Colorization of Black & White Movies with Convolutional Neural Networks

Martin Rejmon
ČVUT - FIT
rejmomar@fit.cvut.cz

December 12, 2019

1 Introduction

The problem my semestral work examines is one that everybody can understand – neural network takes as an input an old black & white movie and produces as an output a video with its colours restored in such a way, as to make them seem as real and lifelike as possible. What is interesting, is that any real progress in this problem has only been made quite recently, with the invention of convolutional neural networks (CNNs) enabling the artificial intelligence to grasp the spatial and temporal dependencies in more than one dimensional inputs (for example videos/images).

More personally, my task (as written on GoogleDocs) was to download a specific CNN [1], already pretrained on colorizing single images, and to repurpose it to colorize whole videos (probably by predicting each individual frame) and also to attempt to fix temporal defects, that is to keep the colors between frames as consistent as possible. As both the assignment and the assigned CNN were slightly old, I was advised to first conduct some research into modern advancements in automatic image colorization.

2 Research

I have arbitrarily narrowed my research only on papers (or projects) which covered automatic colorization of black & white images (or videos) – the word automatic is important here, as that means no reference/example images or color scribbles are provided to the network.

What I found out, is that virtually all of the relevant papers (except the oldest two [2],[3]) [4],[5],[6],[7],[8],[9] (and more) make use of the aforementioned CNNs. The second most important milestone in automatic colorization is the usage of Generative Adversial Networks (GANs) in supervised learning, as shown by Isola et al. [10]. A lion's share of the newer papers use some variations of these in their architecture [11],[12],[13],[14],[15],[16],[17],[18],[19]. Other constructs that deserve a mention are Variational Auto Encoders [20],[21], Deep Autoregressive Models [22],[23],[24] (namely PixelCNN) [25], Capsule Networks [26], something called Invertible Networks [27] and many others.

I took a closer look at three interesting networks, that also had official implementations available.

3 Three networks

First network by Zhang et al. [1] from 2016 (the one in the original assignment) is popularly used in papers as a baseline. It is a deep convolutional net trained from scratch. I will actually dissect it more in the second part of the report, as the paper I have decided to implement is a modification of this one.

As a matter of fact, the second network does not come from a paper, but from the most popular GitHub repository focused on automatic colorization of images (and videos) – Deoldify [28] by Jason Antec. It is a Self-Attention based GAN based on this paper [29], except the generator is an U-net, modified to have self-attention and spectral normalization, among other things. U-net [30] is a popular pretrained deep convolutional network designed for image segmentation.

The third network comes from a recent paper by Chenyang et al. [31], which is coincidentally also the only one I could find that covered automatic video colorization, instead of just image colorization. The architecture is split into two separate networks – the colorization network, which individually colorizes each frame of the video and the refinement network, which takes as input two consecutive colorized frames and two consecutive "confidence maps" (which to my best understanding have something to do with optical flow (and optical flow tries to estimate (in vectors) where points corresponding to the same image objects (based upon their brightness) moved between two images)) and outputs "refined" frames.

They have also adopted U-net, for both their colorization and refinement networks, and modified it to best suit their architectural needs. Unfortunately, I did not manage to make the refinement network work as intended. I discovered the pretrained weights for it hidden in a GitHub issue, but after laboriously getting it to run with minimal instructions, it produces only corrupted output. However, I found a paper with a similar premise, upon which I will elaborate shortly.

4 Temporal consistency

But first, it ought to be mentioned that I also skimmed through several papers focused on networks that consume video, but are not strictly concerned with colorization, such as video synthesis [32] / video style transfer [33] and attempted to figure out how they manage their temporal relationships. It seems, that one of the more popular ways is computing optical flow between consecutive frames and using it as an input somewhere in the architecture, or similar. Optical flow is very computationally expensive, however somewhat recently several papers, such as Flownet [34], Flownet2 [35] and pwc-net [36], have shown that deep convolutional networks give similar or better results than traditional methods, but more importantly, are orders of magnitude faster.

Following that, I found a paper by [37], that strives to improve the overall results of attempting to use any of the image-based networks on each frame of a video. It is a recurrent convolutional network, which also makes use of optical flow. Fortunately, no flow is required during predictions, which makes it quite fast to use, and as such it allowed me to slightly reduce temporal inconsistencies during testing of the previously mentioned networks.

5 Comparison

I have downloaded the DAVIS 2016 dataset [38] for video object segmentation, which is made up of about fifty 2-4 second long sequences at 24fps. The given pretrained models for the three colorization networks were trained on the well-known ImageNet dataset, however the third one was also trained on this particular DAVIS dataset, which gives it a slight advantage. I have run the networks on few sequences from the validation set – selected frames can be seen in Figure 1 and link to the full videos is included in the README file.

As can be seen, the first network by Zhang et al. provides quite well saturated colors, but has unpleasant problems with edge bleeding and overall spatial consistency. The GAN-based network – DeOldify – provides overall the best performance, even in more difficult situations, although it still has few blemishes, for example the "forcefield" around the paraglider, and struggles quite heavily with temporal consistency. Last but not least, the network by Chenyang et al. outputs very desatured colors, with the most extreme example being the car one, but perhaps because of it has good spatial consistency and excellent temporal consistency. The desaturated colors could also be caused by the fact, that I failed to run the refinement part of their network – thus they are essentially competing with only half of their model.

6 Augmenting colorization by jointly learning image segmentation

Because a lot of papers are missing official or unofficial implementations and I also wanted to have some code to show off as a part of my semestral work, I naively decided to implement a paper by Zhao et al. [39]. It is smaller in scope than most other papers I have researched, building upon the work of Zhang et al. [1] (from now on called the the original network).

Both of these papers concern itself with deep convolutional networks, and their architectures can be seen stacked vertically in Figures 2 and 3. At a first glance there is an entire additional branch which outputs per-pixel image segmentation. The resulting segmentation mask is suprisingly not used anywhere else as an input, it is there only for the purposes of computing segmentation loss, which is then added up with the colorization loss and minimized. Authors reason that this helps the neural network learn what the object is and thus what its colors should be. The second smaller addition is that after the neural net produces its downsampled output during inference, it is not subjected to a normal upsampling algorithm, but to a more sophisticated Joint Bilateral Upsampling algorithm, which authors claim works better in preserving edge colors.

7 Color space and probabilities

Before being fed into the network, images are converted to the CIE *Lab* color space, where L represents luminance aka brightness aka input and the other two channels – a and b – take care of chrominance aka color aka output.

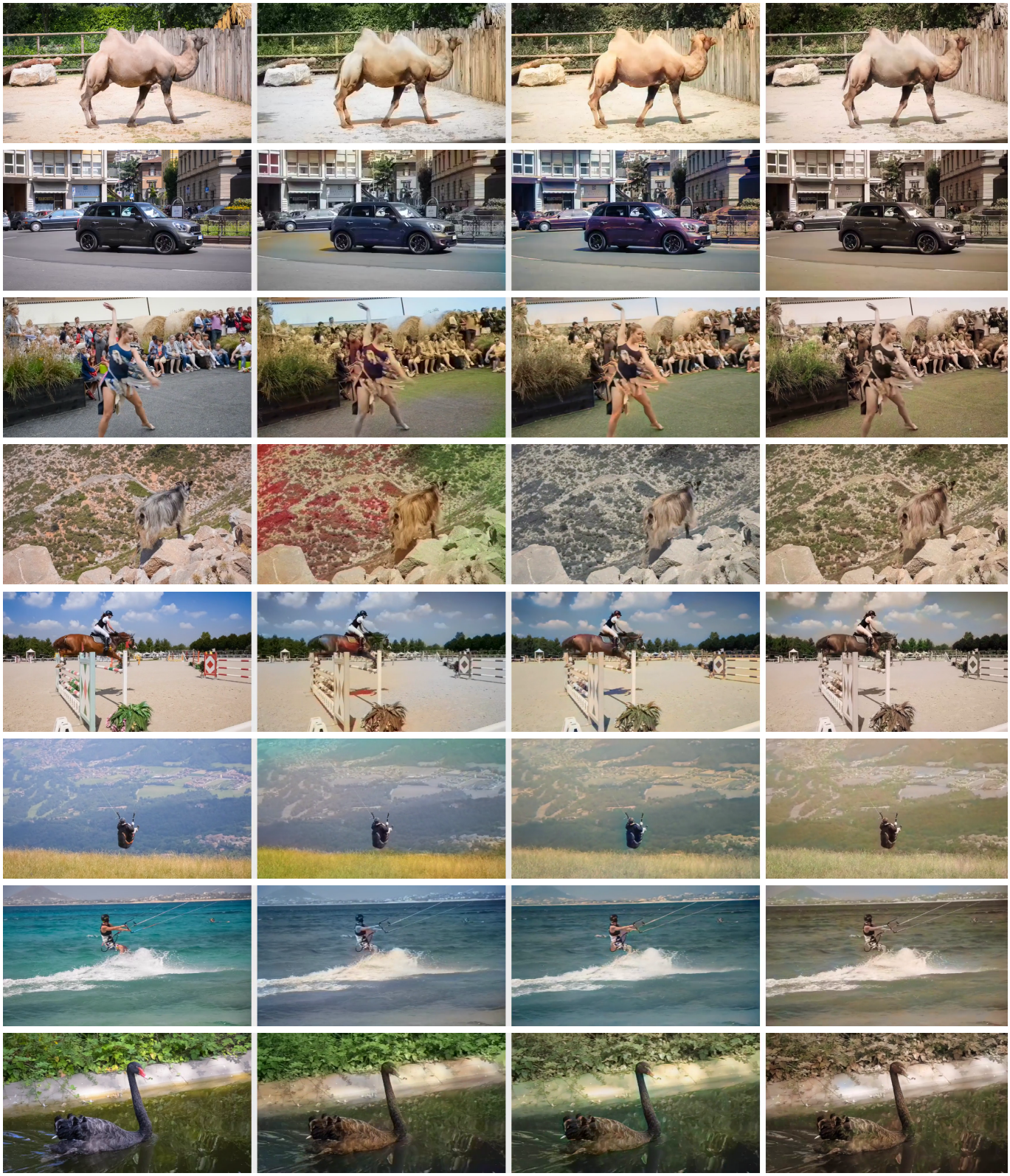

Figure 1: Selected frames from comparison videos of the three networks - first column is ground truth, second one is Zhang et al. [1], third one is DeOldify [28] and fourth one is Chenyang et al. [31].

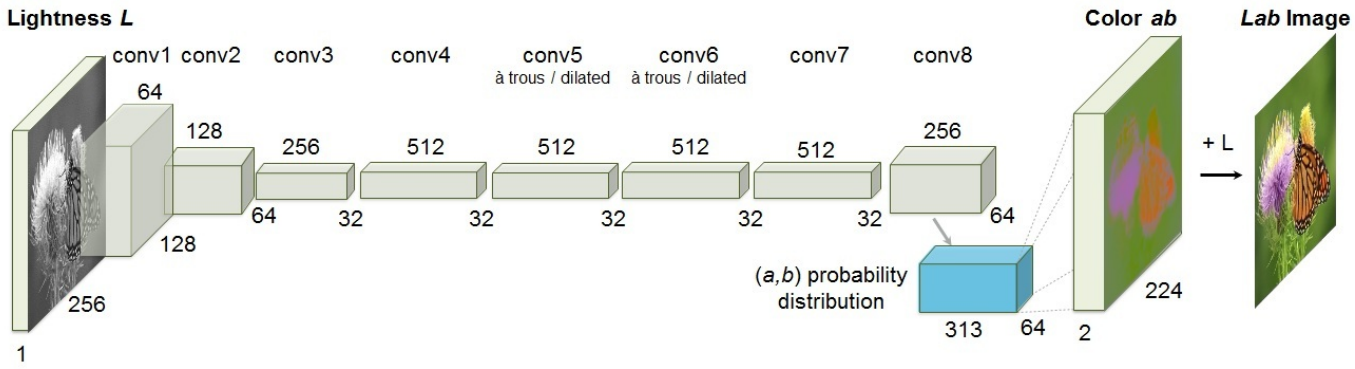

Figure 2: Architecture of the original network by Zhang et al. [1], as presented by them. The numbers above blocks represent number of filters and the numbers below spatial dimensions.

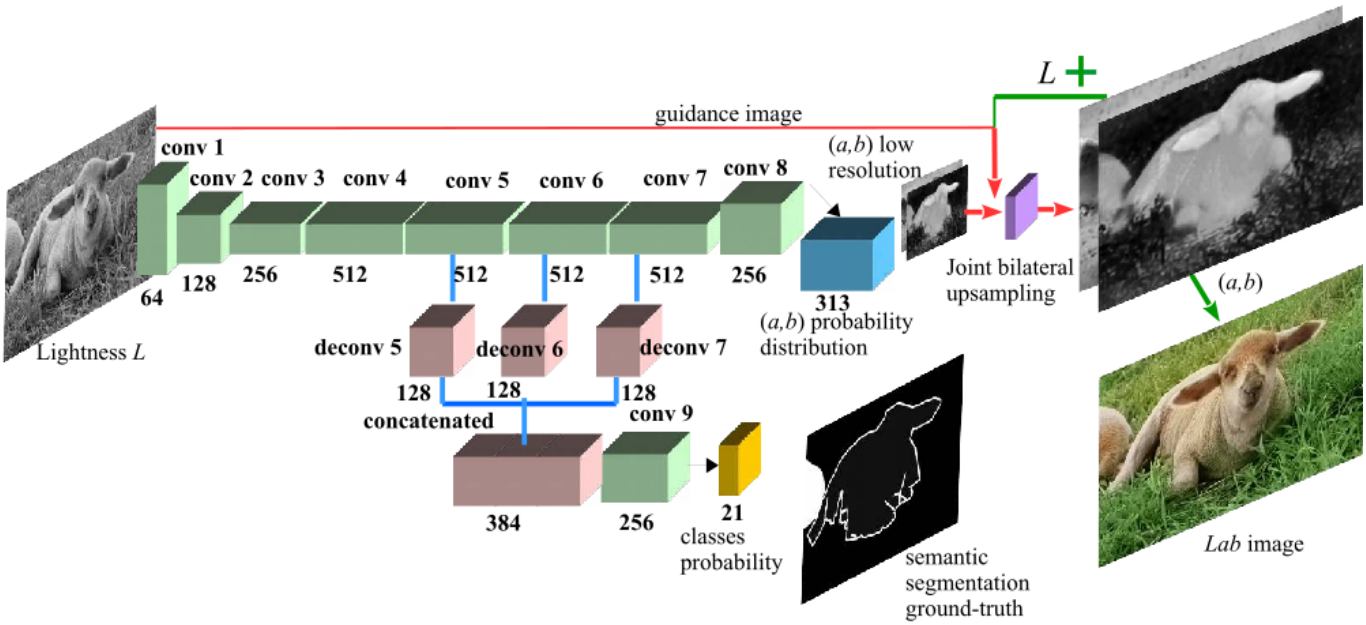

Figure 3: Architecture of the expanded network by Zhao et al. [39], as presented by them.

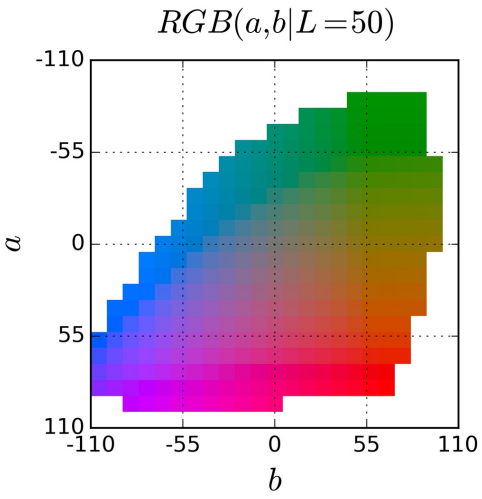

Figure 4: The ab color space discretized into 313 bins, as presented by [1].

The a and b channels are discretized into 313 bins, as can be seen in Figure 4, and the network outputs a corresponding vector of 313 probabilities for each pixel in the image. The ground truth values are one-hot encoded but with smoothed labels, with the catch that the position of values in the color space is considered (for example target is 0.9, its neighboring colors 0.05 and others 0).

The authors have also shown that the more desatured values are an order of magnitude more common than the satured values (in the full ImageNet dataset), due to the overabundance of things such as dirt, clouds, walls, pavement and others. To avoid these values dominating the loss function, each pixel is reweighted at train time based on the rarity of its color.

The image segmentation part is similar except simpler. There are only 21 probabilities for each pixel – the net is trained on the PASCAL VOC2012 [40] dataset, which includes segmentation masks for 21 classes (+ border). The dataset has around 3000 images, split 50:50 into a train and validation set, which I have personally reshuffled to make ratio the around 85:15, to increase the size of the training data. There is also a test set available, but it has no segmentation masks. There is no label smoothing and and the `border` class is ignored (encoded as all zeroes). The losses used for both problems were cross-entropy.

8 Obstacle(s)

A slight problem was, that the authors did not describe their architecture in great detail, nor could I find any appendix or supplement containing such information. What was missing was:

- from what layers were the blocks in Figure 3 composed,
- what were their parameters (other than number of filters),
- how were their weights initalized,
- what other training parameters have they used – learning rate, optimizer/solver, etc.

However, the network which they were expanding upon was described quite satisfactorily in the original paper. Each of the `conv` blocks in Figure 2 is made up from two-to-three convolution-relu layers, followed by a batch normalization layer. There are no pooling layers, instead the downsampling is accomplished by bigger strides . The weights were initialized using k-means [41] and they used the Adam optimalizer, with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $weight\_decay = 10^{-3}$. Learning rate was set at 3×10^{-5}, and was dropped to 10^{-5} and 3×10^{-6} when loss plateaued.

This still left me in the dark concerning the image segmentation branch – nonetheless I have taken the liberty of making the `deconv5-7` blocks basically the same as the `conv5-7` blocks, except with deconvolution instead of convolution layers, and similarly `conv9` is just a copied `conv8` block. The weights were set by taking values from a normal distribution with zero centered mean. I implemented the network in Tensorflow 2.0 with the Keras Functional API. While implementing I have taken some inspiration from the official implementation [42] of the original network in Caffe and also from an unofficial implementation [43] of the same network in Keras – the latter of which being decidedly more useful, as custom layers in Caffe are written in C++ and it's quite nontrivial to grasp their meaning.

9 Results

In the end, my endeavor in implementing this paper unfortunately does not bear much fruit, as after adding the image segmentation branch, the network diverges (both losses continue to increase) either immediately when the training starts or after few epochs. As for the reasons why:

- I have skipped the implementation of the class rebalancing for image segmentation (I did not find it necessary as the only overrepresented class was `background`) and of the Joint-Bilateral-Upsampling filter, however the former should not have such a massive effect and the latter should have none (as it is only used during inference).

- I have very probably gotten the structure (or parameters) of the image segmentation branch wrong, as authors claim that the size of the model of the original network is 128.9 Mb (which almost matches my Keras version of 129.1Mb) and that the size of the expanded network is 147.5 Mb (while my Keras version is 150.8 Mb).

- Furthermore, authors claim that they trained only for 40 epochs on the PASCAL VOC2012 dataset (which has around 1500 train images) and 20 epochs on the Coco-stuff dataset [44] (which has around 9000 images), therefore the network was trained on around 240 000 images – which seems quite low to me. For comparison the original network was trained in 450 000 iterations, which is, with mini-batches of size 40, around 18 000 000 images. This makes me think authors utilized some kind of transfer learning on the expanded network, although they lay claim to no such thing.

- Last but not least, there is a non-zero probability that there is an unwelcome bug in the implementation, mainly in the target image/segmentation preprocessing code, or that I have misunderstood some techniques or statements from the papers.

The code for the implementation is included in the jupyter notebook in the repository - though in its current state it probably does not have many other uses other than for the purposes of this semestral work.

References

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

[2] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. *CoRR*, abs/1605.00075, 2016.

[3] Aditya Deshpande, Jason Rock, and David Forsyth. Learning large-scale automatic image colorization. In *ICCV*, 2015.

[4] Xiangguo Liang, Zhuo Su, Yiqi Xiao, Jiaming Guo, and Xiaonnan Luo. Deep patch-wise colorization model for grayscale images. In *SIGGRAPH ASIA 2016 Technical Briefs*, SA '16, pages 13:1–13:4, New York, NY, USA, 2016. ACM.

[5] Ryan Dahl. Automatic colorization. `https://tinyclouds.org/colorize/`, 2015.

[6] 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. *ACM Trans. Graph.*, 35(4):110:1–110:11, July 2016.

[7] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. *CoRR*, abs/1603.06668, 2016.

[8] Federico Baldassarre, Diego González Morín, and Lucas Rodés-Guirao. Deep koalarization: Image colorization using cnns and inception-resnet-v2. *CoRR*, abs/1712.03400, 2017.

[9] Venkataraman Santhanam, Vlad I. Morariu, and Larry S. Davis. Generalized deep image to image regression. *CoRR*, abs/1612.03268, 2016.

[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] Stephen Koo. Automatic colorization with deep convolutional generative adversarial networks. 2016.

[12] Yun Cao, Zhiming Zhou, Weinan Zhang, and Yong Yu. Unsupervised diverse colorization via generative adversarial networks. *CoRR*, abs/1702.06674, 2017.

[13] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. *CoRR*, abs/1711.11586, 2017.

[14] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *CoRR*, abs/1711.11585, 2017.

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

[16] Shirsendu Sukanta Halder, Kanjar De, and Partha Pratim Roy. Perceptual conditional generative adversarial networks for end-to-end

image colourization. *CoRR*, abs/1811.10801, 2018.

[17] Patricia Vitoria, Lara Raad, and Coloma Ballester. Chromagan: An adversarial approach for picture colorization. *CoRR*, abs/1907.09837, 2019.

[18] Seungjoo Yoo, Hyojin Bahng, Sunghyo Chung, Junsoo Lee, Jaehyuk Chang, and Jaegul Choo. Coloring with limited data: Few-shot colorization via memory-augmented networks. *CoRR*, abs/1906.11888, 2019.

[19] Manoj Sharma, Megh Makwana, Avinash Upadhyay, Ajay Pratap Singh, Anuj Badhwar, Akkshita Trivedi, Anil Saini, and Santanu Chaudhury. Robust image colorization using self attention based progressive generative adversarial network. In *CVPR 2019*, 2019.

[20] Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, and David A. Forsyth. Learning diverse image colorization. *CoRR*, abs/1612.01958, 2016.

[21] Safa Messaoud, David A. Forsyth, and Alexander G. Schwing. Structural consistency and controllability for diverse colorization. *CoRR*, abs/1809.02129, 2018.

[22] Amelie Royer, Alexander Kolesnikov, and Christoph H. Lampert. Probabilistic image colorization. *CoRR*, abs/1705.04258, 2017.

[23] Sergio Guadarrama, Ryan Dahl, David Bieber, Mohammad Norouzi, Jonathon Shlens, and Kevin Murphy. Pixcolor: Pixel recursive colorization. *CoRR*, abs/1705.07208, 2017.

[24] Jiaojiao Zhao, Jungong Han, Ling Shao, and Cees G. M. Snoek. Pixelated semantic colorization. *CoRR*, abs/1901.10889, 2019.

[25] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016.

[26] Gokhan Ozbulak. Image colorization by capsule networks. In *CVPR 2019*, 2019.

[27] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *CoRR*, abs/1907.02392, 2019.

[28] Jason Antic. Deoldify. `https://github.com/jantic/DeOldify`, 2018.

[29] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *ArXiv*, abs/1805.08318, 2018.

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

[31] Chenyang Lei and Qifeng Chen. Fully automatic video colorization with self-regularization and diversity. In *CVPR*, 2019.

[32] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *CoRR*, abs/1808.06601, 2018.

[33] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. Coherent online video style transfer. *CoRR*, abs/1703.09211, 2017.

[34] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015.

[35] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *CoRR*, abs/1612.01925, 2016.

[36] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. *CoRR*, abs/1709.02371, 2017.

[37] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. *CoRR*, abs/1808.00449, 2018.

[38] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.

[39] Jiaojiao Zhao, Li Liu, Cees G. M. Snoek, Jungong Han, and Ling Shao. Pixel-level semantics guided image colorization. *CoRR*, abs/1808.01597, 2018.

[40] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[41] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

[42] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization (official caffe implementation). `https://github.com/richzhang/colorization/tree/master/colorization`, 2016.

[43] foamliu. Colorful image colorization (unofficial keras implementation). `https://github.com/foamliu/Colorful-Image-Colorization`, 2018.

[44] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.