Image Colorization Using Capsule Generative Adversarial Networks

Raggi Hosni Walid Hussein

FICS, The British University in Egypt

Abstract.
Automatic image colorization has increasingly become a heavily researched topic over the last decade. It has become of interest for many application areas including colorization of black & white movies, historical images, surveillance feeds and generally old image restoration. Recent state-of-the-art methods utilize Deep Convolutional Generative Adversarial Networks (DCGAN) for the colorization process. With the introduction of capsule networks, many flaws of the convolutional neural networks began to surface. In this project, the convolutional network layers inside the discriminator of a DCGAN will be replaced with capsule network layers. Further studies are employed to show how capsule networks as a discriminator in a DCGAN perform in the image colorization task.

1. Introduction

Image colorization has always been a highly discussed topic in the computer vision community. Many papers addressed this topic over the past decade and more recently Convolutional Neural Networks and Generative Adversarial Networks have been considered the state-of-the-art methods to approach this task. However, with the introduction of Capsule Neural Networks (CapsNets) [5] in 2017, many of the flaws of Convolutional Neural Networks began to surface and more research is being put towards CapsNets as it is thought to be the future state-of-the-art of image classification tasks.
In this project, Capsule network layers will be used instead of the standard convolutional layers as a discriminator inside a Deep Convolutional Generative Adversarial Network that will be used to colorize images.
Both methods will be used on Linnaeus 5 dataset [45] which provides a solid and diverse category that will be useful in the image colorization task.
Different qualitative measures will then be used to compare between the two approaches and evaluation measurers will be applied as well to further compare the two approaches for image saturation and general colorization.In addition to performing qualitative analysis on the CapsGAN model compared to the traditional GAN model. Several parametric studies are conducted on the CapsGAN model and quantitative analysis are done on those studies to optimize the performance of the model.

2. Background

2.1. Generative Adversarial Networks

In 2014 Generative Adversarial Networks were introduced by Goodfellow et al [1]. Since then, this network has attracted a lot of attention and has been used in various computer vision and image generation tasks.
The GAN consists of two smaller networks namely the generator and the discriminator**.** The idea behind it is that the generator consistently generates images and these images need to be close to the real one. The discriminator tries to classify whether an input image is from the generator or is the real image. Both of these networks are trained in parallel until the generator can consistently ‘fool’ the discriminator network by producing results it can’t classify correctly.

The generator part of the network can be represented by the mapping G(z; θG), where z is a random noise input variable for the generator. By contrast, the discriminator can be represented by the

mapping D(x; θD), where x is a colored image and the output of it is a scalar between 0 and 1 being the probability that x is a real or fake image.

This relation can be described as an optimization problem where the generator is trained to minimize the probability that the discriminator makes a correct prediction and the discriminator is trained to maximize the probability of making the correct guess

This can be expressed mathematically as:

$$\boldsymbol{min_{\theta G}\, J^{(G)}(\theta_D, \theta_G) = min_{\theta G}\, E_z\left[\log\left(1 - D\left(G(z)\right)\right)\right],}$$

$$\boldsymbol{max_{\theta D}\, J^{(D)}(\theta_D, \theta_G) = max_{\theta G}\, (\, E_x[\log(D(x))] + E_z\left[log\left(1 - D\left(G(z)\right)\right)\right]}$$

These two equations can be combined in a single cost function as a minimax game with a value function V(G,D):

$$\boldsymbol{min(G)\, max(D)\, V(G, D) = E_z[log\, D(x)] + E_z[log\left(1 - D\left(G(z)\right)\right)]}$$

Generative Adversarial Networks have been used for applications like image-to-image translation, style transfer, edge mapping...etc. Different versions of GANs currently exist and they are used across many different applications. One of the most successful and widely used GAN architectural guidelines is the Deep Convolutional GANs which was proposed by Radford et al [6]. It introduced various techniques and architectural guidelines to improve the difficult learning of traditional GANs and addressed common issues with the traditional GAN[1]. Isola et al. [2] proposed an architecture adapted from the DCGAN[9] architecture that produced general image-to-image translations that could train on various image-to-image translations and produce stellar results. Such translations examples are aerial to map, edge to photo, black & white to color…etc.

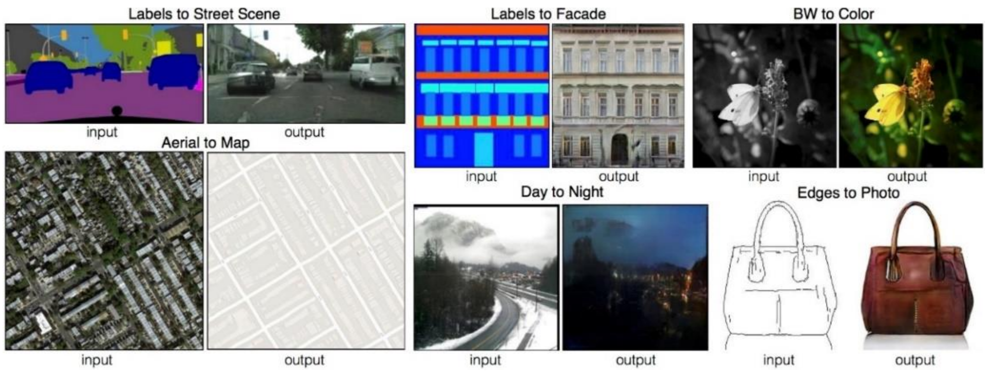


Figure 1 Photo from pix2pix by Isola et al [8], image to image translations with GANs

K Nazeri et al [3] extended work already done by Isola et al [2] for image-to-image translations to make it more general on the colorization problem, especially on higher resolution images as well as introducing strategies to make training faster and more stable.

2.3. Capsule Networks

The very basic idea of capsule networks was first introduced by Geoffrey E.Hinton et al.[4] and was finally implemented in 2017 [5].

Unlike CNNs, capsule networks provide positional equivariance of the image features instead of the positional invariance of CNNs. To give an example on that, a CNN is really good at extracting and detecting features of an image. However, it fails significantly in detecting the spatial relationships between these features. Consider a face with eyes, a nose and a mouth but in incorrect positions or orientation, the CNN would still recognize it as a face since all the features of a face exist in the image, although it is not necessarily a face due to different spatial positions of the features.

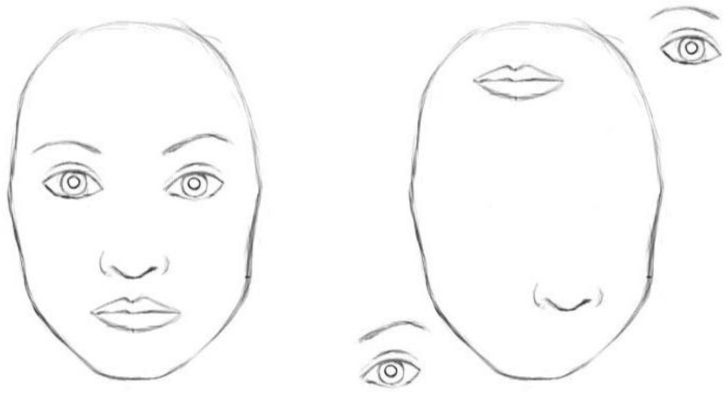


Figure 2 Demonstration of positional Invariance of CNNs and Equivariance of Capsule Networks. CNNs will classify both images as a face. However, a Capsule Network will detect that the second image is not a face due to positional equivariance

Capsule networks work by separating neurons in several groups that are referred to as "capsules", Those groups of capsules capture both the probability that a feature exists in the image along with its positional information. Unlike traditional neural networks, where a single scalar output is used to describe the activities of the features captured, capsule layers compress these outputs into a small output vector. In the training process, the probability that a feature exists is invariant to the local view of the capsule. With the changing position of the features, the capsule value changes as well in a corresponding manner as they represent the positional information of that feature.

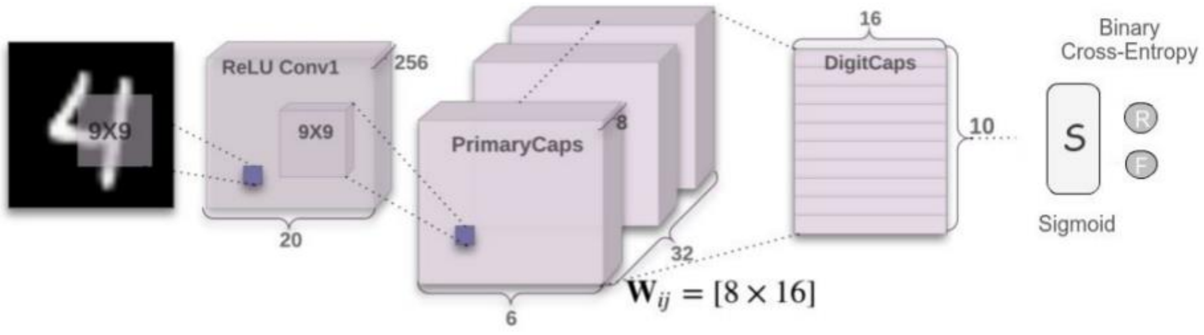


Figure 3 Capsule Network architecture employed by Hinton et al. [5]

The positional information of the features detected by the capsules is encoded as to be in the direction by which the capsule output vector is pointing to. When the feature position in the image is changed, the output vector length does not change. However, its orientation is changed. Going back to the previous example, if a nose feature is detected on the image of a face, suppose the capsule will output a vector of length 0.7, when the nose moves from its original position to a different position, the output vector will rotate correspondingly to represent the change in the nose state/position. However, the length of the output vector stays the same, thus achieving positional equivariance.

By achieving equivariance, it is feasible to interpret the different possible variations of a feature in an image without changing the feature detection probability. Dynamic routing by agreement is done by having higher level capsules receiving the output vectors of the lower level capsules beneath them. Lengths of the received output vectors $\boldsymbol{U_i}$ determine the probability of features being detected in the lower level capsules. The vectors after that are multiplied by weight matrices W that will encode the positional information and spatial relationships between the features of the higher-level capsules and the lower level capsules. The output of that multiplication is the positional probability of the features in the higher-level capsules.

$$\hat{u}_{j|i} = \boldsymbol{W_{ij} u_i}$$

Prediction vectors weighted sum

2.4. Capsule Generative Adversarial Networks

Ayush Jaiswal et al. [8] as well as Raeid Saqur et al. [10] Proposed the idea of using capsule networks instead of the typical convolutional neural networks as discriminators within Deep Convolutional Generative Adversarial Networks (DCGAN) for image synthesis tasks. The presented model outperformed the convolutional based GAN on the MNIST and the CIFAR-10 datasets. Both [8] and [10] employed similar architecture highlighted below

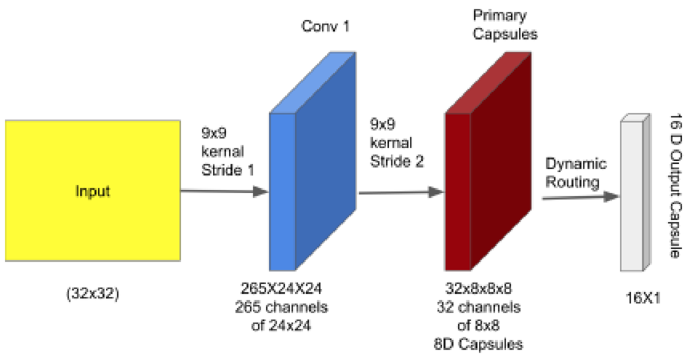


Figure 4 Capsule Network as Discriminator employed by [8] and [10]

Capsule Generative Adversarial Networks have never been used before in image colorization tasks which will be the aim of this project. They have only been used in a very limited number of image classification/generation tasks due to its very recent introduction.

3. Materials & Method

3.1. Dataset

The used dataset in this project is Linnaeus 5 [13]. This dataset is categorized into 5 different classes, (Dogs, Berries, Flowers, Birds, Miscellaneous).

Each class consists of 1200 training and 400 testing images for a total of 6000 training and 2000 testing images.

This data set has a diverse enough category to test the colorization, especially general colorization since it contains colorful pictures represented in flowers and berries as well as landscape images such as skies and green areas as well as having a miscellaneous category that covers daily life pictures such as offices/outdoors/rooms…etc.

Table 1 Linnaeus 5 Categories

Berry	Dog	Flower	Bird	Miscellaneous

3.2. Method

3.2.1. Generative Adversarial Networks

K Nazeri et al. [3] extended the work of [2] and used an architecture for the generator similar to U-net [7] following DCGAN guidelines [6]. The idea behind their network was based on encoder-decorder networks where down-sampling operations are performed on the input having a contractive path (skip layers). The encoding process is then reversed using decoding layers until the input is reconstructed. The main benefit of this approach is the efficiency in using GPU memory mainly due to the skip connections.

Their generator architecture consists of n encoding layers and n decoding layers. Each layer of them is comprised of a 4x4 convolutional layer with a stride of 2 for down-sampling/up-sampling followed by batch normalization and Leaky-ReLU activation with a slope of 0.2. The encoder-decoder are concatenated with the activation map of the mirroring layer. The last layer is a 1x1 convolution which acts as a cross channel pooling layer with a tanh activation.

A similar Generator to the one employed by K.Nazeri et al will be used in the project.

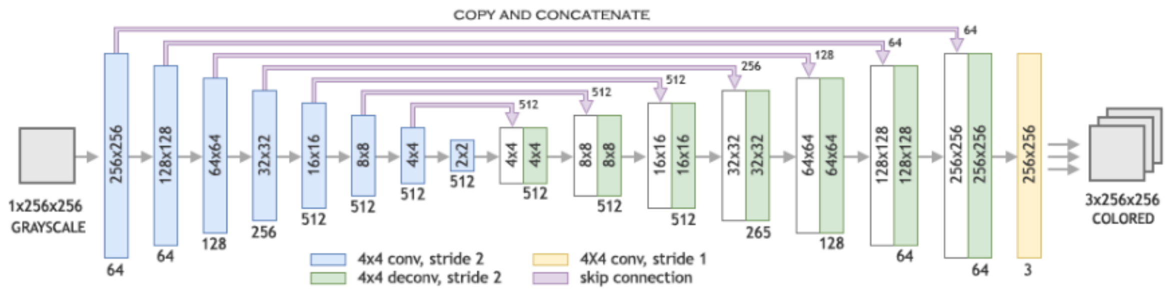


Figure 5 Generator employed by K.Nazeri

An alternative cost function has also been employed in their generator. Originally, in Equation 1 the cost function is mapped by minimizing the probability that the discriminator is correct. This approach has two main problems.

1. If the discriminator is performing well during training, the generator would have a gradient that is near zero during back-propagation which in return will slow down the convergence significantly since the generator would produce similar results.

2. The cost function is asymptotically decreasing without a lower bound, this in return will cause it to diverge to $-\infty$ during minimization.

Therefore, instead of minimizing the probability of the discriminator being correct, the probability that the discriminator is being mistaken is maximized. This cost function was introduced by Goodfellow at NIPS 2016 Tutorial [11] and is mapped as:

$$\boldsymbol{max_{\theta G}\, J^{(G)*}(\theta_D, \theta_G) = max_{\theta G}\, E_z\left[\log\left(D(G(z))\right)\right]}$$

Introduced GAN Cost function by Goodfellow

Which can also be mapped as a minimization problem:

$$\boldsymbol{min_{\theta G} - J^{(G)*}(\theta_D, \theta_G) = min_{\theta G} - E_z\left[\log\left(D(G(z))\right)\right]}$$

mapped as a minimization problem

For the Discriminator, K.Nazeri used a convolutional based discriminator based on DCGAN which is aimed to be replaced in this project.

For their discriminator, a convolutional architecture is used to encode the image and then pass through a sigmoid activation to classify whether the input image is real or fake

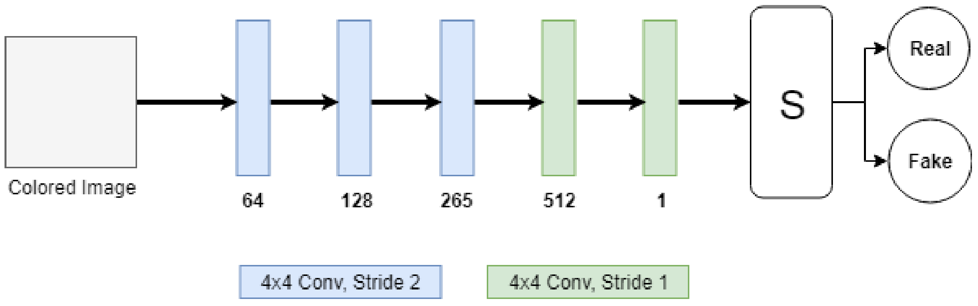


Figure 6 Convolutional Discriminator that will be changed with a Capsule discriminator

The main idea of the project is to replace this traditional convolutional discriminator with a capsule based one.

3.2.2. Capsule Generative Adversarial Network

Generator

The generator used will be similar to the one mentioned earlier by K.Nazeri

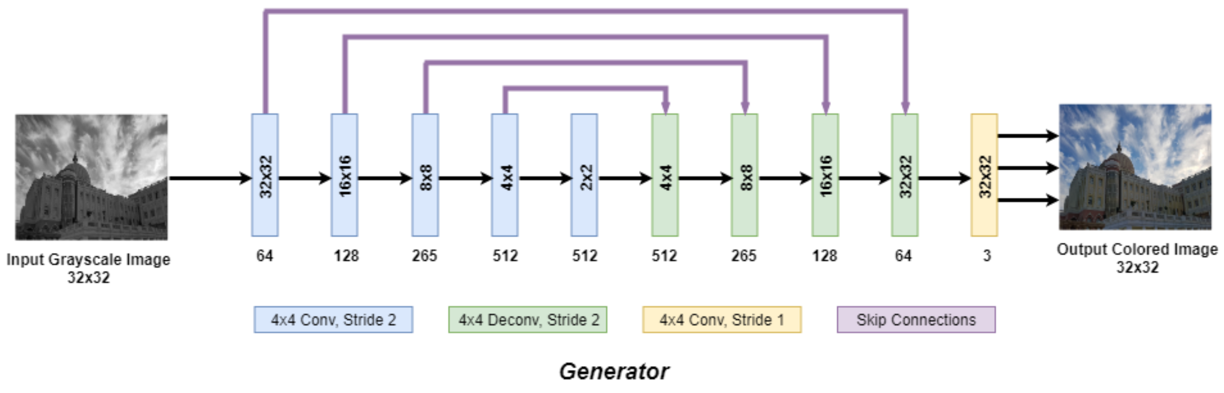


Figure 7 Generator similar to the one employed by K.Nazeri based on U-net.

The generator is trained to maximize the pixel-wise accuracy of the colors between the pixels (p) of the original image (x) and the generated one (y) over all color channels (c). By calculating the ratio between pixels that have similar color and the total number of the pixels in the image. Two pixels are considered to have the same color if the distance between their color channels in the LAB color-space is within a set threshold distance (e) (2%)

$$\frac{1}{n}\sum_{p=1}^{n}\prod_{c=1}^{3}\mathbf{1}_{[0,e_c]}\left(\left|h(x)^{(p,c)}-y^{(p,c)}\right|\right)$$

Pixel-wise Accuracy

Discriminator.

The fact that capsule networks can be used as classifiers as pointed out by Hinton makes it a perfect choice to use as a discriminator in our DCGAN. It will be used as a cross-binary entropy classifier to classify whether an input image created by the generator is a real one or a fake one.

Images fed into the discriminator are 32x32 colored images either coming from the exact value of the image, or with the generated “fake” colored image (done by concatenating the generator output (ab channels) with the L channel of the image), for images above 32x32, they are first down-sampled by simple stride convolutions down to 32x32 and then fed in.

The first layer is comprised of a convolutional layer with 256 filters, a 9x9 kernel and a stride of 1. This layer has a large filter size which allows it in return to increase the receptive fields of the following layers [5]. In Dynamic routing by capsules [5] this layer extracts low level features from the image to feed it later in the CapsNet.

After the first layer, a Leaky-ReLU activation is then used to optimize the performance [6] as well as a Batchnorm layer, both of which are not used in the original paper CapsNet paper [5],

Leaky-ReLU is used as the activation function to serve two main purposes, the first one is to introduce non-linearity, the second being the avoidance of the vanishing gradient problem as mentioned in [12]. The Batch normalization is used to increase the performance of the discriminator as discussed later.

The Capsule network architecture starts off after the first layer.

PrimaryCaps

The first component of the capsule network architecture is the Primary Capsules layers (PrimaryCaps), This layer consists of three main components, the first one is a convolution with 256 filters which maps to an 8D vector that represents 32 feature maps each, this convolution is followed by a reshape function that is applied to split the output into the mapped 8D vectors. After that, the last component which is a squashing function [5] is used.

$$V_j = \frac{||S||^2}{1+\ ||\ S\ ||^2}\frac{S_j}{||\ S_j||}$$

Equation 1 Squashing Function

The squashing function acts as a non-linearity function just like activation functions in convolutional networks such as Leaky-ReLU. The difference is that traditional activation functions work on best on single neurons and in the capsule network case we are dealing with output vectors instead of single neurons. The squashing function squashes the output vector length to 0 if it is of a small length and generally, it tries to limit its squashed length to 1 if its of a long length. Upon applying the squashing function to a capsule group, all the output vectors of individual capsules in the group are squashed.

DigitCaps

After the PrimaryCaps, DigitCaps follows, this layer is the part of the capsule network that implements the dynamic routing by agreement [5], Firstly, since the Dense functions only receive flattened values, the output of the PrimaryCaps is flattened rather than being vectorized like the original paper. The result of the flatten function will enter three layers of DigitCaps (DigitCaps1, DigitCaps2, DigitCaps3). In each of those, a softmax function is used first and then the output is fed in a dense layer that would act as a prediction vector consisting of 160 units (10 capsules * 16 vector units), a Leaky-Relu activation is finally applied after each DigitCaps layer (originally the paper uses a squashing function which is replaced here by Leaky-Relu).

Finally, a dense layer with a unit neuron is applied with a sigmoid function that will act as a binary classifier to determine whether the image is real or fake.

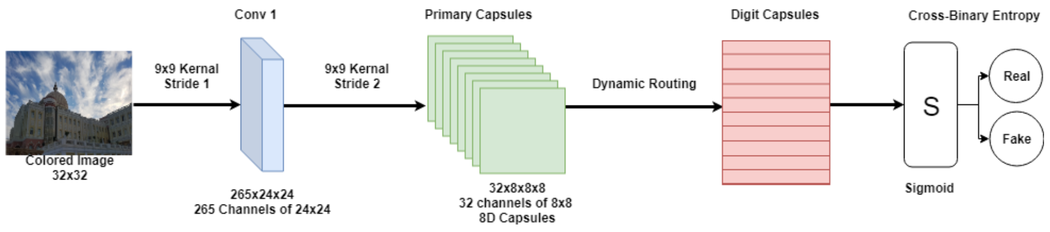


Figure 8 Capsule Discriminator

4. Experimental Results

Different image error and quality metrics has been used over the testing data after 200 epochs using GAN, CapsGAN with Batch normalization and without it such as Mean Absolute Error, Root Mean Squared Error, Peak Signal-to-Noise Ratio as well as Pixel-wise accuracy which measures

4.1 Image Error and Quality Metrics

Table 2 Results comparing the traditional GAN (Kamiyar Nazeri et al.), CapsGAN (Batch normalization) and CapsGAN

Epoch	**GAN**			**CapsGAN with Batch Normalization**			**CapsGAN without Batch Normalization**		
	MAE	**RMSE**	**PSNR**	**MAE**	**RMSE**	**PSNR**	**MAE**	**RMSE**	**PSNR**
50	15.4	30.48	19.25	10.7	29.5	19.5	10.95	29.15	19.6
100	13.4	30.1	19.27	10.35	29.28	19.6	11.3	29.32	19.6
150	13.6	29.8	19.4	9.7	29.08	19.7	11.0	29.24	19.7
200	13.52	29.6	19.4	9.6	29.08	19.7	10.6	29.13	19.7

The CapsuleGAN has overall better Mean Absolute Error, Root Mean Squared Error, and Peak Signal-to-Noise error levels which shows its superior performance compared to traditional GAN. The addition of the initial batch normalization and its effect on performance is discussed later

Table 3 Table showing Convergance of MAE, RMSE , PNSR (GAN vs CapsGAN)

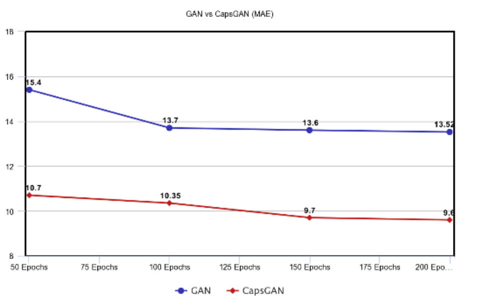
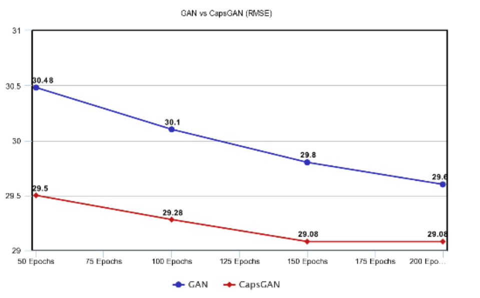
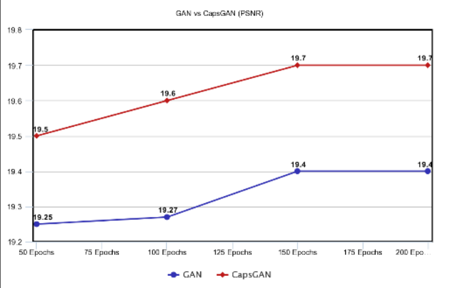
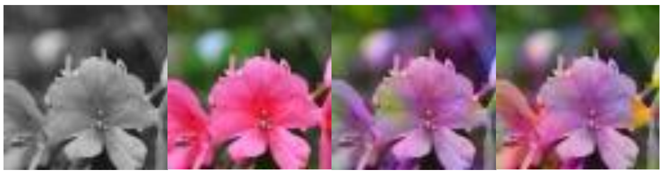
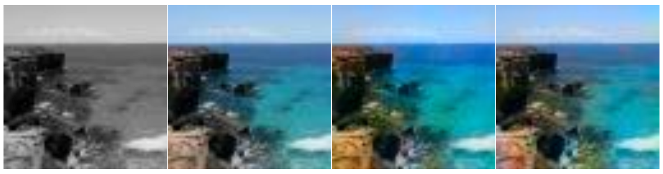
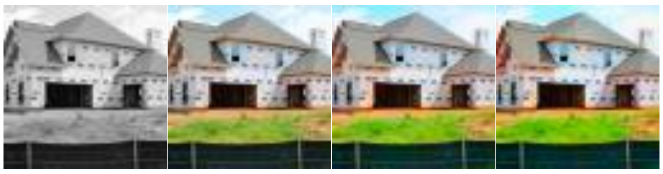
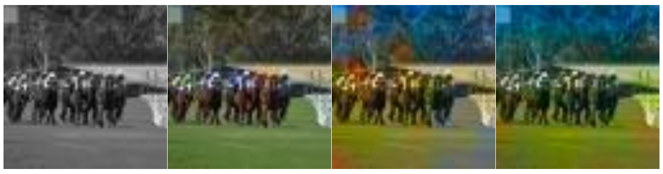
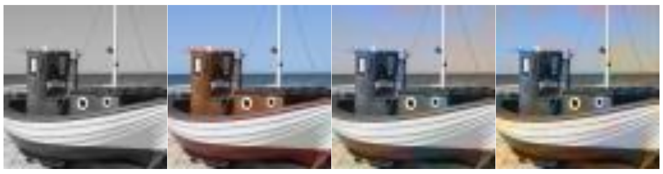
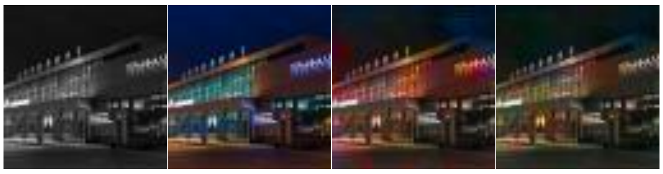
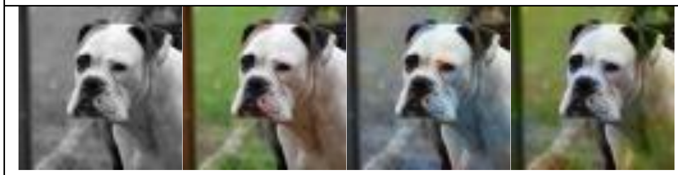
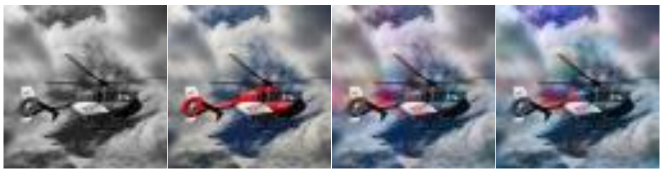
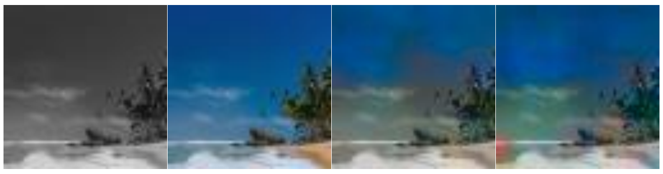
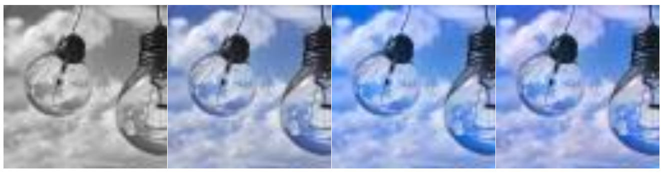
MAE	RMSE	PSNR
		

Table 4 Test Samples *Grayscale, Real, GAN, CapsGAN (MAE, RMSE, PSNR)*

Image	Network	MAE	RMSE	PSNR
	**GAN**	17.1	41.5	15.7
	CapsGAN	12.9	34.2	17.4
	**GAN**	4.5	14.2	25.0
	CapsGAN	5.8	21.0	21.68
	**GAN**	2.9	11.9	26.6
	CapsGAN	3.4	16.2	23.9
	**GAN**	10.9	27.6	19.2
	CapsGAN	8.7	31.6	18.1
	**GAN**	6.6	19.0	22.5
	CapsGAN	5.2	19.7	22.2

	**GAN**	8.0	24.6	20.2
	CapsGAN	5.4	22.0	21.2
	**GAN**	9.4	21.5	21.4
	CapsGAN	1.4	16.5	23.7
	**GAN**	6.2	15.5	24.2
	CapsGAN	5.9	17.2	23.3
	**GAN**	7.4	24.1	20.4
	CapsGAN	5.3	21.9	21.3
	**GAN**	9.1	19.2	22.4
	CapsGAN	6.5	14.2	25.0

4.2. Pixel-wise Accuracy

The results obtained in the pixel-wise accuracy for a threshold (e) of 2% and 5% are shown in the table below, these results reflect how close the produced images are similar to the real ground truth values.

Table 5 Pixel-wise accuracy GAN vs CapsGAN

Epoch	**GAN**		**CapsGAN**	
Threshold	**e = 2%**	**e = 5%**	**e = 2%**	**e =5 %**
50	5.7 %	30.3 %	9 %	31.6 %
100	7.8 %	31.3 %	9.8 %	31.7 %
150	8 %	31.6 %	10 %	32.7 %
200	9.2 %	32 %	10 %	32.7 %

The CapsGAN performs better on both thresholds for the pixel-wise accuracy which shows that it is closer to the image values.

4.3. Parametric Studies

Batch normalization

Not in contrast with the original dynamic routing between capsules [5], a batch normalization layer has been added after the first activation layer in the capsule network. Across different 200 epoch runs, it led to an improved MAE as well as faster convergence than the no batch normalization variant showing as steep curves in the graph. Batch normalization has been proven to improve the convergence speed and stability of the GANs previously [9]. Hence it was added after the first activation layer inside the discriminator.

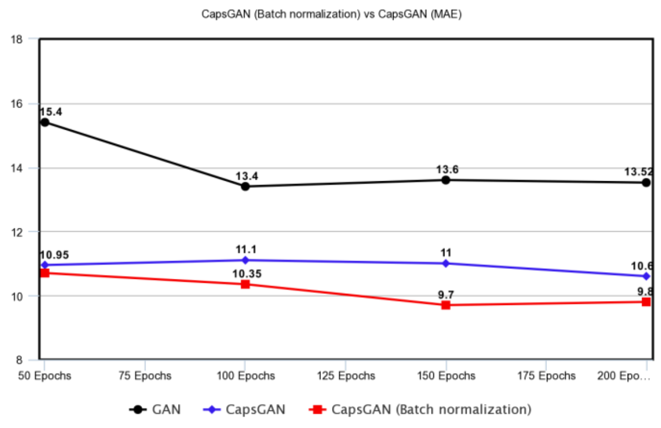


Figure 9 No Batch normalization vs Batch normalization (MAE)

5. Evaluation

5.1. General Colorization

Due to the property of positional equivariance mentioned earlier, capsule networks are expected to have a better general colorization as well as a more consistent colorization over objects inside the image which is expected to be better than previous image colorization methods including GANs which suffered form the problem of desaturation of the images

Notice how the boat and the dog in the GAN colored version has some desaturation, this isn't the case with the CapsGAN colored versions.

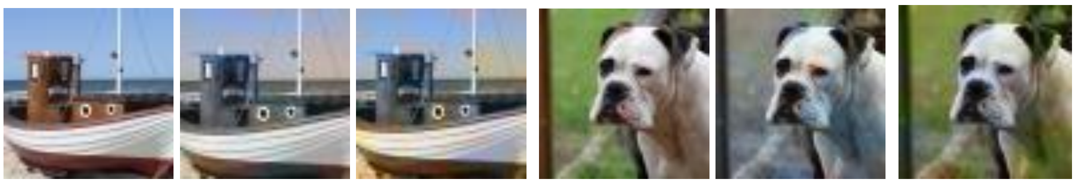


Figure 10 Original, GAN, CapsGAN

To measure this desaturation, the average saturation of the testing data has been calculated

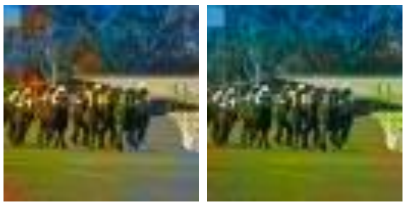


Figure 11 Average Saturation (HSV), left: 0.55 - right: 0.67

Table 6 Average Saturation (HSV) over all testing data: GAN vs CapsGAN

	GAN	CapsGAN
Total Average Saturation	0.3334	0.3802

CapsGAN has more average saturation which is an indication that the images generated are more saturated than that of the GAN.

6. Conclusion

In this project, Capsule Network layers were used in the discriminator of a Deep Convolutional Generative Adversarial Network instead of the traditional convolutional network layers.

The introduced model produced better results on several metrics such as RMSE, MAE, PSNR as well as Pixel-wise accuracy to the real colored values. Also with a better visual color consistency and a better general colorization having solved desaturation problems found in the previous models which was tested by calculating the average saturation calculation.

In addition, several parametric studies were concluded such as trying different kernel sizes for the initial convolutional layer inside the discriminator as well as an initial batch normalization after the first activation layer.

References.

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets.

[2] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. 2016.

[3] Kamyar Nazeri, Eric Ng, Mehran Ebrahimi " Image Colorization with Generative Adversarial Networks", arXiv:1803.05400 , 2018

[4] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In International Conference on Artificial Neural Networks, pages 44–51. Springer, 2011.

[5] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In Advances in Neural Information Processing Systems, pages 3859–3869, 2017.

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

[7] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, pages 234–241. Springer, 2015.

[8] Ayush Jais_wal, Wael AbdAlmageed, Yue Wu, Premkumar Natarajan. "CapsGAN: Generative Adversarial Capsule Network", arXiv:1802.06167,2018

[9] Mehdi Mirza, Simon Osindero, "Conditional Generative Adversarial Nets", arXiv:1411.1784, 2014.

[10] Raeid Saqur, Sal Vivona , " CapsGAN: Using Dynamic Routing for Generative Adversarial Networks "
arXiv:1806.03968 , 2018

[11] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. 2016

[12] Andrej Karpaty. CS231n Convolutional Neural Networks for Visual Recognition course. Stanford University. 2017.

[13] Linnaeus 5: http://chaladze.com/l5/