Improved IR-Colorization using Adversarial Training and current Deep Learning Design Patterns

Matthias Limmer
matthias.limmer@gmail.com
Hendrik P.A. Lensch
hendrik.lensch@uni-tuebingen.de

Daimler AG
Ulm, Germany
Department of Computer Graphics
Eberhard Karls Universität
Tübingen, Germany

Abstract

This paper investigates deep learning based image translation techniques to perform a colorization of near-infrared (NIR) images. Current deep neural network design patterns like inception modules, skip-connections or multi-scalar processing are combined and examined. Adversarial training is used to improve the realism of the resulting image in the translated domain. The presented approach is trained and evaluated on NIR-RGB image pairs from a real-world dataset containing a large amount of road scene images in summer. The dataset was captured by a multi-CCD NIR/RGB camera to ensure a perfect pixel to pixel registration.

1 Introduction

In *advanced driver assistance systems* (ADAS), infrared cameras are frequently used as a sensor for image recognition. Compared to regular RGB cameras, they can achieve a higher sensitivity, because of the absence of color or IR-cut filters. Further, in the case of *near-infrared* (NIR), "invisible" headlights are utilized to illuminate the scene without blinding other road users. The disadvantage, though, is on one hand that such cameras produce in general monochromatic images and on the other hand that these images appear unnatural. The unfamiliar appearance for the human eye is due to the reflectance properties of objects in the NIR light spectrum, which can severely differ from the well known RGB spectrum. In a displaying ADAS function, "familiar" RGB images are preferred to be shown to the driver to increase its acceptance and subsequently its take-rate.

One way to achieve this is the installation of a second sensor only for displaying purposes. This sensor's output is augmented by additional information extracted from the infrared sensor. Neglecting the fact that regular low-cost cameras have issues producing high quality images at night, installing multiple sensors in a vehicle increases the production cost. From an economic point of view, a single sensor solution is therefore strongly encouraged.

Automatic colorization of grayscale images appears to be closely related to IR-Colorization, but has a great advantage. Grayscale images can directly serve as the luminance component in a decorrelated color space, like the CIELAB color space. Hence, most of the

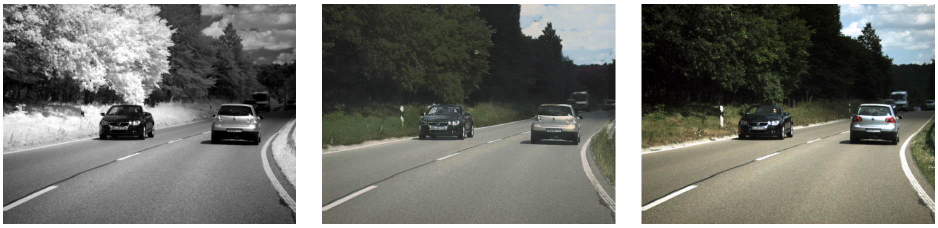


Figure 1: An NIR image (Left) is colorized (Middle) by the approach described in this paper and strongly resembles the corresponding groundtruth RGB image (Right). Best viewed in color.

approaches exploit this circumstance and only estimate the missing chrominance components. Furthermore is the human eye less sensitive to chrominance inaccuracies. For IR-Colorization, though, it is crucial to estimate the luminance component as well.

The approach proposed in this paper utilizes deep *Convolutional Neural Networks* (CNN), but incorporates current developments of network architectures and training schemes, such as *Adversarial Training* [12], *Inception Modules* [39], *U-nets* [32] or multi-scalability [26]. The developed network architecture in this paper resembles an estuary with its many branches joining into the sea and is therefore called *Estuary Network*. Extensive experiments and results, as depicted in Figure 1, show the potential of the proposed approach.

2 Related Work

IR-colorization can be more understood as an image translation problem rather than a classic grayscale colorization problem. While classic colorization techniques only need to estimate additional information, namely the chrominance, IR-Colorization actually needs to transform the input into another domain. Example-based advanced *Color Transfer* techniques for example need to perform such a domain transfer. In these approaches, fitting exemplary transforms are looked up in an offline acquired database and applied to the input to perform the transfer. Most notable approaches transform e.g. day images into night images [35] or summer images into winter images [22]. The major drawback within these approaches is the complex processing cue. A maintenance and retrieval mechanism of the best examples must be implemented, as well as the transfer operation itself. Deep learning approaches, on the other hand, inherently learn these mechanisms during training and, thus, reduce the modeling effort considerably.

In the recent years, quite a few deep learning based grayscale colorization approaches emerged [3, 5, 6, 16, 24, 47], which might be extensible to IR-Colorization. They can be subdivided into approaches that quantize the color space and estimate "color classes" [24, 47] and approaches that directly estimate continuous color values [3, 5, 6, 16]. Class-based approaches work well with smaller, but do not scale for higher image resolutions. The memory required to store probability maps for a larger amount of color classes is unfeasible. Direct inference does not share this constraint, but often suffers in color fidelity because of the indiscriminative loss function [47]. Limmer and Lensch [26] propose an IR-Colorization network based on direct inference. It is capable of producing full resolution RGB images, but due to the nature of the *coherence gap* [26], the raw network output is noisy and blurred. A post-processing step is necessary to blur the noise and recover the lost details at the expense of correctness and realism in the output.

Neural networks that explicitly enforce realism in the training objective are *Generative Adversarial Networks* (GAN) [12]. Their adversarial training scheme was quickly adopted for all kinds of applications: style transfer [25, 43, 49], image inpainting [31], 3D-model generation [44] or grayscale colorization [3], just to name a few. Many other works address the issue of image translation in general, namely transforming an image from one domain into another [2, 8, 18, 20, 28, 40, 50]. These approaches use hand-tailored network topologies or training schemes to achieve optimal results.

Contrary to that, Isola *et al.* [18] introduced an approach that can be applied to a variety of image translation tasks, while achieving reasonable results for all of them. They employ a conditional GAN consisting of a U-Net [32] architecture as the generator and the truncated encoder part of a similar U-Net as the discriminator. Further do they combine the adversarial loss with an additional L_1-loss to train the generator. The results and flexibility of their approach is promising considering IR-Colorization.

Based on the approaches by [18] and [26], this work explores new network structures incorporating current CNN design patterns [37]. To increase the realistic impression of the generated output, adversarial training is used and evaluated. Finally, we show that, by using the right design patterns, existing networks can be enhanced to not only perform IR-colorization, but also improve the quality of such colorizations compared to previous approaches.

3 Approach

3.1 Encoder-Decoder Networks

Encoder-Decoder networks are deep neural networks that first consecutively encode a signal into a code-vector of lower spatial resolution as the input (bottleneck) and then consecutively decode it to a signal in the target domain. Many well performing deep network architectures [14, 36, 38] are only designed as encoders, where the output is of lower spatial resolution than their input. To maintain the input resolution, but also leveraging the encoding effect, *Fragmentation Layers* [10, 41, 42] can be used. Numerous network architectures [4, 26, 27, 34, 46] utilize this technique in a variation commonly known as *dilated convolution*. The common practice, however, is to keep the encoding structure and to design an additional decoder part for the network. The decoder often consists of transposed convolutions with fractional stride, which serve as trainable upsampling operators. If the code vector, though, is not expressive enough, upsampled outputs lack details lost in the downsampling process [18, 29]. To recover this information *skip-connections* have been developed. These skip-connections combine intermediary feature maps of the encoder with corresponding feature maps of the decoder. This is closely related to residual connections, see He *et al.* [14]. Residual connections only span over singular network layers or modules instead of larger network parts. While Long *et al.* [29] combine the feature maps by a simple element-wise addition, Ronneberger *et al.* [32] use channel-concatenation in their *U-Nets* and let a following (transpose-)convolution handle the summation. Isola *et al.* [18] apply a U-Net based network structure in their approach.

3.2 Multi-Scalar Networks

While Encoder-Decoder networks implicitly handle multi-scalar processing of the input, other approaches explicitly handle multi-scalar input, e.g. [7, 9, 26, 27, 45]. These ap-

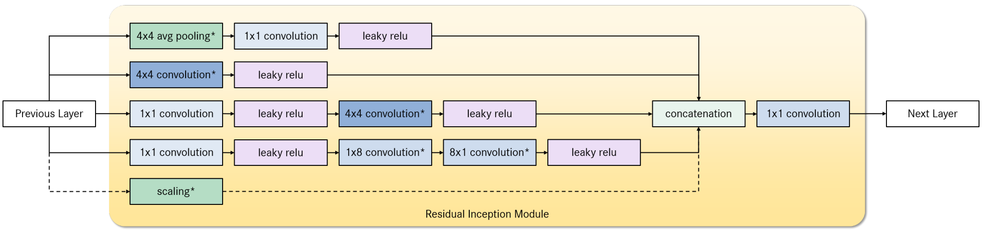


Figure 2: The residual inception module used in this work. If the Inception module is used as a downsampling module, layers with an asterisk have a stride of 2. As an upsampling module, these layers are fractional strided (1/2) transpose convolutions instead of convolutions and bilateral upsampling layers instead of average poolings. The scaling module in the residual connection does nothing, if down- or upsampling is not required. If residual connections are disabled, this branch is skipped.

proaches construct an image pyramid with regular downsampling operations before processing every scale concurrently in their own branch of the network. The outputs of each branch are finally combined after they have been upsampled to the same resolution. Approaches like [7, 45] train the network branch by branch, while [9, 26, 27] train the network with all branches end to end. Results by [26] show that multiple scales strongly benefit the network's recall performance.

3.3 Inception Module

Szegedy *et al.* [38] published as a key component to their *GoogLeNet* the so called *Inception Module*. These are miniature networks that contain various independent branches of network layers, which are concatenated in the end. Each branch in a module has its own layer structure. A module may contain various combinations of computationally lightweight layers, like overlapping average pooling, 1×1-convolutions or 3×3-convolutions. Inception modules result in a greater variety of feature maps and increase the depth of the network, while the computationally lightweight components ensure a controllable increase of computational complexity [38]. Recent developments of the GoogLeNet architecture also incorporate up- and downsampling inception modules and residual connections [14] across regular inception modules [39]. The inception module used in this work is depicted in Figure 2.

3.4 Recombinator and Estuary Networks

Recombinator Networks (RCN) by Honari *et al.* [15] combine multi-scalar processing with the successive upsampling of U-Nets. After building an image pyramid, each scale is first concatenated with the upsampled output of the previous branch, before it is processed by various trainable layers. Up- and downsampling routines in RCNs are static network layers, like average pooling and nearest neighbor upsampling.

In this work, a new network structure is outlined, called Estuary Networks. Similar to RCNs, they also combine multi-scalar networks with U-Nets. An Estuary Network first produces multiple scales of its input. Each scale is then processed by its own encoder network branch, which subsequently encodes its input to the spatial size of the lowest scale using a downsampling module. In every downsampling step the amount of channels is doubled beginning with a base channel size n_c until a certain depth is reached. The decoder

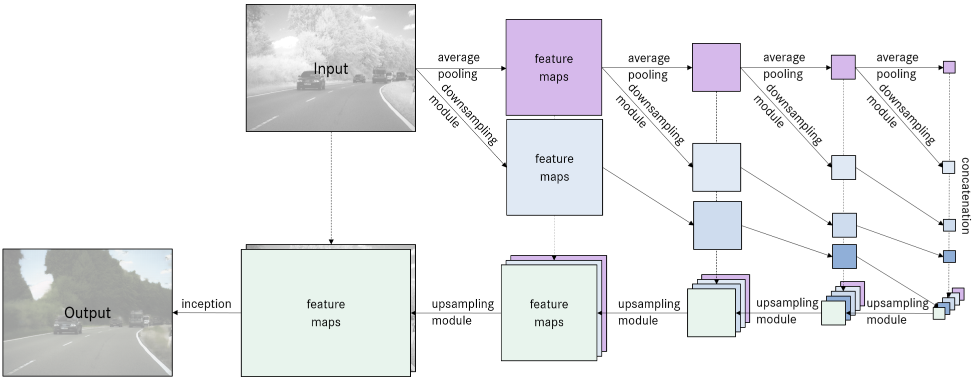


Figure 3: Schematic overview of an estuary network. Downsampling and upsampling modules can either be inception modules or (fractional) strided (transpose-) convolutions. All feature maps in each scale are concatenated before the next upsampling module.

of an Estuary Network is structured similarly to the decoder of the U-Net. The only difference is that an estuary decoder combines all available feature maps of a resolution across all branches, also called hypercolumns [13]. Contrary to RCNs, all down- and upsampling routines are trainable. The downsampling modules can either be strided convolution layers or downsampling inception modules. Upsampling modules are, thus, fractional strided transpose-convolutions or upsampling inception modules. The amount of channels of a single branch inside an inception module is 1/4 of the current amount of channels, so that a concatenation of the feature maps results in the correct amount of channels. Residual connections for upsampling and downsampling modules are implemented as average pooling and bilateral upsampling. Before a down- or upsampling module is used, feature maps are normalized by a batchnorm-layer [17], followed by an activation function. In the encoding branches the activation function is a leaky ReLU with a slope of 0.2 and in the decoding part a regular ReLU. The Estuary Network structure is displayed in Figure 3.

3.5 Adversarial Training

Generative Adversarial Networks have been introduced by Goodfellow *et al.* [12]. They describe a network architecture and training scheme that trains concurrently two separate networks. One network, the generator, produces samples of a target distribution that are rated by another network, the discriminator. The discriminator is trained to distinguish, if an input sample came from the generator or from the original dataset, while the generator is trained to "fool" the discriminator. This procedure is called adversarial training.

Let G be the generator and D be the discriminator and the tuple $(x,y) \in P_{x,y}$ a sample from the dataset $P_{x,y}$, where x is from the source domain P_x and y the correspondent from the target domain P_y. $G : x \to y'$ is then a function that generates fake samples $y' \sim P_y$, which resemble y. Further is $D : y \to d$ a function that computes a probability $d \in [0..1]$ that y was a real sample from P_y. The adversarial loss $\mathcal{L}_{AD}$ can then be defined as:

$$\mathcal{L}_{AD}(G,D) = \mathbb{E}_{y\sim P_y}[\log D(y)] + \mathbb{E}_{x\sim P_x}[\log(1-D(G(x))]$$

This loss increases if generated samples from G can not be distinguished from real samples by D. Consequently, $\mathcal{L}_{AD}$ is minimized to optimize D. For optimizing G, this formulation is slightly modified to define a minimization problem as well:

$$\mathcal{L}_{AG}(G,D) = \mathbb{E}_{x\sim P_x}[\log(D(G(x))]$$

Here, the loss increases, if samples of G are not classified as samples from P_y. According to [18], it is beneficial to support D additionally with the original input sample x. Both losses can therefore be conditioned on x by feeding the tuples (x,y) and $(x,G(x))$ to D. Combining the adversarial loss with one or multiple traditional objectives, like the L_1-distance, is reported to stabilize adversarial training (see [2, 18, 23, 31, 43, 44, 50]). Other approaches also define alternative or modified adversarial losses [30, 33, 48]. In this work an additional L_1-distance loss is used to train G:

$$\mathcal{L}_{L_1}(G) = \mathbb{E}_{x,y\sim P_{x,y}}[||y - G(x)||_1]$$

The resulting loss function for G is thus:

$$\mathcal{L}_G(G,D) = (1-\lambda)\cdot\mathcal{L}_{AG}(G,D) + \lambda\cdot\mathcal{L}_{L_1}(G),$$

where $\lambda \in [0..1]$ is a linear interpolation factor between $\mathcal{L}_{AG}$ and $\mathcal{L}_{L_1}$.

4 Experiments

In the following, various experiments are conducted. After describing a few details of the used dataset in Section 4.1, the basic training configuration is explained in Section 4.2. In Section 4.3 several variations of the adversarial loss are discussed, followed by experiments regarding skip- and residual connections in Section 4.4. Finally Estuary networks are compared against other networks suitable for image translation tasks in Section 4.5.

4.1 Dataset

The dataset used for NIR-RGB translation tasks consists of 38.495 pixel-registered NIR-RGB image pairs of rural road scenarios with a resolution of 768×1024. These pairs have been sampled from video sequences taken on sunny days in summer and show a variety of roads, landscapes, fields and trees. This setting was chosen, because it contains a vibrant color distribution compared to other possible settings. The dataset is split into 800 evaluation and 32.795 training image pairs, where each fraction was recorded on different days and locations.

4.2 Training

All networks have been trained with almost the same hyperparameters. Every training was performed for 60.000 iterations with a batchsize of 1 and a constant learning-rate of $\eta = 0.001$ on the full-resolution images of the training set. The input images were normalized to a range of $[0..1]$. Weights were initialized using the *Xavier* initializer of Glorot and Bengio [11] and optimized by the *Adam* optimizer of Kingma and Ba [21]. Both components use their default parameters. Variables β and γ of all batchnorm layers were left trainable and, contrary to common practice, running averages of the mean and standard deviation were not used during recall, because using them resulted in severely distorted outputs. The evaluated estuary networks contained seven downsamplings in the longest branch of the generator,

Table 1: L_1-error of Estuary Networks conditionally and unconditionally trained with various λ.

Architecture	L_1-error				
	$\lambda = 0.01$	$\lambda = 0.01$, C	$\lambda = 0.99$	$\lambda = 0.99$, C	$\lambda = 1.00$
Estuary-16-S	0.123329	0.131221	0.093502	**0.090415**	0.090995
Estuary-16-IS	0.135006	0.124661	0.089726	**0.083169**	0.085258

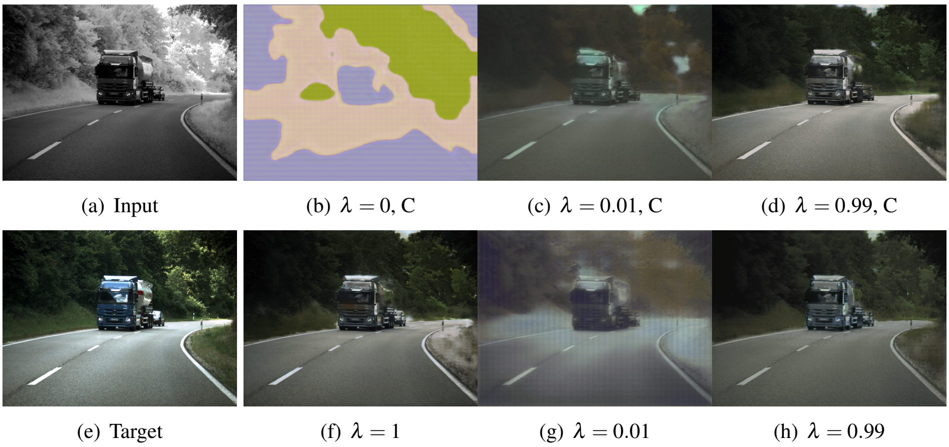


Figure 4: Example colorizations of network Estuary-16-IS conditionally (upper row) and unconditionally (lower row) trained with various values of λ.

starting with $n_c = 16$ and doubling up to $n_c = 128$. The discriminator for all experiments is a singular downsampling convolution layer with a filter bank size of $n_c = 128$. One-sided-label-smoothing [33] is used in its loss $\mathcal{L}_{AD}$, which, depending on the experiment, is either conditional or unconditional. The used training framework is *Tensorflow* [1].

In the following, various variants of network architectures are referred to by appending suffixes to the architecture names depending on the existence of certain design patterns. The usage of inception modules instead of (transposed) convolution layers is indicated by "I". Using skip-connections, residual connections or a conditional discriminator are indicated by "S", "R" and "C", respectively. An Estuary network using all mentioned design patterns with $n_c = 16$ is thus referred to as "Estuary-16-ISRC".

4.3 Adversarial Loss

To examine the influence of the adversarial loss on the IR-RGB translation performance, the following parameters have to be taken into consideration. First, the interpolation factor λ between $\mathcal{L}_{L_1}$ and $\mathcal{L}_{AG}$ determines how strongly the adversarial objective dominates the training. When λ is set to 0, the network is trained only with $\mathcal{L}_{AG}$, when set to 1, the network is solely trained with $\mathcal{L}_{L_1}$. Derived from [18], the standard value of λ is 0.99 in all experiments if not denoted otherwise. λ values of $0, 0.01, 0.99$ and 1 have been examined. The learning rate was decreased for $\lambda \in \{0, 0.01\}$ to $\eta = 5 \cdot 10^{-5}$ for the training to converge. Second, D can either be conditional or unconditional. This only takes effect when $\lambda \neq 1$. Estuary networks with $n_c = 16$, without residual, but with skip-connections and with and without inception modules were trained with these seven parameterizations of λ. Table 1 lists the L_1-error on the evaluation dataset.

Table 2: L_1-error of Estuary Networks with and without skip- and residual connections.

Architecture	L_1-error			
	—	R	S	SR
Estuary-16	0.136024	0.139218	0.100473	**0.093502**
Estuary-16-I	0.124239	0.133915	0.094941	**0.089726**

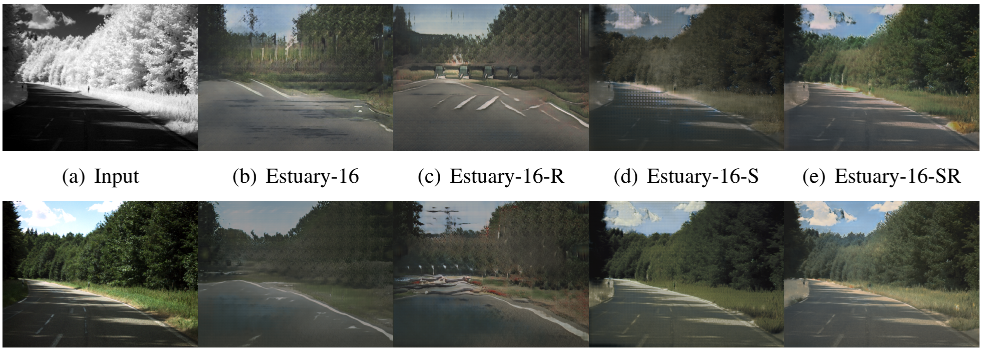


Figure 5: Colorizations for an example image of the evaluation dataset showing the influence of skip-connections and residual connections.

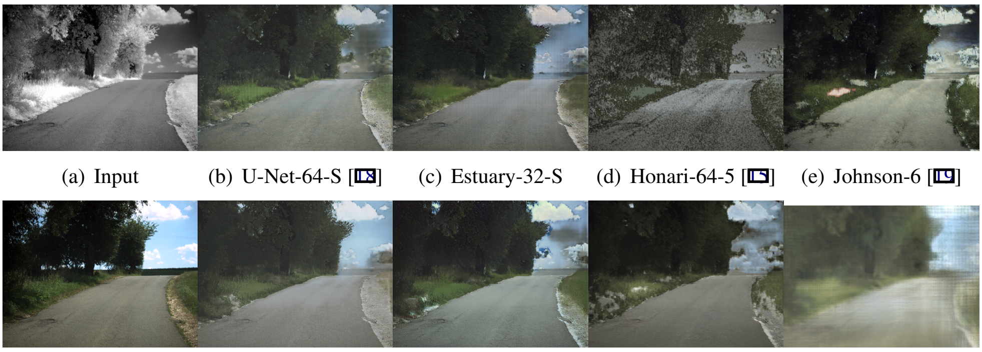
It can be seen that setting $\lambda = 1$ and or $\lambda = 0.99$ with a conditional discriminator results in the lowest L_1-loss. It seems that conditioning the discriminator encourages it to penalize similarly to the L_1-loss alone. Slightly worse performs the unconditional discriminator with $\lambda = 0.99$. The adversarial loss apparently tends to push generated samples away from the optimum in an L_1-sense. Networks trained with $\lambda = 0$ hardly converge, even with low learning rates. Networks trained with $\lambda = 0.01$ converge much slower compared to $\lambda = 0.99$ and $\lambda = 1$ and, thus, generate output images of lower quality. Figure 4 displays a comparison of output images trained with these λ configurations.

4.4 Skip- and Residual Connections

In the following, the influence of skip- and residual connections and their interaction in estuary networks with inception modules is evaluated. Therefore, structurally similar estuary networks are trained with and without skip- and residual connections ($\lambda = 0.99$, unconditional). Table 2 shows the L_1-error of the evaluated networks. Skip connections increase the L_1-performance considerably compared to their counterparts. Residual connections, though, appear not to show a clear tendency regarding the L_1-error. This reflects the findings in [39] reporting that residual connections mostly increase training speed, but not improve the convergence point. Inspecting Figure 5, residual connections appear to have another effect, though. Generated images with residual connections appear sharper and less blurry. It must also be denoted that the images generated without skip-connections hardly reconstruct original details. They rather consist of image patch patterns that resemble real ones.

Table 3: L_1-error for various network architectures, their amount of trainable parameters in million and average execution time for an 1024×768 image on an NVIDIA TITAN X (Pascal) GPU in seconds.

Architecture	L_1-error	n_{param} [m]	timing [s]
U-Net-64-S	0.108879	54.4	0.089
U-Net-64-IS	0.103525	25.1	0.171
Estuary-32-S	0.100113	68.6	0.241
Estuary-32-IS	**0.083664**	28.3	0.323
Limmer-3-12-3-bp	0.103510	1.4	0.263
Johnson-6	0.126073	2.0	0.128
Honari-64-5	0.179843	0.5	**0.077**
Honari-64-5-I	0.120775	**0.2**	0.204


(a) Input (b) U-Net-64-S [18] (c) Estuary-32-S (d) Honari-64-5 [15] (e) Johnson-6 [19]

(f) Target (g) U-Net-64-IS (h) Estuary-32-IS (i) Honari-64-5-I (j) Lim-3-12-3-bp [26]

Figure 6: Colorizations for an example image of the evaluation dataset using various network architectures for image translation tasks.

4.5 Network Comparison

In this section the various network architectures for image translation tasks are compared against each other. The baseline network is on one hand the U-Net-64 structure by [32] used in [18] as is and modified by replacing all (transpose-)convolutions with inception modules. On the other hand, the second baseline network is the multi-scale network by [26] without their post-processing step. This network was not trained with an adversarial loss. Further evaluated networks are Johnson-6 [19], which is used in [50], and Honari-64-5 [15] with $n_c = 64$ and 5 branches. Lastly Estuary-32 networks with and without inception modules and skip connections are compared. Estuary-32 networks were chosen, because they have a similar amount of network parameters compared to U-Net-64 networks. Table 3 shows the L_1-error of these topologies and the amount of trainable parameters and their timing in seconds on a single NVIDIA TITAN X (Pascal) GPU.

Regarding the L_1-error, U-Nets [18, 32] and Estuary networks outperform network architectures by Johnson *et al.* [19], Honari *et al.* [15] or Limmer and Lensch [26]. Estuary networks tend to perform visually similar to U-Nets. Using inception modules instead of simple convolutions boosts the recall performance considerably and reduces the amount of trainable parameters a large amount at the cost of execution speed. The RCN architecture (Honari-64-5) has the highest execution speed, but can not reach the objective and visual performance of the other network architectures, even when using inception modules. Figure 6 shows images generated by the discussed topologies. The inception variants of Estuary networks and U-Nets perform visually similar and appear to produce sharper results than their variants without inception modules.

5 Conclusion

This paper introduced measures to improve deep learning based IR-Colorization. The proposed network architectures combine CNN design patterns, like U-Nets, multi-scalar networks, inception modules, skip- and residual connections and adversarial training. Experiments conducted in this paper indicate that replacing simple (transpose-)convolutions with inception modules has a large positive impact on the objective estimation error and reduces the amount of network parameters considerably at the cost of execution speed. Skip-connections further decrease this error and are an inevitable component to get appealing output images, while residual connections appear to have almost no impact. Using adversarial training combined with a traditional loss function tends to increase the realistic impression of its output, while reducing the objective performance. Using a conditional adversarial loss did not have a big impact on the objective performance. Although Estuary networks with Inception modules often acquire the best objective error, U-Nets without Inception modules possess a better trade-off between execution speed and objective performance, while maintaining a similar visual appearance.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[2] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks. Technical report, 2016. arXiv:1612.05424.

[3] Yun Cao, Zhiming Zhou, Weinan Zhang, and Yong Yu. Unsupervised Diverse Colorization via Generative Adversarial Networks. Technical report, 2017. arXiv:1702.06674.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *International Conference on Learning Representations*, 2015. arXiv:1412.7062.

[5] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep Colorization. In *Proceedings of the International Conference on Computer Vision*, 2015.

[6] Ryan Dahl. Technical report, 2016. URL http://tinyclouds.org/colorize/.

[7] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *Advances in Neural Information Processing Systems*. 2015.

[8] Hao Dong, Paarth Neekhara, Chao Wu, and Yike Guo. Unsupervised Image-to-Image Translation with Generative Adversarial Networks. Technical report, 2017. arXiv:1701.02676.

[9] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning Hierarchical Features for Scene Labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[10] Alessandro Giusti, Dan Claudiu Ciresan, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Fast Image Scanning with Deep Max-Pooling Convolutional Neural Networks. In *Proceedings of the IEEE International Conference on Image Processing*, 2013.

[11] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, 2010.

[12] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*, 2015.

[13] Bharath Hariharan, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. arXiv:1512.03385.

[15] Sina Honari, Jason Yosinski, Pascal Vincent, and Christopher Pal. Recombinator Networks: Learning Coarse-to-Fine Feature Aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[16] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *Proceedings of ACM SIGGRAPH*, 2016.

[17] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the International Conference on Machine Learning*, 2015.

[18] Phillip Isola, 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*, 2017.

[19] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proceedings of the European Conference on Computer Vision*, 2016.

[20] Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts. Technical report, 2016. arXiv:1612.00215.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

[22] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient Attributes for High-Level Understanding and Editing of Outdoor Scenes. *ACM Transactions on Graphics*, 2014.

[23] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding Beyond Pixels Using a Learned Similarity Metric. In *Proceedings of the International Conference on Machine Learning*, 2016.

[24] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning Representations for Automatic Colorization. Technical report, 2016. arXiv:1603.06668.

[25] Chuan Li and Michael Wand. Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks. In *Proceedings of the European Conference on Computer Vision*, 2016.

[26] Matthias Limmer and Hendrik P. A. Lensch. Infrared Colorization Using Deep Convolutional Neural Networks. In *Proceedings of the International Conference on Machine Learning and Applications*, 2016.

[27] Matthias Limmer, Julian Forster, Dennis Baudach, Florian Schüle, Roland Schweiger, and Hendrik P. A. Lensch. Robust Deep-Learning-Based Road-Prediction for Augmented Reality Navigation Systems at Night. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, 2016.

[28] Ming-Yu Liu and Oncel Tuzel. Coupled Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*. 2016.

[29] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[30] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class Generative Adversarial Networks with the L2 Loss Function. Technical report, 2016. arXiv:1611.04076.

[31] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context Encoders: Feature Learning by Inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.

[33] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. Technical report, 2016. arXiv:1606.03498.

[34] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In *International Conference on Learning Representations*, 2014. arXiv:1312.6229.

[35] Yichang Shih, Sylvain Paris, Frédo Durand, and William T. Freeman. Data-driven Hallucination of Different Times of Day from a Single Outdoor Photo. *ACM Transactions on Graphics*, 2013.

[36] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2014. arXiv:1509.01951.

[37] Leslie N. Smith and Nicholay Topin. Deep Convolutional Neural Network Design Patterns. Technical report, 2016. arXiv:1611.00847.

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[39] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. Technical report, 2016. arXiv:1602.07261.

[40] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised Cross-Domain Image Generation. Technical report, 2016. arXiv:1611.02200.

[41] Markus Thom and Franz Gritschneder. Rapid Exact Signal Scanning With Deep Convolutional Neural Networks. *IEEE Transactions on Signal Processing*, 2017.

[42] Fabian Tschopp. Efficient Convolutional Neural Networks for Pixelwise Classification on Heterogeneous Hardware Systems. Technical report, 2015. arXiv:1509.03371.

[43] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Proceedings of the International Conference on Machine Learning*, 2016.

[44] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. Technical report, 2016. arXiv:1610.07584.

[45] Saining Xie and Zhuowen Tu. Holistically-Nested Edge Detection. *International Journal of Computer Vision*, 2017.

[46] Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *International Conference on Learning Representations*, 2016. arXiv:1511.07122.

[47] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful Image Colorization. *Proceedings of the European Conference on Computer Vision*, 2016. arXiv:1603.08511.

[48] Junbo Zhao, Michaël Mathieu, and Yann LeCun. Energy-based Generative Adversarial Network. Technical report, 2016. arXiv:1609.03126.

[49] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of the European Conference on Computer Vision*, 2016.

[50] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. Technical report, 2017. arXiv:1703.10593.