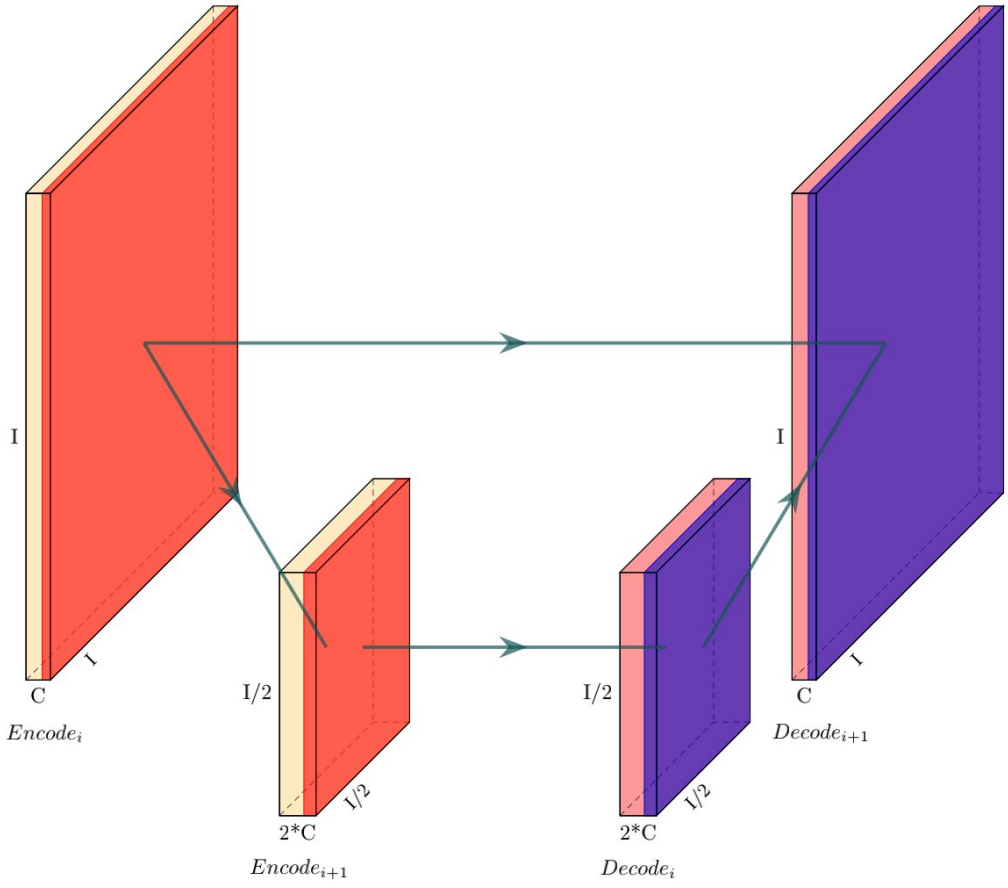

Figure 2: Same resolution skip connections in CNN encoder-decoder

The contributions of our work are as following:

1. We have shown that it is possible to fuse features from different resolutions in conditional GANs, and have provided a corresponding multi-scale Generator architecture.

2. We have empirically shown that using multi-scale features can potentially improve the performance of certain image-to-image translation tasks.

3. We provide deep insights into the feature reconstruction process in image-to-image translation, and the underlying role of skip connections and feature propagation in the Generator architecture.

1.1 Overview

The convolutional cGAN model used to perform image to image translation is termed as the Pix2Pix network. If we take a very high level glance at the workings of the Pix2Pix generator, we could break the process down into a few core processes, such as — a) a sequential convolutional *encoder*, which down-samples the input and extracts feature maps b) a decoder, which up-samples and reconstructs features, and c) skip-connections to propagate intermediate features from encoder to decoder.

If we were to investigate the intuition behind using skip connections, we would find that many image to image translation outputs have a lot of features in common with their inputs (and almost all features are retained in some tasks, like colorization in figure 1). So, intermediate feature maps extracted early on may be useful during the reconstruction process.

But these features are somewhat lost in the down-sampling pipeline. Thus, skip-connections are utilized to directly *propagate* these features to the decoders.

This concept is highly intuitive, and works well in practice. Figure 2 shows an example of such a skip connection. This is the primary basis of the existing Pix2Pix generator — performing same level skip connections to aid the decoder's feature reconstruction.

While same resolution or same scale connections are both intuitive and convenient in terms of implementation, they are also restrictive in a way. All the features extracted at different levels throughout the sequential pipeline of encoders can potentially contain useful information. An interesting research question arises — what if, the decoder could obtain useful information from feature maps of other resolution levels?

This is the primary objective of our work — to investigate the effect of propagating multi-scale feature maps to the reconstructing process. Rather than restrictively propagating same scale features, the generator model can potentially learn to use the most useful feature maps, from several possible resolution levels.

1.2 Problem Statement

The typical Generator used in image to image translation [2] follows the U-Net [3] architecture, and propagates features from encoder layers to decoder layers of the same resolution.

We wish to redesign the existing Generator model by fusing feature maps of different resolutions, thus allowing the model to exploit multi-scale features. We evaluate our approach against the traditional U-Net generator used in Pix2Pix, on several datasets and application domains.

1.3 Motivation and Scope of Research

Deep learning is a highly diverse field in and of itself, with many different sub-domains. One of the several reasons behind the rapid growth and success of deep learning is perhaps the exchange of knowledge and cooperation between sub-fields. Concepts which hold true in one domain often prove to further research in other, widely different domains.

For example, the attention mechanism [4] used in sequence modelling in Natural Language Processing (NLP) has been adopted in convolutional GANs, giving rise to the self-attention GAN (SAGAN) [5]. The encoder decoder architecture is also quite commonly used in both CV and NLP tasks. Again, encoder-decoders

with skip connections, proposed in the U-Net [3] model showed great success in segmentation tasks — and were later used in the image to image translation generator. As a matter of fact, image segmentation and image to image translation are highly cooperative fields.

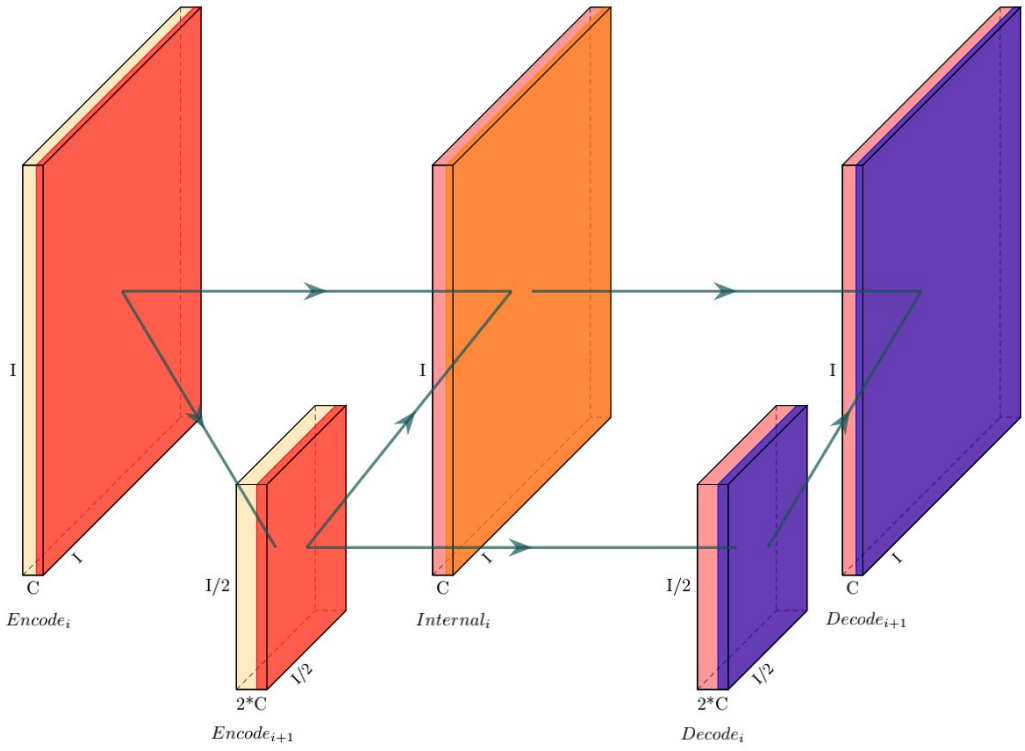

Figure 3: An example of a multi-level skip connection. An internal node aggregates information from current resolution level and the immediate prior level.

The usage of multi-scale feature propagation has been recently explored in the domain of image segmentation, and has shown improved results when compared with traditional skip connections. Thus, there is some scope for researching the utilization of similar processes in generative modelling.

While image to image translation is a highly popular GAN variant, most recent works in this field (and in the area of conditional GAN modelling, in general), have focused significantly on improving the loss functions and addressing the unsolved equilibrium problem inherent in GANs. The original Pix2Pix architecture is still widely followed, without much modification. As such, we believe that further

research into the base architecture of GAN generator and discriminator networks can be useful contributions to the overall field of study.

1.4 Research Challenges

There are several challenges we have to face in our avenue of research. They are as following:

1. GAN models are highly expensive in terms of resource requirement, because it involves the simultaneous optimization of two individual neural networks. The multi-scale generator involves considerably more convolutional layers than the traditional generator, and thus training deeper multi-scale generators would be limited by resource availability.

2. It is considerably more difficult finding optimal hyperparameters in GAN training when compared to more supervised deep learning methods. This is because the GAN loss is highly stochastic in nature.

   In supervised learning (for example, classification) a reduced loss value can be easily considered as a positive outcome. However, an increased GAN loss can be both a positive or negative outcome, because of the adversarial nature of the optimization. In practice, good hyperparameters can be found by repeatedly training the models to completion and making intuitive tweaks (the naive hyperparameter tuning process), which needs much time and effort and resources.

3. There is no standard, unified metric of evaluating an image to image translation GAN model. It can be applied to a wide variety of application domains, and each of those domains may be evaluated by different performance metrics (and sometimes there are no metrics at all, only human judgement).

   It is possible that a method can be good for one application domain but unsuited for another. Thus, the outcome of the research needs to be thoroughly evaluated from many different application perspectives, in order to judge its performance across various domains.

1.5 Structure of Thesis

In the introductory section, we have thus far provided an overview of our ideas and the intended objective of our research. We have also outlined the motivation and rationale behind our research, the scopes which our study encompasses, and the evident challenges throughout the overall process.

Next, we will go through a comprehensive literature review of topics related to our work, particularly on the nuances of Generative Adversarial Networks — knowledge of which is necessary and helpful for understanding our research topic.

The third section would relay the concrete outline of our proposed ideas on redesigning the existing Pix2Pix generator network. The fourth section, afterwards, will contain details and particulars of our experiments thus far, and the performance compared to existing methods.

In the final section, we intend to discuss the implications of our work as well as future directions of the research study, before concluding the article with the bibliography of all referenced materials.

2 Literature Review

In this section, we present briefs and detailed explanations, as necessary, of topics of notable relevance to our work.

2.1 Convolutional Neural Networks

It is apparent that much of our thesis work focuses around Deep Learning, particularly Deep Learning with image-based applications. Deep Learning has grown integral in the field of Computer Vision, and a lot of this growth can be attributed to the success of Convolutional Neural Networks (CNN).

Convolutions are local (neighborhood-based) operations which hold the important property of *translation invariance*. This allows convolutions to be immune to the shifting locations of patterns in inputs, which makes them excellent at extracting spatial information. As a matter of fact, images themselves are a spatial

function of two spatial axes.

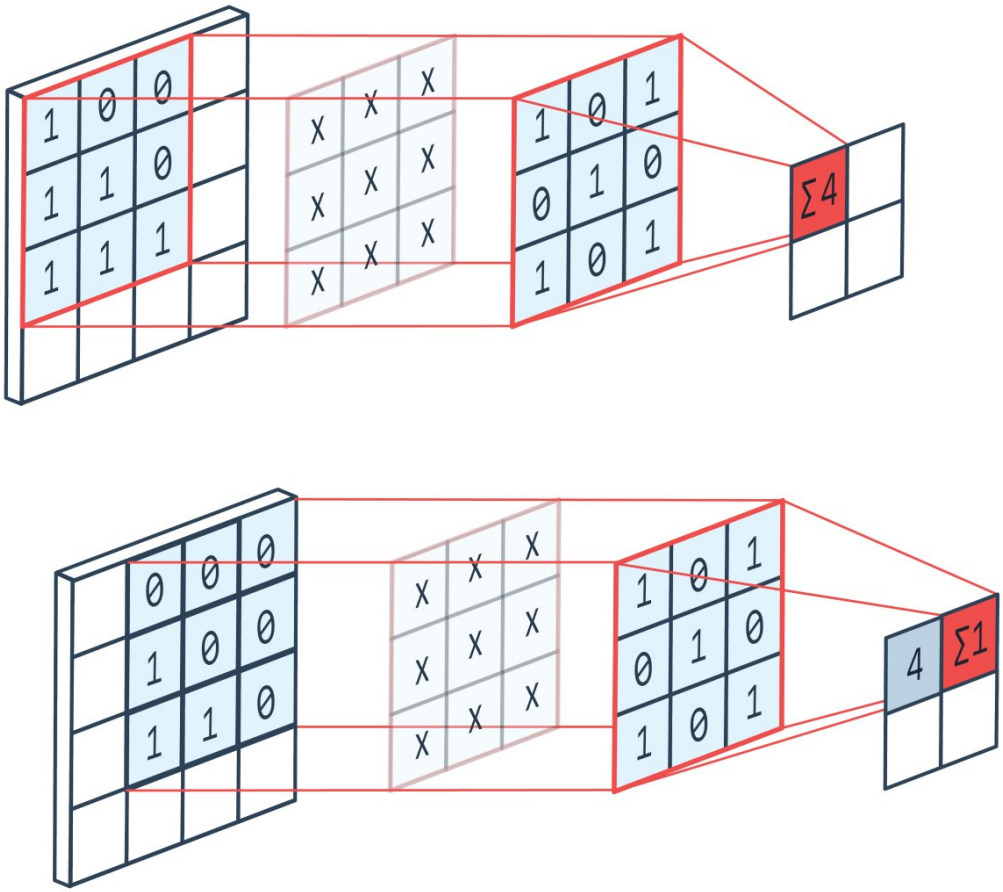

Figure 4: How 2D convolution works - A windowed, weighted sum

In practice, a convolution involves moving a sliding grid or a sliding window, usually termed as a kernel or a filter, across the input. Each element in the output is a weighted sum of a particular corresponding region in the input where the weight is actually the kernel grid, as can be seen from figure 4.

Traditional Computer Vision frequently uses convolutions with specific filters or kernels to perform a variety of tasks. For example, it is possible to extract the edges present in an image by convolving the Sobel [6] operator.

Convolutional Neural Networks work with the principle of differentiable convolutional kernels. As the kernels or filters have derivatives (and thus gradients), it

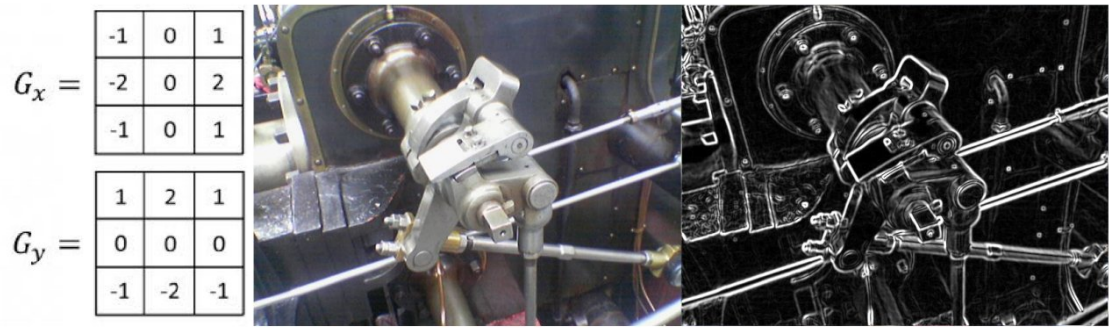

Figure 5: Sobel operator, input and corresponding edge output [7] [9]

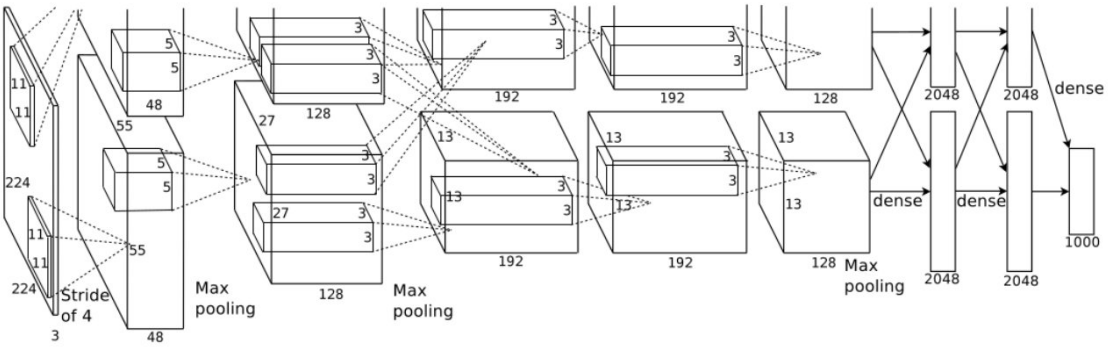

Figure 6: Alexnet architecture [10]

is possible to *learn* the ideal kernel for some arbitrary task, through the iterative optimization process that is characteristic of Deep Learning.

The popularization of CNNs can be attributed to the work of Krizhevsky et al. 2012 [10], in their work, *"Imagenet classification with deep convolutional neural networks"*. It was the first instance of a large scale CNN trained on a GPU device, that outperformed other contemporary approaches at that time, on the highly challenging ImageNet dataset [11].

CNNs have been used with great effect at tasks like image classification, object detection, localization, facial recognition, audio processing and representation learning. A more recent application of CNNs is in the sub-domain of Generative Adversarial Networks (GANs)[1].

Any GAN that works with images is also technically a CNN, because they employ convolutional and deconvolutional layers. Understanding the rationale behind the architecture of CNNs as well as the various layers that make it up is

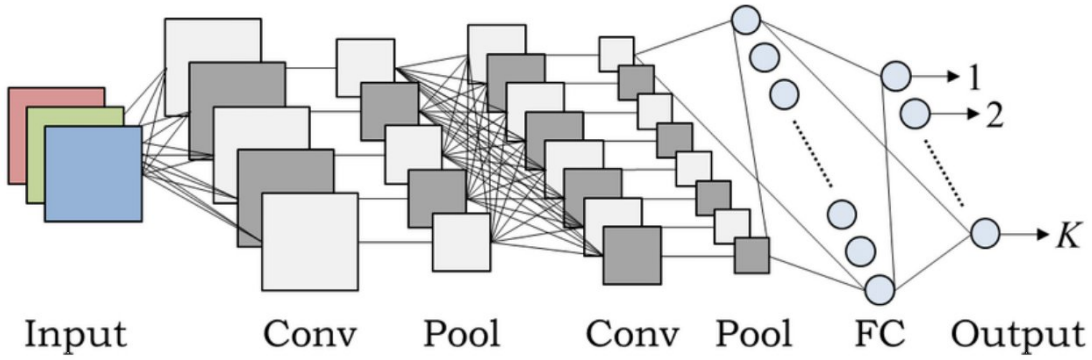

Figure 7: Example of CNN architecture(source)

crucial in designing successful GAN models. Fundamental insights like how max pooling destroys spatial features (which is why max pooling is frequently absent from many GAN architectures), or how batch normalization affects the covariate shift - these concepts are necessary in GAN development and can only be well understood by extensively studying Convolutional Neural Networks.

2.2 Receptive Fields

In order to understand some of the concepts later on (in Section 2.10.2), let us have a quick review of an important concept pertaining to convolutions — the receptive field.

Consider a simple 3 x 3 convolution. Such a convolution implies that each pixel in the output corresponds to an exact 3 x 3 region in the input. We can say that the *receptive field* of such a setup is 3 x 3.

Again, consider that there are two convolutions (3 x 3) in order. The output of the second convolution relies on a 3 x 3 region of the output of the first layer. Again, the first layer outputs rely on a 3 x 3 region of the original input. Effectively, this means that the final output relies on a 5 x 5 input region! So the receptive field of this setup, is 5 x 5.

As a matter of fact, one of the reasons behind creating deep convolutional neural networks is to increase the receptive field, such that the output corresponds to a sufficient amount of input region.

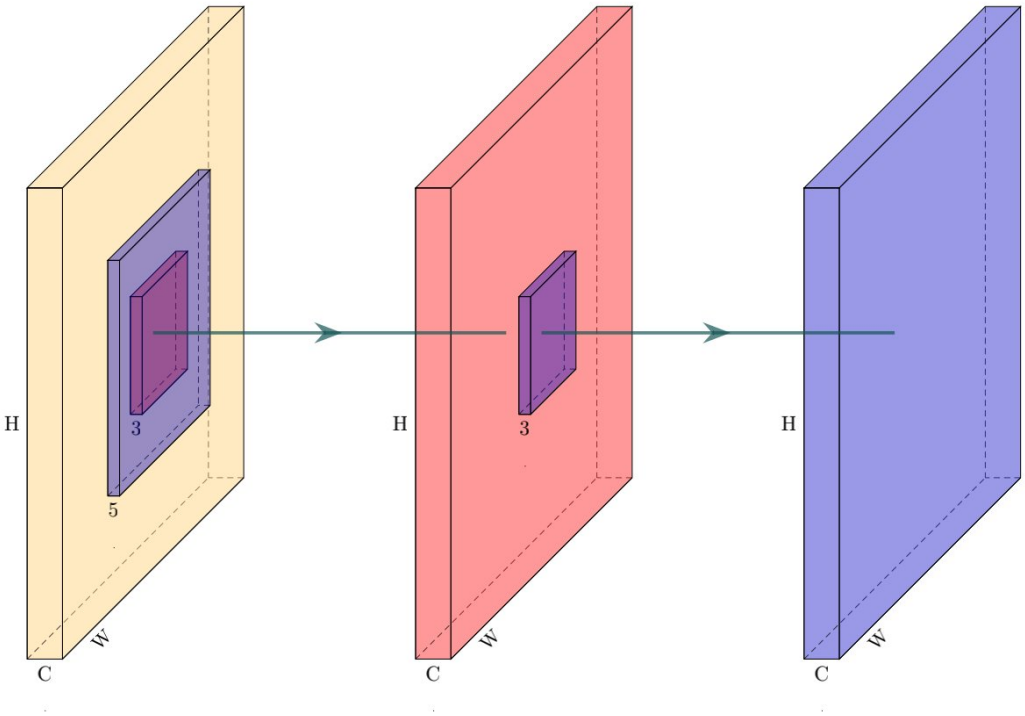

Figure 8: Two consecutive 3x3 convolutions results in a receptive field of 5x5

2.3 Pooling

Apart from convolutions, CNNs also typically contain other operations — most notable of which are pooling and batch normalization.

The supposed idea behind pooling is that not all pixels are important in a feature map of a large region. Pooling reduces the spatial resolution of large feature maps, and retains important features. The concept of importance however is presumptuous in many ways. For instance, the most commonly used variety of pooling is Maximum Pooling, which only retains the maximum valued pixel in the specified neighborhood size. Needless to say, there is not much justification behind the maximum valued pixel being the most important feature.

In practice, pooling does work very well for tasks where spatial features aren't

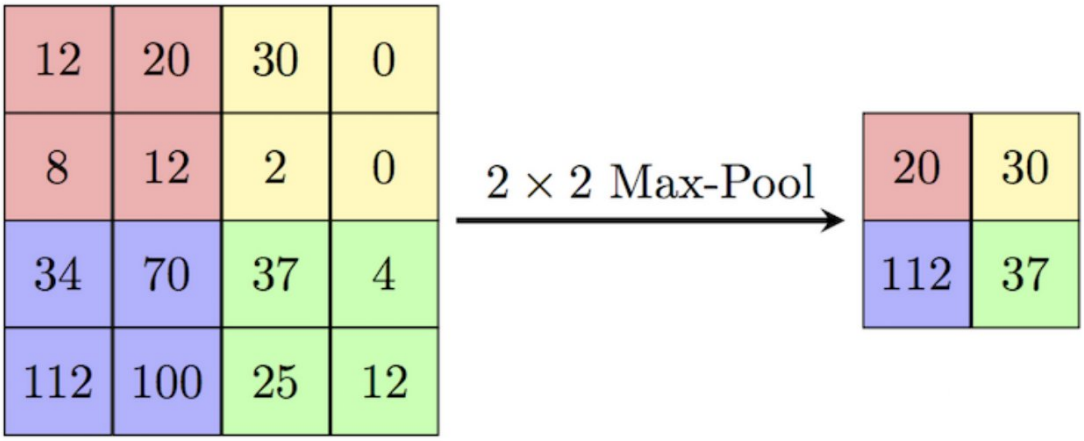

Figure 9: 2D Maximum-Pooling example (source)

as important as textural features — which is a common feature of image classification benchmark datasets. Pooling reduces the feature map resolutions, and also reduces the computational expensiveness of CNN architectures.

Pooling does not mix well with the field of generative modelling, however. GANs perform the task of reconstruction and synthesis, as opposed to the typical predictive classification. Pooling unwittingly destroys a lot of features which are useful during the reconstruction process, and have thus been replaced with strided convolutions in most popular GAN architectures.

2.4 Internal Covariate Shift and Batch Normalization

It is common practice in both machine and deep learning algorithms to normalize model inputs. Although it is theoretically possible to optimize with unnormalized data, in practice, this leads to a skewed cost objective which requires more iterations for convergence. Normalizing or standardizing the inputs makes the optimization objective easier to train on.

In deep neural networks, each network layer applies a complex non-linear transformation on the layer inputs. When the normalized inputs pass through several such layers, they lose their normalized property. This phenomenon is termed as *internal covariate shift* in relevant literature, and is considered as one of the factors which increases the training time.

Batch Normalization is a simple attempt to solve this problem. Each batch of data is again standardized batch-wise, before passing into the next layers. This significantly speeds up training time and is highly useful in many scenarios.

However, it remains to be said that batch normalization is less useful in other scenarios. For example, its effect is negligible when small batch sizes are used. Furthermore, studies in GAN modelling showed that Batch Normalization is often harmful to the GAN learning process. After all, the goal of the GAN is to learn the distribution of the training dataset — but batch normalization may affect that distribution, thus ending up as harmful in the long run.

2.5 U-Net

The U-Net is a very popular and highly useful architecture in the Deep Learning field. It was initially developed by Ronneberger et al. 2015 [3] for bio-medical image segmentation. Since then, U-Net has spawned a significant number of variants, and are the state of the art architectural paradigm in several sub-fields of Deep Learning, particularly segmentation and translation tasks.

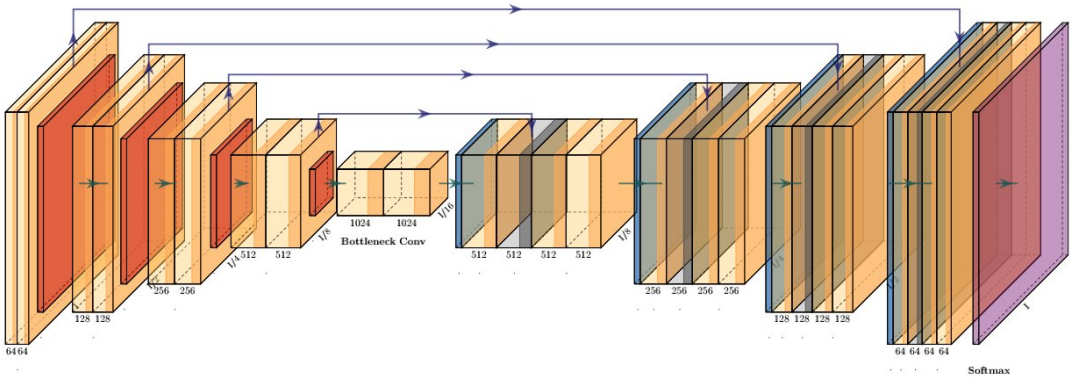

Figure 10: U-Net architecture [3]

Bio-medical image segmentation is different from the typical image classification task, in that it needs to determine no only the presence of the disease or affliction, but also the region or area of the anomaly. Rather than converging into

logits or class-confidence scores, segmentation requires pixel-wise predictions — whether a pixel belongs to the background or to some class.

The U-Net architecture is uniquely suitable for this task. Roughly, the network can be divided into two sub-networks — the encoder, and the decoder. The purpose of the encoder is in the familiarity zone of feature extraction, like typical CNNs used in image classification. An input image is reduced down to a simpler and rich representation vector.

In the original paper, the encoder consisted of three repetitions of blocks of two convolutions and one max-pool.

The purpose of the decoder is to use the condensed representation to perform a precise localization, using transposed convolutions (deconvolutions). Effectively, the learned representation is upsampled into a pixel-grid of identical shape to the original input.

The prime contribution of the U-Net is that it noted that the deconvolution layers are limited by the features supplied through the bottleneck. There are other useful features that can be helpful in localization present in the intermediate encoder layers. Thus, skip connecting those feature maps to the corresponding decoder layers allows those features to bypass the bottleneck, and significantly increases the segmentation performance.

U-Net is an end-to-end, fully convolutional network (FCN). There are only convolutional layers, and no dense (or fully-connected) layers. Because of the absense of dense layers, the U-Net can accept inputs of any arbitrary shape.

2.6 U-Net++

Over the years there have been lot of improvements of original U-Net [3] architecture. Similarly, a recent improvement is to redesign the skip connections of the original architecture. It is discussed in "Unet++: Redesigning skip connections to exploit multiscale features in image segmentation" [12]. In this paper, authors propose two different improvements of original architecture as U-Net+, U-Net++. Both consists of U-Nets of varying depths whose decoders are densely connected

at the same resolution via the redesigned skip connections.

U-Net's encoder-decoder architecture for image segmentation has two drawbacks. First, depending on the task difficulty and the amount of labeled data available for training, the optimal depth of an encoder-decoder network varies from one application to the next. A simple method would be to train models of varying depths separately and then combine the results during the analysis of inference time [22],[23],[24]. However, because these networks do not share a common encoder, this simple approach is inefficient in terms of deployment.

Furthermore, because these networks are self-taught, they do not benefit from multi-task learning. [25],[26]. Second, the design of encoder-decoder network skip connections is overly restrictive, requiring the fusion of same-scale encoder and decoder feature maps. While the same-scale feature maps from the decoder and encoder networks are striking as a natural design, they are semantically dissimilar, and no solid theory guarantees that they are the best match for feature fusion.

The architectural changes introduced in this paper [12] enable the following advantages.First, because it incorporates U-Nets of varying depths into its architecture, the new architecture is immune to network depth selection. All of these U-Nets share a portion of an encoder, and their decoders are linked. All of the constituent U-Nets are trained simultaneously while benefiting from a shared image representation by training the new architecture with deep supervision. Not only does this design improve overall segmentation performance, but it also allows for model pruning during inference time. Second, the new architecture is not hampered by unnecessarily restrictive skip connections that only allow the encoder and decoder to fuse feature maps of the same scale. The aggregation layer can decide how various feature maps carried along the skip connections should be fused with the decoder feature maps thanks to the new architecture's redesigned skip connections, which present feature maps of varying scales at a decoder node.

From Figure-7 we can see that the evolution is based on various use of skip connections between features of different scales. In U-Net+ different intermediate node are created using encoder and skip connection which in turn influences the

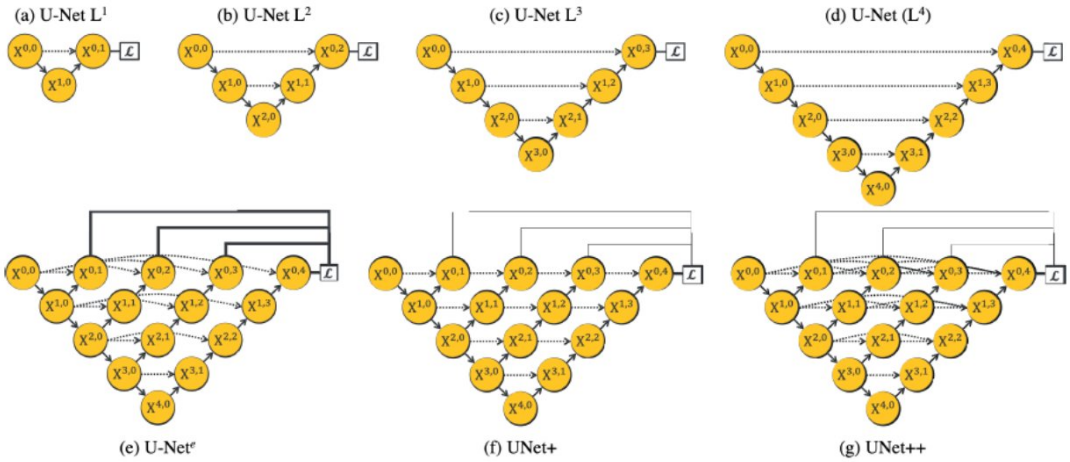

Figure 11: Evolution from U-Net to U-Net+ and U-Net++[12]

final outcome. This empirically shows better results as different scale features influences the final result. Similarly in U-Net++ an encoder or intermediate node gets skip connection from all of the previous decoder and intermediate nodes

2.7 Generative Adversarial Networks

Generative Adversarial Networks, or GANs[1], are a deep learning based generative model. More generally, GANs are a model architecture for training a generative model, and it is most common to use deep learning models in this architecture.The GAN architecture was first described in the 2014 paper by Ian Goodfellow, et al. titled "Generative Adversarial Networks."[1].

Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset. 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(in ideal cases), meaning the generator model is generating plausible examples.

In this way, the two models are competing against each other, they are adversarial in the game theory sense, and are playing a zero-sum game.

In this case, zero-sum means that when the discriminator successfully identifies real and fake samples, it is rewarded or no change is needed to the model parameters, whereas the generator is penalized with large updates to model parameters. Alternately, when the generator fools the discriminator, it is rewarded, or no change is needed to the model parameters, but the discriminator is penalized and its model parameters are updated.

At a limit, the generator generates perfect replicas from the input domain every time, and the discriminator cannot tell the difference and predicts "unsure" (e.g. 50% for real and fake) in every case. This is just an example of an idealized case; we do not need to get to this point to arrive at a useful generator model.

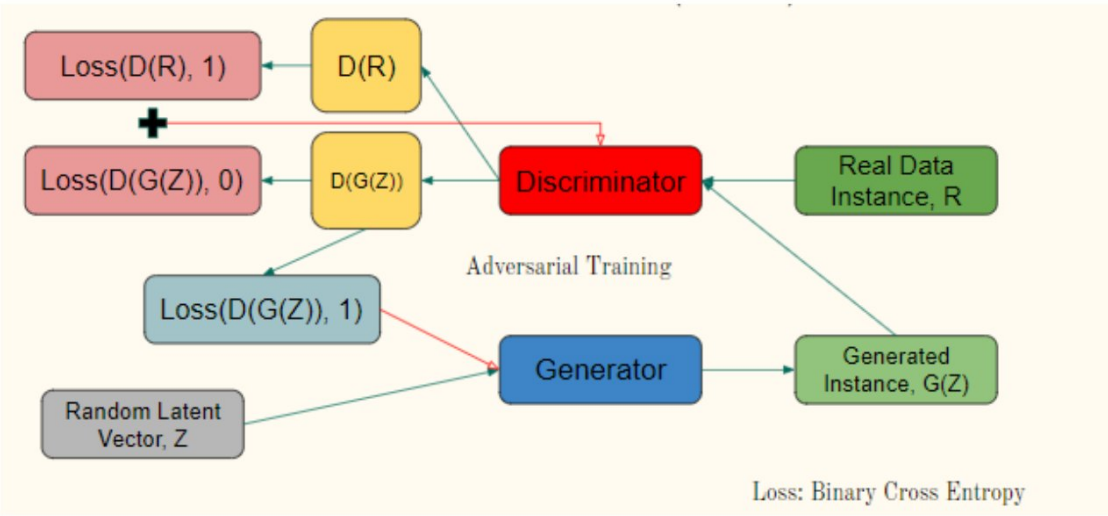

Figure 12: GAN architecture

2.7.1 GAN loss

The original GAN paper by Goodfellow et al. 2014 defined the losses to be as below:

$$Loss_D = \max \log D(X) + \log(1 - D(G(Z))) \tag{1}$$

$$Loss_G = \min \log(1 - D(G(Z))) \tag{2}$$

However, in practice it was found that the above formulation leads to saturated gradients for the generator. This led to a re-formulation of the original equation to its more useful form:

$$Loss_G = \max log(D(G(Z))) \tag{3}$$

These loss equations can actually tell us a lot about what the GAN model is attempting to do. By eq2, the discriminator is maximizing the log likelihood of real images and the inverse log likelihood for fake images, that is, predicting real images as real and fake images as fake.

In eq3, the generator is *minimizing* the probability of generated images being predicted as fake. However, in eq4, the generator is instead *maximizing* the probability of generated images being predicted as real! This subtle change solves the vanishing gradient problem, while keeping the training objective intact. This loss reformulation is a key progress of GAN optimization.

2.8 Conditional Generative Adversarial Networks

The conditional GAN[16], or cGAN, is a different type of GAN method. In classical GANs, the input to the generator is actually a random vector from a latent space, which results in a random generated image. In cGANs however, we provide some contextual information upon which the target output image is somewhat dependent. The generator in cGAN models can have either both a latent vector and some contextual information, or only the context information on its own. This additional information is leveraged by the generator to produce results with a specific context in mind.

The cGAN loss definition is not much different from the one used in classical GANs. If we observe the generator loss equation below:

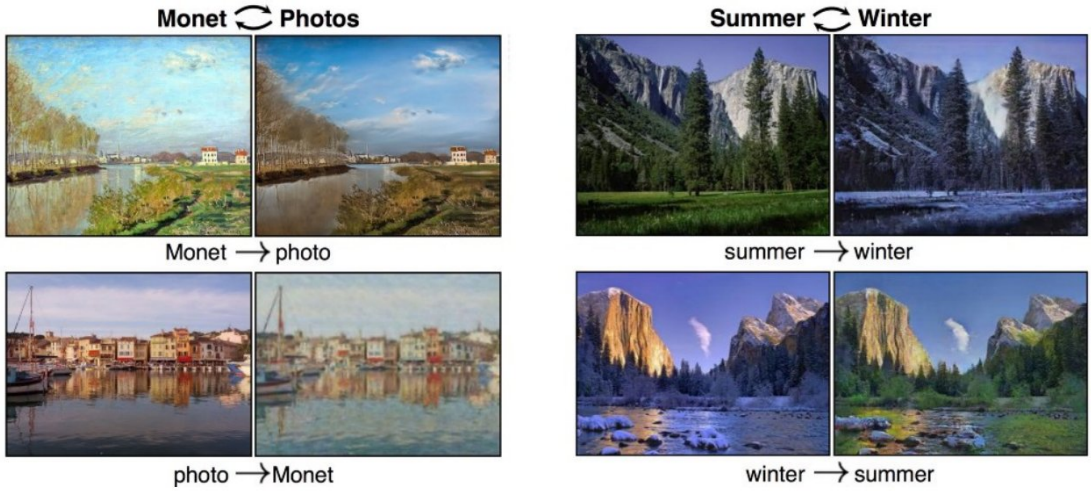

Figure 13: Unpaired Image to Image translation (CycleGAN)

$$Loss_G = \max log(D(G(Z|Y))) \tag{4}$$

This is the same as the regular GAN $Loss_G$, except that the generator response is now defined as $G(Z|Y)$. Here, Y is the *condition*; $Z|Y$ is thus a conditional input.

Pix2Pix and CycleGAN are popular examples of cGANs. The most important contribution of conditional GANs is that, the authors have shown a way to leverage supervision, even though GANs are an unsupervised process. Basically, cGANs take advantage of labels to reach a better optimum than it would be possible otherwise.

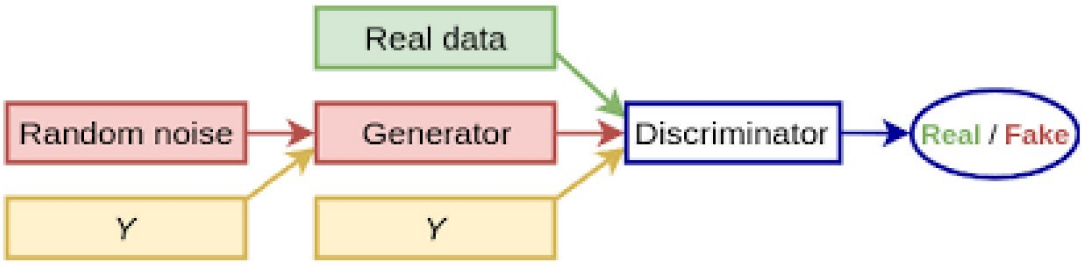

Figure 14: High level overview of CGAN architecture(source)

2.9 Deep Convolutional GANs

Deep Convolutional GAN or DCGAN[18] is one of the popular and successful network design for GAN. It mainly composes of convolution layers without max pooling or fully connected layers. It uses convolutional stride and transposed convolution for the downsampling and the upsampling. The figure below is the network design for the generator.

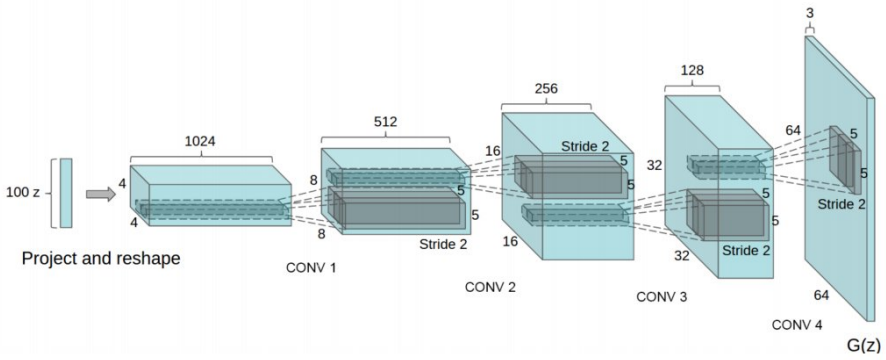

Figure 15: DCGAN Architecture [8]

DCGAN has following characteristics:

- Replace all max pooling with convolutional stride
- Use transposed convolution for upsampling.
- Eliminate fully connected layers.
- Use Batch normalization except the output layer for the generator and the input layer of the discriminator.
- Use ReLU in the generator except for the output which uses tanh.
- Use LeakyReLU in the discriminator.

The simplicity of DCGAN contributes to its success. We reach certain bottleneck that increasing the complexity of the generator does not necessarily improve the image quality. Until we identify the bottleneck and know how to train GANs more effective, DCGAN remains a good start point.

2.10 Image to Image Translation: Pix2Pix

Pix2Pix[2] network is basically a Conditional GANs (cGAN) that learn the mapping from an input image to an output image. This process is also called image-to-image translation.

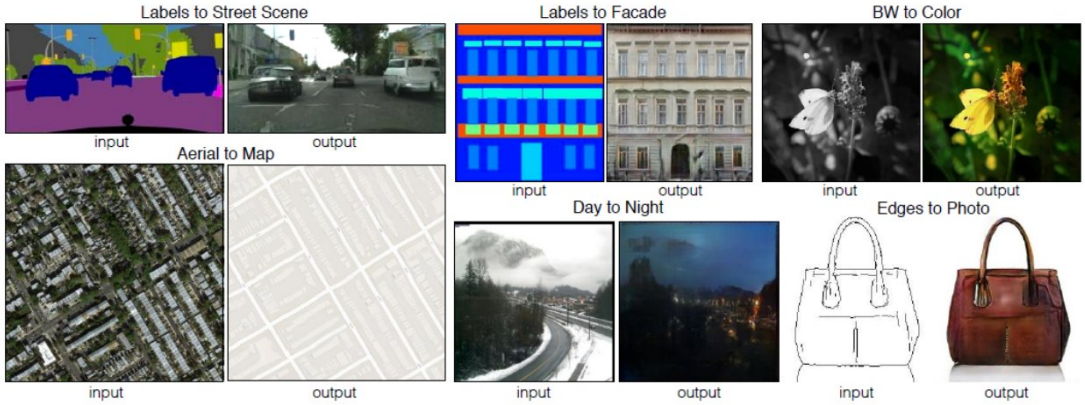

Figure 16: Image to image translation[2]

2.10.1 The Generator's Architecture

Generator network uses a U-Net-based architecture. U-Net's architecture is similar to an Auto-Encoder network except for one difference. Both U-Net and Auto-Encoder network has two networks The Encoder and the Decoder.

U-Net's network has skip connections between Encoder layers and Decoder layers. But auto-encoder does not have this kind of skip connection.

The generator network is made up of encoder(downsampler) and decoder(upsampler) network. There are six skip-connections in a Generator network. The concatenation happens along the channel axis.The Encoder network of the Generator network has seven convolutional blocks. Each convolutional block has a convolutional layer, followed a LeakyReLU activation function.Each convolutional block also has a batch normalization layer except the first convolutional layer. The Decoder network of the Generator network has seven upsampling convolutional blocks. Each upsampling convolutional block has an upsampling layer, followed by a convolutional layer, a batch normalization layer and a ReLU activation func-

tion.

2.10.2 The Discriminator's Architecture (PatchGAN)

The Discriminator model used in the Pix2Pix, also known as the PatchGAN[2], is also a fully convolutional neural network (FCN).

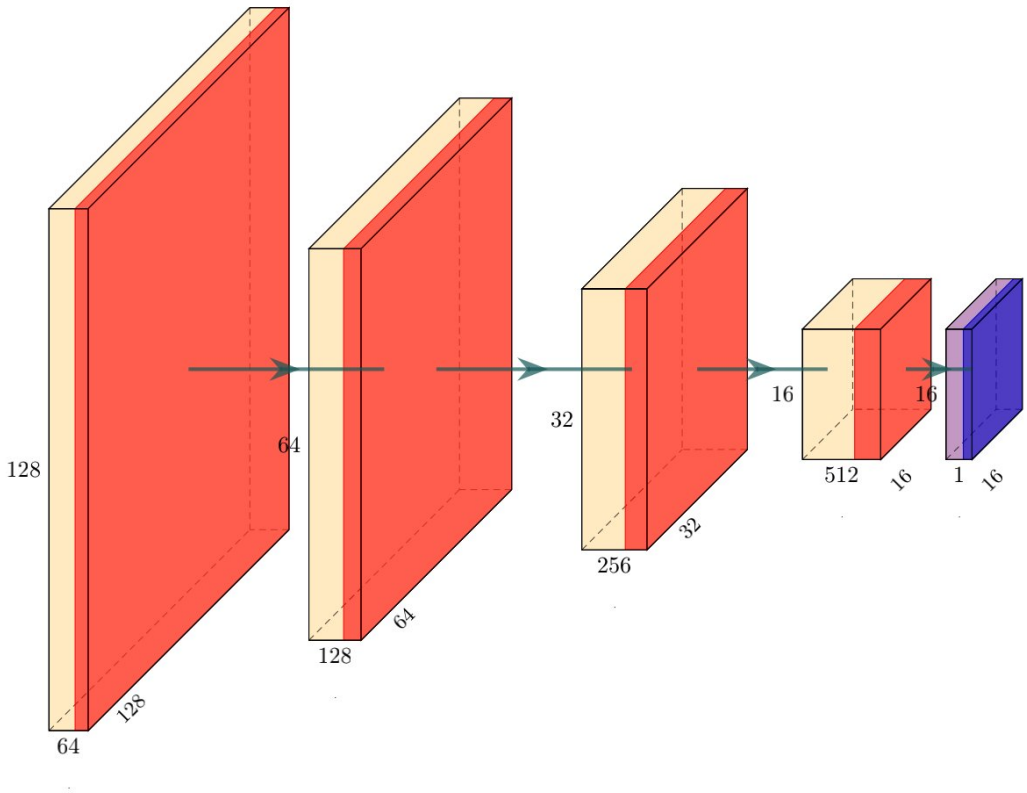

Figure 17: Pix2Pix Discriminator Architecture (PatchGAN)[27]

What is interesting about the PatchGAN is, that instead of making a single prediction, *Real* or *Fake*, it outputs an entire grid of values at the final layer, as can be seen from figure 17. This grid, or "patch" is the cause behind the name PatchGAN.

To understand the intuition behind the patch output, we need to understand the concept of receptive fields in convolutions. PatchGAN operates under the idea that it is not actually necessary to predict whether the entire image is real or fake.

Rather, it is sufficient to predict whether a part of the image is real or fake. Each pixel in the output grid corresponds to a specific partial region in the input image, and the size of this region is the effective receptive field of the CNN architecture.

Isola et al. empirically showed that a PatchGAN discriminator of 5 levels of depth (a patch output of 16 x 16) is enough to produce comparable performance to a patch size 1 x 1. To reach a 1 x 1 patch, the network needs to be even deeper, so the patch output saves a considerable amount of computation.

2.10.3 Pix2Pix Loss

The loss function for Pix2Pix is basically the cost function of conditional GAN, and is formulated as below:

$$L_{cGAN}(G, D) = E_{x,y}[logD(x, y) + E_{x,z}[1 - log(1 - D(x, G(x, z)))]] \quad (5)$$

The adversarial loss used is a simple binary cross entropy, applied element-wise per patch pixel. For example, if there is a 16 x 16 patch output from the discriminator, then after a sigmoid activation, that patch is converted into a grid in the range (0, 1), representing the probability of sub-regions being real. The grid is then measured up against grids of zeros and ones, using binary cross-entropy.

The generator loss isn't a pure adversarial loss. We also have to minimize the difference between the reconstructed image and the original image. To do so, we need a measure of error between two images. The L2 norm, or mean square error is a simple and straightforward approach to this dilemma, but L2 norm has the unwanted side effect of 'redistributing' inputs (L2 decay). In effect, this makes the model prefer generating blurred outputs to minimize the L2 loss.

Instead, to preserve the sharpness of the image, L1 norm is used in order to calculate a pixel-wise loss between the original and the generated image.

$$L_{L1}(G) = E_{x,y,z}[||y - G(x, z)||] \quad (6)$$

Thus the final loss function is:

$$G^* = argminmaxL_{cGAN}(G, D) + \lambda L_{L1}(G) \quad (7)$$

Here, λ is a constant of proportionality, to adjust the weight or importance of the L1 loss. Commonly, it takes values such as 10 or 100.

2.11 FID score

The Frechet Inception Distance (FID) is a metric which is used to evaluate the quality of the generated images by the generator in GAN. It is first introduced in paper "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium" [28]. The main goal of this evaluation metric is to compare the similarities between generated image and natural image. In this case the inception score estimates the quality of image by using an Inception V3 model for image classification of 1,000 known objects on those synthesized images.

The scores are made up of the integral of the marginal probability of the predicted classes and the confidence of the conditional class predictions for each synthetic image (quality and diversity). But the inception score does not compare synthesized image with real image, thus the goal for FID is to compare the statistics of a collection of synthetic images to the statistics of a collection of real images from the target domain to evaluate synthetic images. In FID score the model remains the same along with the coding layer that capture computer-vision-specific features of images. The activation calculated for both real and generated image then summed up as a multivariate Gaussian by calculating the mean and co-variance of the image.

Activations across the collection of real and generated images are then calculated using these statistics. The Frechet distance, also known as the Wasserstein-2 distance, is then used to calculate the distance between these two distributions.The score is called "Frechet Inception Distance" because it uses activations from the Inception v3 model to summarize each image. A lower FID indicates better image quality; a higher score indicates a lower image quality, and the relationship may be linear.When systematic distortions, such as the addition of random noise and blur, are applied, the authors of the paper show that lower FID scores correlate with better-quality images.

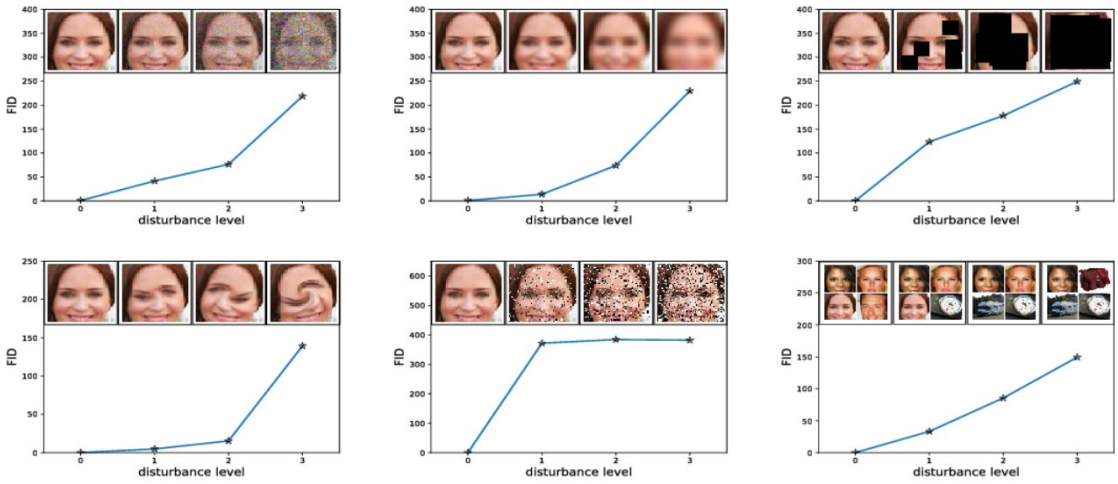

Figure 18: Increased image distortion is linked to a high FID score in this example.Taken from: GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium[28].

2.12 Differentiable Augmentation

One of the problems with GANs are that they are data hungry. If we use a small dataset, then the Discriminator can easily memorize the "real" data items and converge, thus ruining the adversarial balance. Whatever outputs the Generator will create, the Discriminator can judge accurately as fake – so generator gets no gradients.

Like other Deep Learning paradigms, researchers have attempted using augmentations to address these problems. However, applying augmentations directly on inputs have the effect of distorting the distribution of the data. The generator has to learn to mimic the distorted distribution. On the contrary, if the transformation is applied to the outputs of the generator as well as the corresponding real images, then generator doesn't have to "mimic the distortions"; it instead learns to create outputs in such a way that it would be affected by the distortions just how a regular, real image would. To the discriminator, it seems as though there is more data than there actually is, due to this transformation.

If the transformation is placed on the outputs of the Generator, then the output of this transformation is entering the Discriminator. This means the transforma-

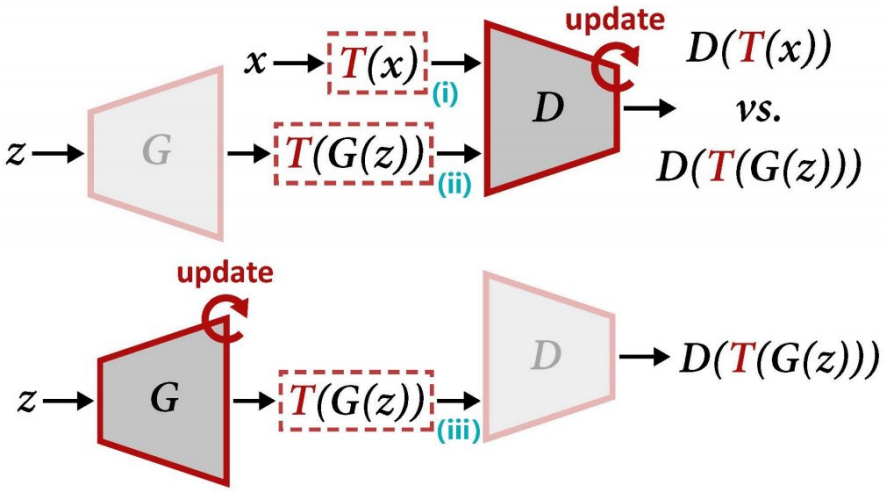

Figure 19: Differentiable Augmentation [19]; since the operation T is differentiable, this allows us to place it in between G and D (iii)

tion operation T must become a part of the computation graph and must allow gradients to pass through! That is, for any transform T(X), a valid operation dT(X)/dX must exist if that operation is to be a part of the computation graph. This is what we call a differentiable augmentation.

Figure 20: (left) StyleGAN (right) StyleGAN + DiffAug (source)

Thus far, DiffAug has been used with great effect to perform high quality image synthesis on standard GANs.

3 Proposed Approach

In this section, we describe our proposed experiments and changes which we wish to empirically evaluate.

3.1 Swish/Mish Non-Linearity

The ReLU activation overcame the initially popular Sigmoid activation when it came to training neural networks because it solved the problem of saturated or 'vanishing' gradients that came with the Sigmoid activation. Furthermore, ReLU is very fast to compute, which leads to both fast training and inference, a very attractive property when it comes to Deep Learning.

However, ReLU activation does not have any gradients for inputs less than equal zero. This may not always be ideal, as practically, gradients in the negative direction may also have some importance. Thus an alternative to ReLU is the LeakyReLU, which has a slope of α in the negative direction, rather than a slope of zero. LeakyReLU is often helpful in certain learning tasks.

Radford et al. 2015 [18] showed in their DCGAN paper that the downsampling layers in the discriminator benefit from having access to negative gradients. Thus, in the DCGAN architecture, the generator is comprised of ReLU activations, while the discriminator is comprised of LeakyReLU activations. This convention is followed in most image based convolutional GANs, Pix2Pix included.

In Pix2Pix, the generator has both downsampling and upsampling layers, owing to its encoder-decoder nature. The encoder portion of the Pix2Pix model contains LeakyReLU activations, while the decoder portion contains ReLU activation. Again, the PatchGAN discriminator also has LeakyReLU activations.

Recent works have introduced new activation functions, namely swish[13]/SiLU, and the later mish[14], which account for negative gradient and have been shown to be more effective than LeakyReLU and ReLU in a number of computer vision learning tasks. We wish to empirically determine whether replacing LeakyReLU with these activation functions would provide us with any improvement in the

convergence process.

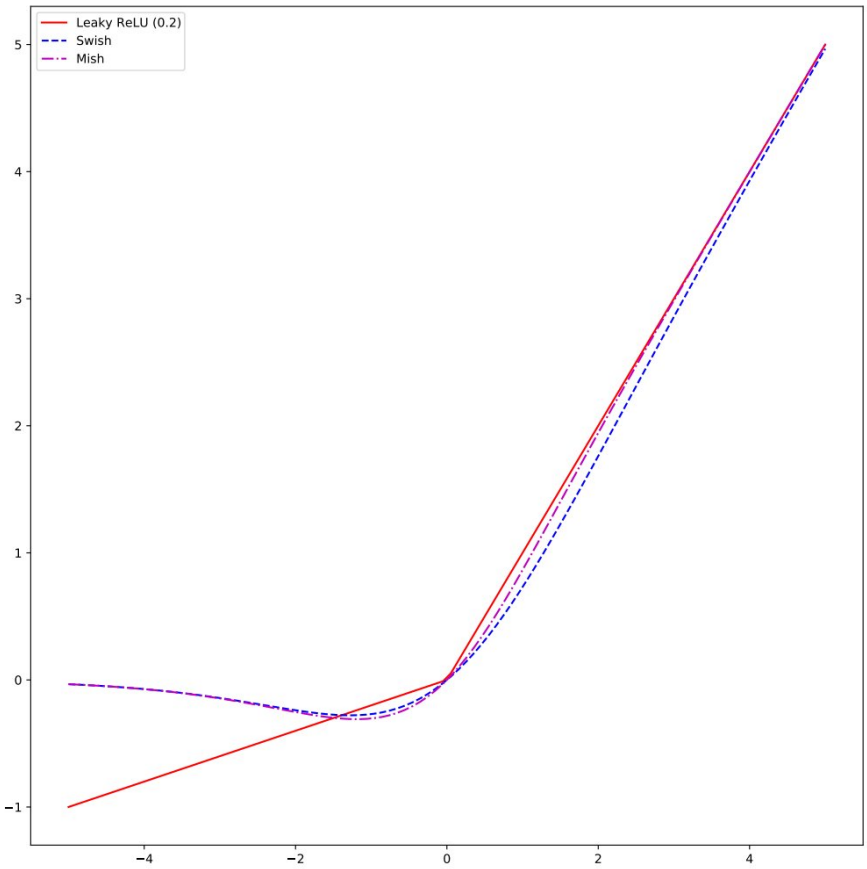

Figure 21: Swish, mish and leaky relu activations

3.2 Redesigning Skip Connections

If we observe the existing Pix2Pix or U-Net Encoder-Decoder architecture, we may note (from figure 22 (left)) that skip connections are present only among same scale features.

In the recently introduced U-Net++[12], which is used for medical segmentation, the authors argue that while concatenating features of the same scale seems natural, there is no theoretical guarantee that this is the best approach. Indeed, they later empirically show that, better segmentation results are obtainable by aggregating features from many scales. Note in the picture on the right, the final

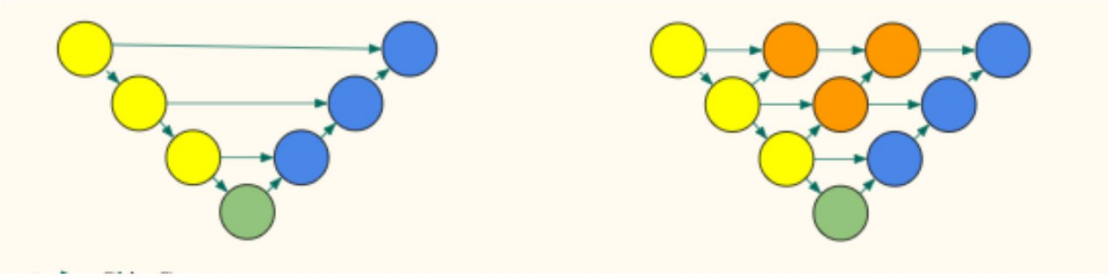

Figure 22: (left) Typical Skip Connections (right) Multi-Scale skip connections . Yellow and green nodes represent encoder and decoder nodes respectively. The green node represents the bottleneck. Orange nodes are the additional internal nodes needed for feature aggregation.

convolutional block in the decoder (color coded blue) receives aggregated features from every step of the downsampling operation in the encoder.

In original U-Net architecture, the whole process is divided into encoder and decoder process.In encoder process the input is repeatedly down sampled (using convolution) and then in decoder process it is repeatedly upsampled (using deconvolution) and skip conncetions (using concat) are added only on the same feature scale. We can see the details on Figure 23.

From the paper of redesigning skip connection [12] we use U-Net+ architecture. Detailed view of the architecture can be seen from Figure 24. From the figure, every downward blue line corresponds to convolution, side-wise blue lines corresponds to skip connections and upward red line corresponds to deconvolution. We can see the architecture has internal node that take skip connection from same scale and upsampling from lower scale and provides skip connection or upsampling accordingly. This not only provides better feature extraction from different scale input but also gives ensambled architecture. Architecture ensambling for U-Net has empirically shown better results on image segmentation. Thus we hypothesize that this improved U-Net architecture should perform as better generative model as it is more powerful in extracting and recognizing features.

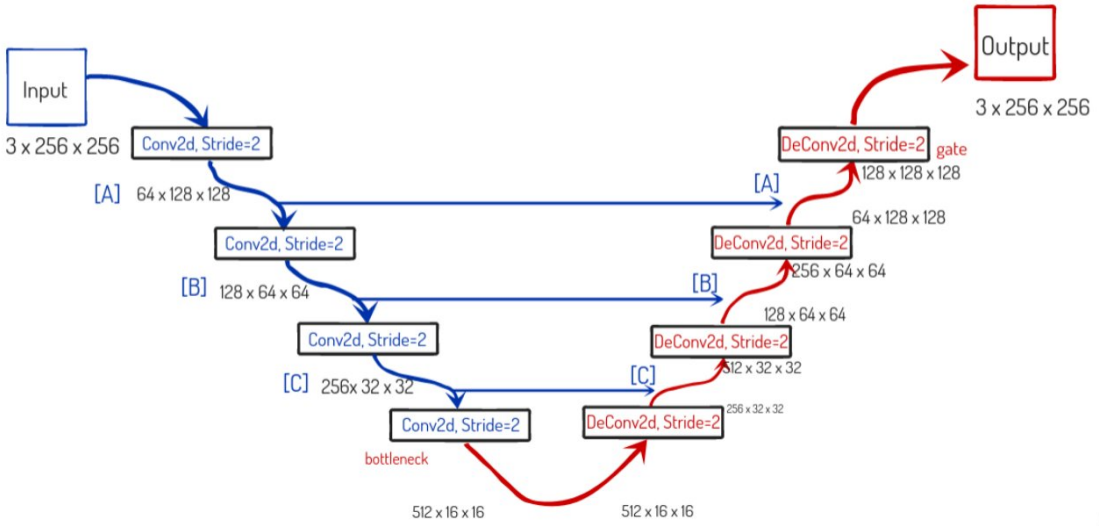

Figure 23: Typical U-Net Generator (Depth 4)

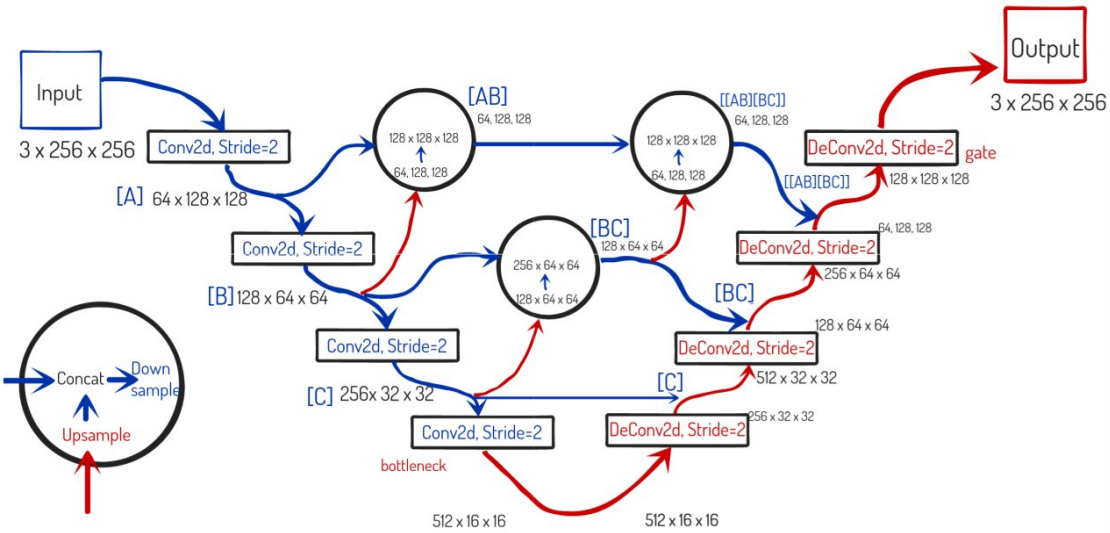

Figure 24: Multi-scale (U-Net+) Generator (Depth 4)

3.2.1 Architecture Details

Figures 23 and 24 show how the multi-scale generator is to be formulated. The overall skeleton of the original Pix2Pix generator is conserved, however, we add some internal nodes for feature aggregation and propagation.

The purpose of each internal node is fairly straightforward, and can be summed up in the pseudocode shown in algorithm 1.

Algorithm 1: Internal Node

Input(*Feature_A(C, H, W), Feature_B(2C, H/2, W/2)*);

Result: Aggregated_Feature, AB(C, H, W);

B_u := **ConvTranspose(C_{out}=2C, Kernel=4, Stride=2)**(B);

AB_conc := **Concatenate**(A, B_u);

AB := **Conv(C_{out}=C, Kernel=4, Stride=1)**(AB_conc); ;

Return AB

First, the feature map from the level below is upsampled with a deconvolution or a transposed convolution, which doubles the spatial resolution and reduces the channel dimension. The upsampled feature map is of identical dimensions to the skip connection feature map, and they are concatenated. This upsampling causes a *partial reconstruction* of the output. Effectively, rather than only the decoder deconstructing the output, the model is now doing partial reconstructions at multiple points in the generator architecture.

We now have an effect of an ensemble of reconstructions. The model reconstructs from richer feature maps, at the final reconstruction layer (gate).

One immediately noticeable concern and limitation is that the number of parameters will exponentially increase, with the increase in model depth. A multi-scale generator of five depth levels has approximately 54 million parameters, which is comparable to the classical Pix2Pix generator of a depth of eight levels. However, due to the partial reconstructions, the multi-scale generator can show a comparable and even improved performance at lower depths.

For all our experiments and evaluation, we use a depth of five levels.

3.3 Aligned Differentiable Augmentation

Thus far, DiffAug has commonly been applied to only standard GANs. We apply DiffAug to a conditional setting in our work.

Furthermore, for the task of paired image to image translation, it is necessary for input and output to be "aligned" in some manner. Currently, there are three types of differentiable augmentations available — differentiable random color jitter, random shift and random cutout.

We do not use random cutout for our experiments. Instead, we use aligned color jitter and aligned random shift. After random shift coordinates are calculated, the same shift is applied to both the real B and the generated fake B. We follow a similar approach in applying the same random color transform.

4 Experiments

We evaluated our approach on two datasets, which are widely different in content and application.

4.1 Datasets

4.1.1 NYU depth dataset

The NYU-V2 depth dataset is a dataset consisting of indoor scenes, for the prime objective of semantic segmentation. However, the nature of the dataset also coincidentally makes it highly suitable for depth estimation. The dataset consists of:

1. 1449 densely labeled pairs of aligned RGB and depth images
2. 407,024 unlabelled frames
3. Accelerometer data

4. Instance information

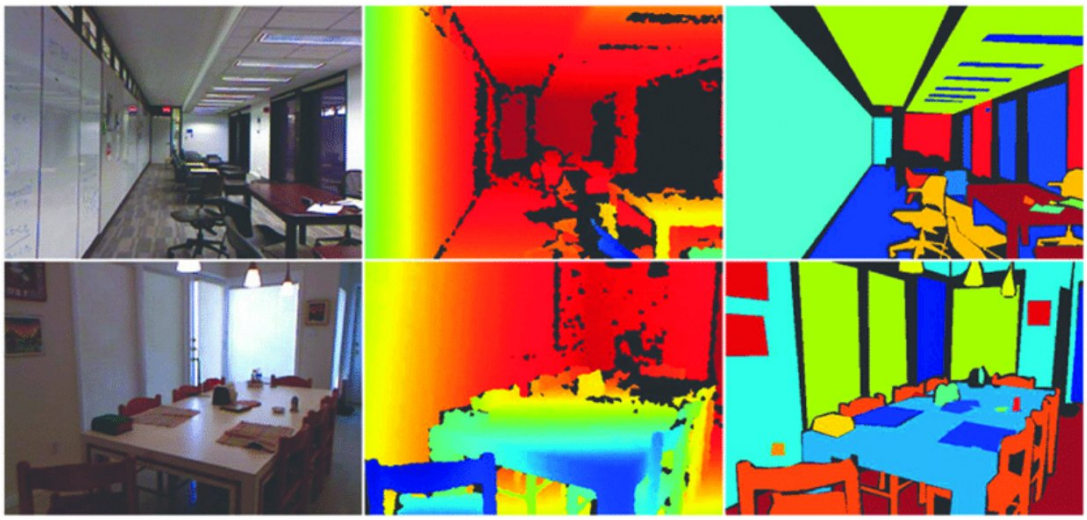

Figure 25: NYU-V2 depth dataset: (left) RGB (middle) Depth (right) Semantic Segmentation

4.1.2 CMP Labels2Facades Dataset

The facades dataset is a dataset containing images of the frontal faces, or "facades", of buildings. Each image is accompanied by a corresponding semantic map, which is essentially a very low level representation of the original image.

There are several challenges to the facades dataset, most notable of which is its small size. There are only 400 training examples and 100 test examples, which is a difficult scenario for the data-hungry GAN paradigm. Furthermore, the building images in facades contain a lot of small, fine details, which do not always correspond with the input semantic maps. This leads to the generation of noticeable artifacts.

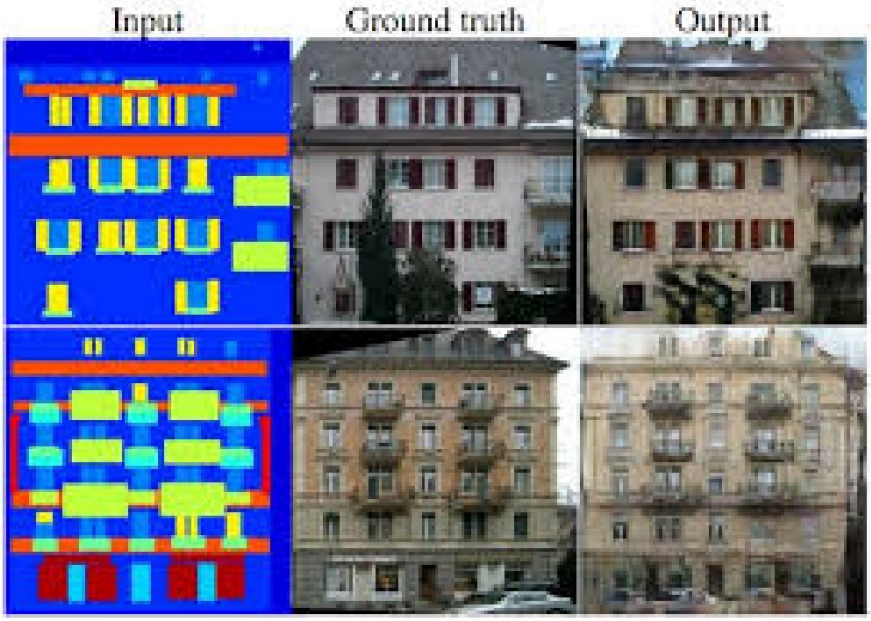

Figure 26: CMP facade dataset: (left)Input, (middle) ground truth, (right) Pix2Pix output

4.2 Comparison of Activation Functions

The effect of the choice of activation functions is seemingly minimal. We only performed the experiment of using different activation functions with the original Pix2Pix architecture, and not with our multi-scale generator. This is because our initial experiments did not show much difference, in our several trials.

Table 1: Effect of Activation Functions (NYU Depth V2, Pix2Pix), *Averaged over 3 runs

Activation	RMSE
LeakyReLU (Lore et al. [17])	0.875
LeakyReLU*	0.892
Swish*	0.872
Mish*	0.883

The results of using Swish and Mish do not show any significant difference. We believe our experiment is lacking in regards to proper evaluation of the effect

of activation functions — it is necessary to perform more trials, in order to obtain a robust measure of the standard deviation of each activation. Furthermore, the experiments should also be evaluated on multiple datasets instead of a single one.

Because the results were inconclusive and it is too expensive to train Pix2Pix models many times on many datasets, we decided to proceed with the original LeakyReLU activation for further experiments with the multi-scale generator.

4.3 Performance on NYU-Depth V2 and Label2Facades

We compared the results of our multi-scale generator model with the results reported in Lore et al. 2018 and by evaluating the performance of the pretrained model provided in the official Pix2Pix repository.

It is necessary to note that in the original Pix2Pix paper, the authors evaluated their model performance with human AMT evaluators (Amazon Mechanical Turks). Instead of human evaluation, we evaluate the generated images for Labels2Facades with the Frechet Inception Distance (FID) score, that is widely used to measure GAN image synthesis performance.

Table 2: Comparative results on NYU-Depth V2 and Labels2Facades; *Not official, trained by us

Method	Dataset	Metric	Score
Lore et al. [17]	NYU-Depth-V2	RMSE	0.875
Pix2Pix*	NYU-Depth-V2	RMSE	0.89
Multi-Scale Gen.+AlignedDiffAug(ours)	NYU-Depth-V2	RMSE	**0.796**
Multi-Scale Gen.(ours)	NYU-Depth-V2	RMSE	**0.811**
Pix2Pix	Labels2Facades	FID	160.6
Multi-Scale Gen. + AlignedDiffAug(ours)	Labels2Facades	FID	**148.9**
Multi-Scale Gen.(ours)	Labels2Facades	FID	**154.4**

From the comparative results, it can be seen that multi-scale generator has a definitive improvement in model performance. Furthermore, it can also be ob-

served that much like standard GANs, the usage of DiffAug in conditional GANs also has an effect in improving the performance. The effect is more notable in Labels2Facades, which is a significantly small dataset and thus has more need of augmentation and data efficiency.

4.4 Training Details

We trained all our models for 200 epochs, in order to follow the standard set in the original Pix2Pix paper and perform a proper comparative evaluation.

Instead of the identical initial learning rates showed in Pix2Pix, we find that a smaller generator learning rate performs better for the multi-scale generator. For the multi-scale generator and discriminator, we used learning rates of 0.00001 and 0.00002 respectively, with the Adam optimizer.

A multi-scale generator of depth 5 was used, which has 54 million trainable parameters, similar to the Pix2Pix of depth 7. We used a linear learning rate scheduling policy from the 100th epoch. All augmentations were aligned. Random horizontal flip and random resized crops were performed as typical augmentations, while we used a DiffAug of random shift in the NYU-Depth V2 and both random shift and random colorization in Labels2Facades. The inputs to the models used a resolution of 256 x 256.

4.5 Visualized Results

We show some results from NYU-Depth V2 and Labels to Facades respectively in this section.

From the results of the Facades dataset, the generated images are visually quite similar. Close inspection however shows an artefact in the top left corner of the original Pix2Pix output, which is absent in the multi-scale generator's output. We presume that differences like this lead to a lower FID score of the multi-scale generator, compared to the original Pix2Pix model.

Figure 27: Depth Estimation

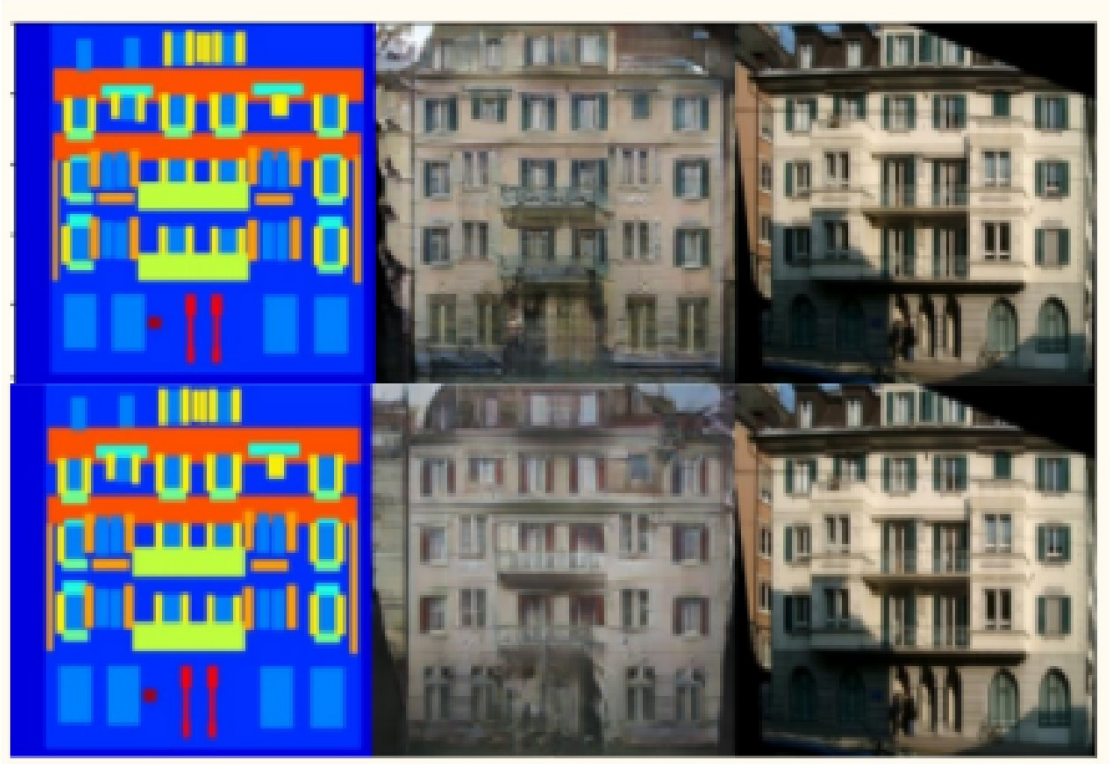

Figure 28: (above) Pix2Pix, Isola et al. 2017, (below) Multi-scale generator

5 Conclusion and Future Work

Thus far, we have performed image to image translation with a different generator architecture, and have provided some intuition and empirical results on how the choice of the generator architecture affects the resultant synthesized image. We have also attempted the usage of differentiable augmentation in a conditional GAN setting, with positive results as hypothesized in the original paper. We have evaluated our improved model on 2 different datasets and shown necessary comparisons on experimental result section.

Future extensions of our work consists of the following endeavours:

1. Evaluating the multi-scale generator on more diverse subsets of the image to image translation field, such as colorization, image inpainting, and so on.

2. Empirical evidence of the effect of non-linear activations

3. Training a deeper multi-scale generator. We plan on using DeepSpeed, a

recently introduced library which offloads parameter optimization on to the CPU, allowing larger models to be trained on a single GPU. DeepSpeed has thus far been highly useful in training models pertaining to natural language processing, and we believe it will be useful in training the parameter-heavy multi-scale generator

4. The expensiveness of the parameters in the multi-scale generator is one of its limitations. However, a multi-scale generator of lower depth than a typical generator shows comparable performance. We believe attempting GAN distillation can provide us with powerful multi-scale generator of low depth levels.

5. Other improvements have been proposed to the original Pix2Pix, in terms of loss functions and normalization. We wish to integrate our generator architecture with other proposed methods and re-evaluate the current state of the image to image translation field.

6 Bibliography

References

[1] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In Advances in neural information processing systems, pp. 2672-2680. 2014.

[2] Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-to-image translation with conditional adversarial networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1125-1134. 2017.

[3] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In International Conference on Medical image computing and computer-assisted intervention, pp. 234-241. Springer, Cham, 2015.

[4] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." arXiv preprint arXiv:1706.03762 (2017).

[5] Zhang, Han, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "Self-attention generative adversarial networks." In International conference on machine learning, pp. 7354-7363. PMLR, 2019.

[6] Sobel, Irwin, R. Duda, P. Hart, and John Wiley. "Sobel-Feldman Operator."

[7] By Simpsons contributor, CC BY-SA 3.0,[source]

[8] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[9] By Simpsons contributor, CC BY-SA 3.0, [source]

[10] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012): 1097-1105.

[11] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In 2009 IEEE conference on computer vision and pattern recognition, pp. 248-255. Ieee, 2009.

[12] Zhou, Zongwei, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. "Unet++: Redesigning skip connections to exploit multiscale features in image segmentation." IEEE transactions on medical imaging 39, no. 6 (2019): 1856-1867.

[13] Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for activation functions." arXiv preprint arXiv:1710.05941 (2017).

[14] Misra, Diganta. "Mish: A self regularized non-monotonic neural activation function." arXiv preprint arXiv:1908.08681 (2019).

[15] Jolicoeur-Martineau, Alexia. "The relativistic discriminator: a key element missing from standard GAN." arXiv preprint arXiv:1807.00734 (2018).

[16] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).

[17] Gwn Lore, Kin, Kishore Reddy, Michael Giering, and Edgar A. Bernal. "Generative adversarial networks for depth map estimation from RGB video." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1177-1185. 2018.

[18] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[19] Zhao, Shengyu, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. "Differentiable augmentation for data-efficient gan training." Advances in Neural Information Processing Systems 33 (2020).

[20] Lucic, Mario, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. "Are gans created equal? a large-scale study." Advances in neural information processing systems 31 (2018): 700-709.

[21] Bhat, Shariq Farooq, Ibraheem Alhashim, and Peter Wonka. "AdaBins: Depth Estimation using Adaptive Bins." arXiv preprint arXiv:2011.14141 (2020).

[22] T. G. Dietterich, "Ensemble methods in machine learning," in International workshop on multiple classifier systems. Springer, 2000, pp. 1–15.

[23] S. Hoo-Chang, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," IEEE transactions on medical imaging, vol. 35, no. 5, p. 1285, 2016.

[24] F. Ciompi, B. de Hoop, S. J. van Riel, K. Chung, E. T. Scholten, M. Oudkerk, P. A. de Jong, M. Prokop, and B. van Ginneken, "Automatic classification of pulmonary peri-fissural nodules in computed tomography using an ensemble of 2d views and a convolutional neural network out-of-the-box," Medical image analysis, vol. 26, no. 1, pp. 195–202, 2015.

[25] Y. Bengio et al., "Learning deep architectures for ai," Foundations and trends R in Machine Learning, vol. 2, no. 1, pp. 1–127, 2009.

[26] Y. Zhang and Q. Yang, "A survey on multi-task learning," arXiv preprint arXiv:1707.08114, 2017

[27] Liu, Yifan, Zengchang Qin, Tao Wan, and Zhenbo Luo. "Auto-painter: Cartoon image generation from sketch by using conditional Wasserstein generative adversarial networks." Neurocomputing 311 (2018): 78-87.

[28] Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." arXiv preprint arXiv:1706.08500 (2017).