Image Colorization

Galvão, Duarte
KTH - Royal
Institute of Technology
dadsg@kth.se

Santos, Pedro
KTH - Royal
Institute of Technology
pedrops@kth.se

Sánchez-Mateos Lizano, Raúl
KTH - Royal
Institute of Technology
raulsml@kth.se

Abstract

Automatic image colorization has been of significant interest over the years given its broad range of applications. Over the last decade, several solutions have been proposed to this problem and, on the last few years, state of the art results were achieved using deep neural networks. In this project, we explore and compare three methods for image colorization based on the use of deep models. The first consists of a fully convolutional neural network trained under a supervised scheme, whereas the second method uses a conditional Deep Convolutional Generative Adversarial Network (DCGAN). This method is then extended by placing a classifier in parallel with the discriminator and back propagate the classification error through the network (VAC+GAN). The CIFAR-10 dataset is used to validate and compare all approaches.

1 Introduction

The automatic colorization of grayscale images has been an active area of research over the last decade due to its broad range of applications. Such repertoire include, for example, the restoration of aged or degraded images [1] and creative applications such as comics/manga automatic colorization [2, 3], among others. The key idea behind the color generation process is that the semantics of the scene and its surface texture provide ample cues about the colors of many regions in the image. Thus, by learning the statistical dependencies between the colors and semantics/textures, a plausible and consistent colorization can be generated. The main goal is, usually, not to recover the original colors but instead to produce visually compelling results that can potentially fool a human observer.

Throughout this project we compare three different approaches. The first method was proposed by Zhang et al, 2016 [4]. On this work, a fully convolutional neural network (FCNN) is trained, under a supervised scheme, to output the colored version of an image. We replaced the FCNN architecture with an U-Net [5] and tested its performance using the CIFAR-10 dataset. Secondly, a conditional Deep Convolutional Generative Adversarial Network (DCGAN) was implemented, following the work done by Nazeri et al, 2018 [6]. This network was also validated using the same dataset. This method is then extended by placing a classifier in parallel with the discriminator and back propagate the classification error through the network (VAC+GAN). It has been showed in some works [7] that the GAN framework can be augmented using side information (such as labels) in order to improve the quality of generated samples. The conditioning of the generator and discriminator on the class label would require us to feed the label of the given gray image at test time making the process less automatized. Thus, we opted to only provide information about the class labels to the generator indirectly through the backpropagation of the classifier. With this we hope to increase the generated image quality and global coherency.

For all previously described methods visually acceptable images were generated and all approaches mis-colorized some images. While the samples from the models are, in most cases, roughly similar for all methods, the GANs mis-colorize less images. The VAC+GAN showed an improved global image coherency.

2 Related work

Models for automatic image colorization appeared in the early 2000s [8, 9], however these methods required significant user intervention in order to produce faithful results. Recently, state of the art performance was achieved using deep neural networks models. One of these methods was proposed by Zhang et al, 2016 [4]. On this work, a FCNN is trained, under a supervised scheme, to output the colored version of an image. Another method, proposed by Nazeri et al, 2018 [6], uses a conditional Deep Convolutional Generative Adversarial Network (DCGAN) to generate the colored version of an image from its gray version. Both these methods require no user intervention and are capable of producing state of the art results. Isola et al. 2018 [10] address the reconstruction/conversion of images using a conditional GAN (cGAN). This work proposes a solution to image-to-image translation problems using deep models.

Other works that are closely related to our project are the conditional GAN proposed by Mirza et al. [7], the ACGAN in Odena et al. [11] and the VAC+GAN architecture from Bazrafkan et al. [12, 13]. These works explore ways to improve the quality of the generated samples by incorporating additional information (such as class labels) into the standard GAN framework.

3 Data

The chosen dataset was CIFAR-10, which contains in total 50 000 32x32 images, split in ten classes. On Nazeri et al, 2018 [6] the same dataset is used for experiments.

3.1 Data split

Two train-validation data splits were tested. The first one consists of a naive split in which a number of randomly selected samples composes the validation set and the remainder are used for training. The second data split is done in a manner to prevent biasing towards any particular color. So, the data is sorted by mean hue value of all the pixels, and evenly split between the two sets according to a specific ratio in such a way that both contain similar color distributions. The chosen validation set size was 10% (ratio) of the total number of samples

3.2 Filtering & pre-processing

The CIFAR-10 dataset contains some gray-scale images, which could be detrimental to the training of our models. So, 581 gray images from the original dataset were discarded.

Finally, the images are split into the gray-scale and ground-truths. This is done by converting all images to the L*a*b* color space. Then, the L channel (representing the brightness) is used as the gray-scale images, and the full L*a*b* channels are used as the ground-truths. By doing this, since the color of a pixel is fully encoded on its a* and b* channels, we can fix the L* channel and only generate the values for the two later channels. By doing this we are able to produce consistent results. The images are normalized into the interval $[-1, 1]$ during the pre-processing step.

4 Methods

4.1 Fully convolutional network (U-Net)

The first proposed method consists of a fully convolutional network model [14]. The architecture is based on the idea of a encoder-decoder network in which the input is progressively downsampled throughout the encoder (contractive) path until it reaches the bottleneck of the network. Then, this process is reversed by the decoder network (expansive path) using a composition of upsampling operations. The architecture of the model is symmetric, i.e. the number of encoding units is the same as the number of decoding units. The bottleneck forces the model to find an efficient information representation on the middle layers of the network. This shrink in the information flow from the input to the output layer can harm the performance of the network so skip connections between the contractive and expansive paths were added. This model is known as U-Net [5]. Note that, since the network is only composed of convolutional layers, the memory requirements become more feasible, allowing to train the network with bigger batch sizes. This architecture was proposed on [6], which we closely followed with the exception of some minor tweaks.

The encoder (contractive) path consists of 4×4 convolution filters with stride 2 and '*same*' padding attribute. For each layer batch normalization is used followed by a Leaky-ReLu activation with slope 0.2. At each step the number of filters is doubled.

The decoder (expansive) path consists of 4×4 transposed convolutions with stride 2 and '*same*' padding. For each layer batch normalization is used followed by a ReLu activation and concatenation with the activations coming from the respective (mirrored) layer of the encoder. The last layer of the network consists of a 1×1 convolution followed by an tanh activation function.

The network receives as input x, the gray version of an image with dimensions $[B, W, H, 1]$, where B represents the batch size, W the image width and H the image height. Its output has $[B, W, H, 2]$ dimensions where the two channels represent the A and B values encoding the colors of the respective image. Figure 1 illustrates the network architecture used for the CIFAR-10 dataset.

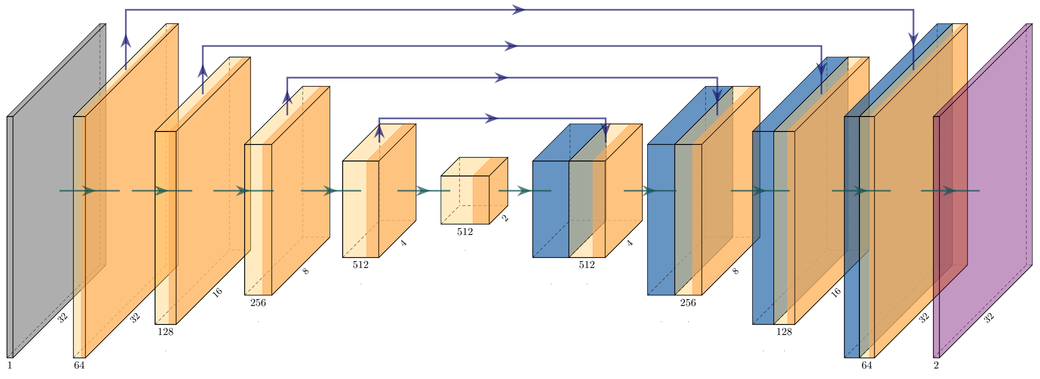


Figure 1: U-Net architecture illustration for the CIFAR-10 dataset.

This network is trained in a supervised way to either minimize the Mean Absolute Error (L_1-norm) or the Mean Squared Error (L_2-norm) between the predicted colors $h(x, \theta)$ and the true colors y:

$$J_{L_1}(x, \theta) = \frac{1}{N} \sum_{c=1}^{2} \sum_{p=1}^{P} \left| h(x, \theta)^{(c,p)} - y^{(c,p)} \right| \tag{1}$$

$$J_{L_2}(x, \theta) = \frac{1}{N} \sum_{c=1}^{2} \sum_{p=1}^{P} \left(h(x, \theta)^{(c,p)} - y^{(c,p)} \right)^2 \tag{2}$$

where N denotes the number of images, θ the parameters of the model and P the number of pixels. Each image is indexed with a tuple (c, p) where c encodes the channel and p the pixel location.

4.2 Generative adversarial network (DCGAN)

The main idea of the Generative Adversarial Network (GAN) framework [15, 16] is to set up a game between two players. One of them, called the generator G, creates samples that are intended to come from the same distribution as the training data. The other player, called the discriminator D, examines samples coming from either the generator or the training set and classifies them as being real or fake. The discriminator learns using supervised learning techniques to classify the inputs as being real or fake. The generator is trained to fool the discriminator, i.e. generate samples that are indistinguishable from the real data. Both players are trained simultaneously until a (nash) equilibrium is reached. At the end of this procedure the discriminator can be discarded and the generator used to produce samples similar to the real data. Figure 2 (left) illustrates the GAN framework.

The generator is represented by a function $G(z, \theta_G)$, differentiable w.r.t. to θ_G, where z is a noise input variable. The discriminator is represented by a function $D(x, \theta_G)$ differentiable w.r.t. to θ_D where x is a colored image. The output of the discriminator consists of a number between 0 and 1, that can be interpreted as the probability of the input x originating from the training data. Then, in an iterative fashion (one player at a time) and against a static version of its adversary, G is trained to minimize the probability that the discriminator makes a correct prediction for the generated data, while D is trained to maximize the probability of assigning the correct label to its input:

$$\min_{\theta_G} J(\theta_D, \theta_G) = \min_{\theta_G} \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))] = \min_{\theta_G} -\mathbb{E}_{z\sim p(z)}[\log(D(G(z)))] \tag{3}$$

$$\max_{\theta_D} J\left(\theta_D, \theta_G\right) = \max_{\theta_D} \left(\mathbb{E}_{x\sim p(x)}[\log(D(x))] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]\right) \tag{4}$$

The equations above define the cost functions required to train a GAN and can be optimized using the well-known backpropagation algorithm. Note that the generator cost function already implements the heuristic suggested by Goodfellow in [16].

For the purpose of image colorization, each player D and G consists of a fully convolutional neural network (FCNN). Furthermore, two modifications to the previously described GAN framework are done.

In the first place, since the objective is to colorize a given gray input image and not to produce an arbitrary image with colors a conditional GAN was used [7]. To accomplish this, the gray image x with dimensions $[B, W, H, 1]$ is used as input to the generator. The discriminator also receives, alongside either a real sample y or a sample from the generator (both with dimensions $[B, W, H, 2]$), the respective gray image x. The noise of the generator is provided in the form of Dropout.

Secondly, the generator cost function was also modified by adding a L_1 or L_2 norm regularization term between the output of the generator and the true colored image y. This forces G to produce colorized images that are close to the real images, as well as preserve the original images structure. With these additions the modified cost functions are presented below:

$$\min_{\theta_G} J\left(\theta_D, \theta_G\right) = \min_{\theta_G} \left(-\mathbb{E}_{z\sim p(z)}\left[\log\left(D\left(G\left(z|x\right)|x\right)\right)\right] + \lambda \left\|G\left(z|x\right) - y\right\|_L\right) \tag{5}$$

$$\min_{\theta_D} J\left(\theta_D, \theta_G\right) = \min_{\theta_D} \left(-\mathbb{E}_{y\sim p(y)}[\log(D(y|x))] - \mathbb{E}_{z\sim p(z)}\left[\log\left(1 - D\left(G\left(z|x\right)|x\right)\right)\right]\right) \tag{6}$$

The network architecture of the generator is the same as the one described on the previous section. The architecture of the discriminator consists of a composition of 4×4 convolutions with stride 2 and '*same*' padding. All layers include batch normalization and leaky-ReLu activation function with slope 0.2. On the last layer of the network a sigmoid activation function is used. This GAN architecture is similar to the one proposed by Nazeri et al. [6].

4.3 Versatile auxiliary classifier with generative adversarial network (VAC+ GAN)

The GAN-framework described on the previous section was extended by introducing a classifier C, called Versatile Auxiliary Classifier (VAC) [12, 13], in parallel with the discriminator. The idea is that, by adding a classifier network that back-propagates through the generator, one can provide additional information to G, improving the quality and global coherency of the generated samples. The classifier accepts samples from both the generator and the dataset, and is trained to minimize the cross-entropy loss (CE) for the correct class label. Figure 2 (right) illustrates the VAC+GAN framework.

The generator loss (Eq. 5) now becomes:

$$\min_{\theta_G} J\left(\theta_D, \theta_G\right) = \min_{\theta_G} \left(-\mathbb{E}_{z\sim p(z)}\left[\log\left(D\left(G\left(z|x\right)|x\right)\right)\right] + \lambda \left\|G\left(z|x\right) - y\right\|_L + \gamma CE\right) \tag{7}$$

where CE denotes the cross-entropy loss for the classifier C. The discriminator loss (Eq. 6) remains unchanged.

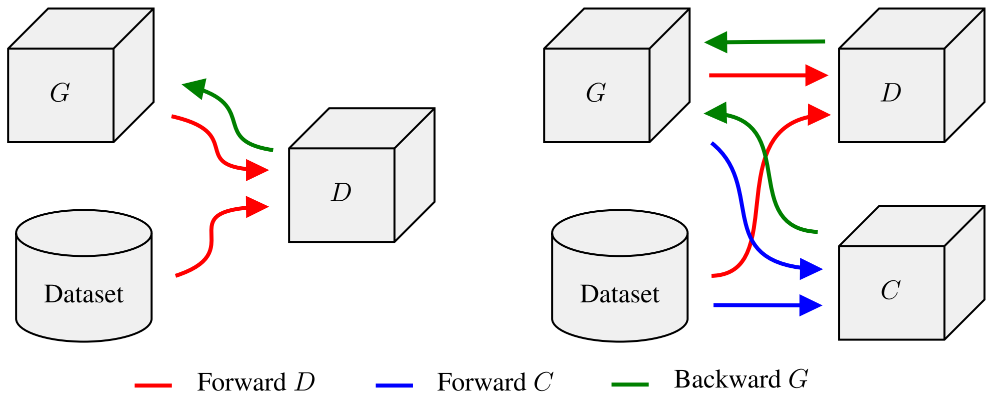


Figure 2: GAN (left) and VAC+GAN (right) frameworks illustrations. G denotes the Generator network, D the discriminator network and C the classifier network.

5 Experiments

All networks (U-Net, GAN and VAC+GAN) were implemented from scratch using a Python environment and Tensorflow. The source code is available at:

https://github.com/duartegalvao/Image-Colorization-with-Deep-Learning

5.1 Training strategies

The models were trained using the Adam optimizer [17] and the learning hyperparameter settings were tuned in order to get the most performance out of each network. The weights were initialized using the Glorot normal initializer [18].

The U-Net network (for both MSE and MAE losses) was trained with an initial learning rate of $\eta = 1e\text{-}03$, decayed by 0.3 each 17500 update steps. The model was trained for 200 epochs and a batch size of 128 was used.

For the GAN network, apart from the already mentioned details on the previous sections, some improvements were tried in order to stabilize the learning procedure. Some of these included one-sided label smoothing, different Dropout strengths for the generator network, different number of updates at each iteration for the generator and discriminator, among others. The network was trained with an initial learning rate of $\eta = 3e\text{-}04$ decayed by 0.1 at each 20000 steps. The model was trained for 200 epochs with a batch size of 128 and an L_1 norm (MAE) regularizer with parameter $\lambda = 100$. Figure 3 plots the evolution of the training losses for the previously mentioned hyperparameters.

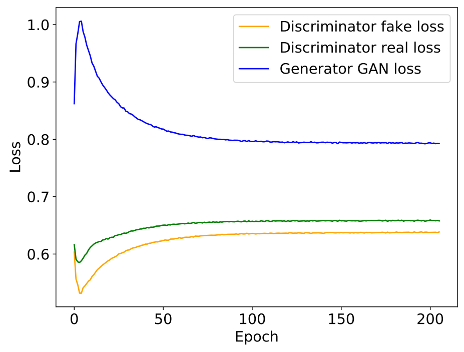
Finally, the VAC+GAN network was trained with the same hyperparameters as the GAN. The γ parameter was set to 0.01.

For all networks the learning process converged and typical loss curves were observed.

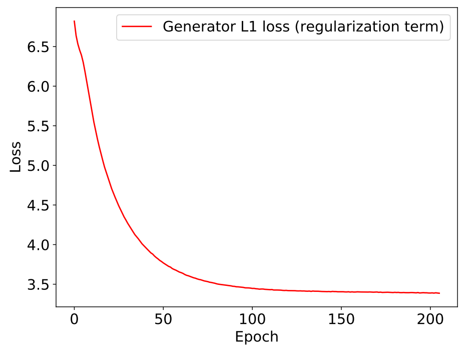
5.2 Results

Firstly, the U-Net was tested with both MAE and MSE losses. Both networks were able to produce visually acceptable solutions. Figure 4 shows, on columns (c) and (d), some samples generated by both networks. The two models showed a similar performance, however, the generated samples from the network trained with MSE appeared slightly more blurry. Therefore the U-Net trained with the MAE loss was selected for further comparison with the GANs architectures.

For the GAN, similar results where obtained in comparison to the U-Net. However, the number of mis-colorized images decreased. For most of the generated images the generated colors look realistic. Our results are close to the ones shown in Nazeri et al, 2018 [6]. Figure 4 displays some resulting samples on column (e).



(a) GAN losses.



(b) L_1 regularization term loss.

Figure 3: GAN training losses. (a) plots the GAN losses and (b) plots the L_1 regularization term that was added to the generator loss.

With the VAC+GAN, the results were similar when compared to the GAN. However, in most of the cases, some small areas of the image that were misinterpreted on the GAN model are now fixed, improving the global coherency of the generated image. Figure 4 displays the results on column (f).

Table 1 shows the losses calculated on the CIFAR-10 dataset. For all networks some images were mis-colorized. One should note that the resulting images were mostly qualitatively compared due to the non-existence of a good quantitative metric. In some cases the MAE error metric revealed inappropriate to estimate the performance of the network. Sometimes a visually appealing image was generated but still yielded a high loss because the generated color scheme was different than the original image. Moreover, a turing test has been carried out in such a way that a person must recognize whether the image shown is the original or the colored one. It can be seen that the results show that in the best case (VAC+GAN), it is possible to differentiate in 40% of the cases.

	Train (MAE)	Validation (MAE)	Test (MAE)	Turing test
UNET	0.0239	0.0691	0.0677	60.30%
GAN	0.0288	0.0672	0.0675	47.43%
VAC + GAN	0.0371	0.0666	0.0668	40.26%

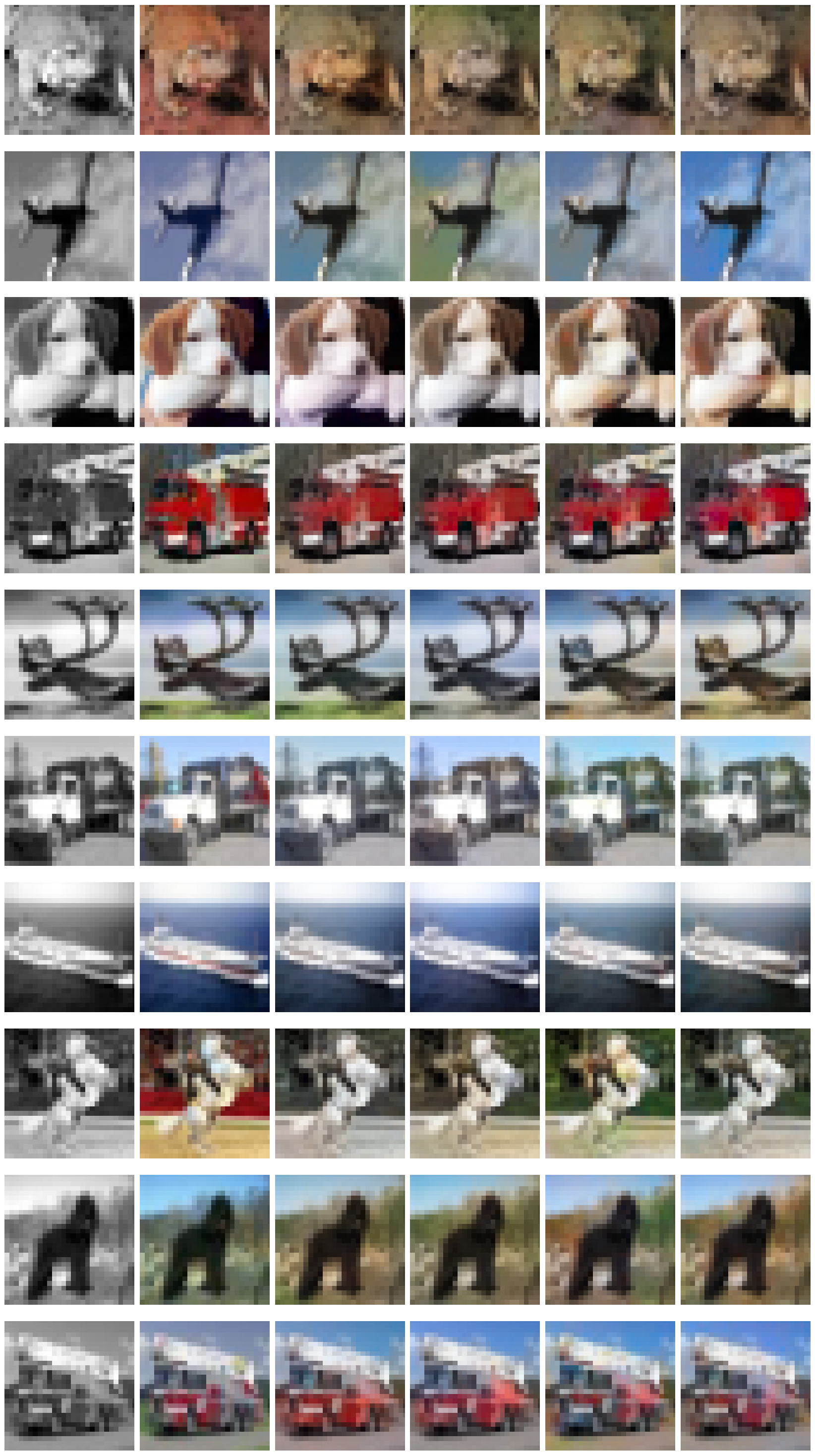
Table 1: Results for the CIFAR-10 dataset calculated for 5 000 samples. The U-Net was trained to optimize the MAE loss. Both GANs are trained with the L_1 regularizer.

6 Conclusion

Throughout this project we implemented and compared different approaches to the Image Colorization problem. For all the implemented solutions visualize acceptable image were generated. For all tested methods some images were mis-colorized. We believe that this happened because the model misinterpreted the texture of some parts of the image. In our opinion, the use of larger images could help to mitigate this. The use of the GANs architectures helped to reduce the number of mis-colorized images. The addition of the auxiliar classifier in the VAC+GAN greatly improved the global coherency of the generated images.

Importantly, and in our opinion a big shortcoming in this work, is the non-existence of a good quantitative metric to measure performance. The resulting images were mostly qualitatively compared making the process more error-prone and subjective.

As a suggestion to future work, we would like to train/evaluate the described methods with images of higher dimension. We would also like to do a more extensive search for hyperparameters.



(a) (b) (c) (d) (e) (f)

Figure 4: Results of the image colorization process for the different implemented methods. (a) Grayscale, (b) Original, (c) U-Net with MSE, (d) U-Net with MAE, (e) DCGAN, (f) VAC+GAN

References

[1] Deoldify - colorizing and restoring old images. `https://github.com/jantic/DeOldify`.

[2] Paulina Hensman and Kiyoharu Aizawa. cgan-based manga colorization using a single training image. *CoRR*, abs/1706.06918, 2017.

[3] Chie Furusawa, Kazuyuki Hiroshiba, Keisuke Ogaki, and Yuri Odagiri. Comicolorization : Semi-automatic manga colorization. *CoRR*, abs/1706.06759, 2017.

[4] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[6] Kamyar Nazeri and Eric Ng. Image colorization with generative adversarial networks. *CoRR*, abs/1803.05400, 2018.

[7] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[8] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. Transferring color to greyscale images. *ACM Trans. Graph.*, 21(3):277–280, July 2002.

[9] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, August 2004.

[10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[11] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2017.

[12] Shabab Bazrafkan, Hossein Javidnia, and Peter Corcoran. Versatile auxiliary classifier + generative adversarial network (vac+gan); training conditional generators. *CoRR*, 05 2018.

[13] Shabab Bazrafkan and Peter Corcoran. Versatile auxiliary classifier with generative adversarial network (vac+gan), multi class scenarios. *CoRR*, abs/1806.07751, 2018.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. *NIPS*, pages 2672–2680, 2014.

[16] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[18] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.