Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

# Meta-Learning an Image Editing Style

Bachelor's Thesis


Jérémy Scheurer

jeremys@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich


**Supervisors:**
Oliver Richter, Gino Brunner
Prof. Dr. Roger Wattenhofer

February 28, 2019

# Acknowledgements

I would like to thank my advisors Oliver Richter and Gino Brunner for their insightful help and assistance during my Bachelor-Thesis. When losing myself in some dark hole of programming, they were always able to give me back some perspective and direction. I would also like to thank Yu-Sheng Chen, the author of [1], for providing previously unpublished code and answering questions about it. Then also thanks to my friends Claudio Ferrari and David Yenicelik for reviewing my work. Lastly, I would like to thank Prof. Dr. Roger Wattenhofer and his Institute for giving me the freedom and opportunity to choose my own research topic and follow through with it.

# Abstract

Most of the existing work which tries to learn an image editing style falls into the categories of learning in a paired or unpaired manner. The former method learns on pairs of unedited and edited images and the latter learns a style just from edited images. But almost all existing approaches, paired or unpaired, use thousands of images to learn an editing style. Our work tries to address this data inefficiency by using Meta-Learning. To this end, we evaluate the effectiveness of different Meta-Learning algorithms on the problem of learning an image editing style from a paired image database in a supervised manner. Specifically, we use an adapted U-Net architecture [1] to train on image pairs. We then combine it firstly with Model-Agnostic Meta-Learning [2] and secondly with an adaptation of $RL^2$ [3] to the supervised domain, which we call $SL^2$. None of the Meta-Learning algorithms improves the results compared to a standard optimization with Adam [4]. Nevertheless, we provide quantitative and visual evidence showing that learning a prior image-editing style with Deep Learning models, let's us learn a new editing style with very few images. We also show that using a prior image-editing style improves the results of training on a specific expert with a larger data set.

# Contents

Chapter 1

# Introduction

## 1.1 Why Image Editing?

*"Use a picture. It's worth a thousand words."* This idiom appeared in a newspaper in 1911 [5] and is still true today. As digital cameras are becoming more ubiquitous due to decreasing costs, and most smartphones are equipped with high-quality lenses, the accessibility to photography has increased. This can also be seen in the rising popularity of online image sharing platforms such as Instagram, Flickr, and Pinterest. But although the integration of cameras into smartphones has improved the user-friendliness and accessibility of photography, the art of taking a good photo is still a difficult process. Today this process most often does not stop after taking the image, but continues into using image editing tools to improve the taken picture. There are several reasons why people use such techniques. Digital cameras take incomplete and noisy samples of a scene, to reconstruct a high-quality image. Also, cameras respond linearly to the incoming light, thus they can only approximate the non-linear transformation our eyes are able to perform. Not only that, but cameras are limited in resolution, dynamic range, field of view etc. which leads to even more lossy representations of reality onto images. These are the reasons why people turn to image enhancement methods, which try to alleviate the above-mentioned problems. This can be done by changing the color rendition, the sharpness, saturation, contrast, luminance etc. But even in the case of having an image with a perfect rendition of reality, one might still want to edit the image in certain ways. Be it to highlight certain features of the photo, to create a certain mood, to make certain colors pop out or just to make the image more subjectively pleasing. Thus image editing is a critical aspect of a photographer's workflow, which offers great power by enabling her to fix and improve a photo.

## 1.2 Problems of Image Editing

Unfortunately, the process of editing an image is very tedious, it requires a lot of time and expertise. There is a myriad of parameters that one can adjust to improve an image, each one of them influences the image in a specific, sometimes complex way. Using different adjustments at the same time might lead to some unexpected results due to side-effects. Although there exist countless books and tutorials on good practices, editing an image is not straight forward, as the photographer might need to trade-off different objectives. All this leads to a situation in which the photographer needs a large knowledge of the image editing tools and how they affect an image. It follows that automated photo adjusting packages would greatly facilitate the process of image editing and make it more available to everyday consumers. Firstly by reducing the time spent on editing an image and secondly by removing the need for expert knowledge of the tools.

## 1.3 Insufficient Commercial Approaches

Today there exist several image editing software packages that offer automated editing tools, although very limited ones. Some of them provide "global auto-edit" tools which usually try to improve the general image quality by using some rules that are based on heuristics. Some example rules are histogram stretching, equalization and according to Bychovsky et.al. [6], simple methods such as fixing the black and white points of the image to the darkest and brightest pixels. The most prominent "global auto-edit" tools are probably provided by Adobe Lightroom and Apple Aperture (discontinued).
These packages usually offer other rule-based features that in popular language have become known as "filters". Instead of making adjustments based on some general heuristics, a photographer can manually edit an image and save the parameters she used. The person editing the image can then apply these same parameters to other images.
Unfortunately, both above-mentioned techniques have big shortcomings. The "global auto-edit" techniques pose the problem, that the results are not personalized to a specific user. That is, because general heuristics are used to improve an image. These cannot factor in personalization, as the possibilities of adjusting parameters are countless and oftentimes require trade-offs in different parts of an image. But the resulting edited photo should be pleasing to the intended audience (e.g the photographer) and not some general user, i.e. the decision if an image looks good is subjective. The importance of personalizing the image editing process was also shown and discussed in [7], [8], [9]. Then, in both cases, rule-based techniques are used, i.e. the adjustments to the pixels are not really dependent on the image itself. That means that if one increases the brightness in one image and applies the setting to another photo, the result could be an

image that is too bright. At best, some global-auto edit tool might take into account certain properties of the input, nevertheless, the edit will still be based on some basic pre-programmed rules. Filters do indeed take the importance of personalization into account. Unfortunately, a user first has to find these pleasing parameters, which we have declared to be a difficult task.

## 1.4 Challenges and Contributions

The solution to the above-mentioned problems is to learn an image editing style with model and feature free methods. These firstly alleviate the problem of rule-based approaches, because they are able to learn non-linear functions which adjust parameters according to a given image. That means that they are for example able to brighten up one darker image, but also realize that another image might already be bright enough and thus doesn't need much more adjustments. Secondly, the methodologies we work with train a model on a paired image dataset of edited and unedited images. Assuming that one has these before and after images, the model learns a style automatically without requiring an inexperienced user to find some optimal adjustment settings. There are also other approaches to this problem that only require a dataset of edited images (unpaired problem setting). But almost all previous work that tries to learn an image editing style (paired or unpaired), relies on very large datasets. This data inefficiency drastically reduces the practicability of learning a new image editing style. Because for all practical purposes, a dataset of several thousand images is usually not available. In this work, we attempt to solve this data inefficiency by applying Meta-Learning algorithms to the problem of learning an image editing style. For our architecture, we use the Deep Photo Enhancer (DPE) network from Chen [1]. We will make use of the Adobe 5K dataset [6] which contains images that are edited by 5 different photographers. Our goal is to combine DPE with different Meta-Learning algorithms which should solve the data inefficiency problem. We train our Meta-Learning algorithms on 4 experts to then be able to learn from very few images of the fifth expert. Concretely we make the following contributions:

1. We compare different Meta-Learning algorithms on the task of learning an image editing style.

2. We give analytic and visual evidence that the results of training a Deep Learning model on a large dataset of a specific expert, can be improved by learning a prior image editing style from different experts.

Chapter 2

# Related Work

There is a lot of work attempting to tackle the problem of improving the subjective quality of an image. The techniques to learn such a style can be divided into two approaches, the ones who use actions or features to predict an improved image and the others who do image-to-image translations. Predicting each individual pixel, on one hand, makes the task more difficult than predicting a fixed set of actions or features (which might have substantially lower dimensions than a whole image). On the other hand, it yields a higher model capacity to express arbitrary image adjustments. Conversely, the first method might be easier to train and provide good results, but at the same time it will always be restricted by the actions and features it uses to produce an improved image.

*Feature/Action based Methods*: In most of the early work, hand-crafted features inspired by photographic practice are used to improve an image. In [7] for example an end-to-end pipeline for automated image editing is proposed. With a simple interface, a user is able to edit images according to her personalized style. To reduce the number of images a user has to edit, the interface proposes images which yield a maximal information gain about the editing style by using [10]. This can already be viewed as a first attempt at solving the data-inefficiency problem of learning an editing style. When the learned editing style has to be applied to a new image, the most similar image of the already edited photos is chosen according to some learned similarity metric. The editing style of this photo is then applied to the new image. In a user study, they were able to conclude that users indeed prefer a personalization of image editing operations. Similarly [11] is also able to learn from very few images, but again the user has to manually manipulate an image to create a dataset to learn from.
Another early approach is proposed by Bychkovsky et al. [6]. To this end, they introduce an image data set of 5000 RAW photos which are then edited by 5 professional photographers. This allows them to then train a Gaussian Process Regression (GPR) [12] model in a supervised manner. Note however that they again do not model an image-to-image regression, they train on features inspired by photographic practice and try to predict the luminance (which is important to the editing style) of an image. They then show that they are able to learn

from few (0-30) images of an expert, by pre-training a covariance function on a larger (4K) dataset of that same expert or also of another expert. They call this second option transfer-learning and show that it indeed improves the results. In our work, we will make use of this very idea when applying Meta-Learning.
In a similar fashion [13], [14], [15], [16] all use features to learn an image editing style, with [15],[16] making use of a ranking approach. More recent work often makes use of Deep Learning methods, for example [9],[17], [18] with the last two using Reinforcement Learning algorithms. Furthermore, many approaches who work on features or actions, are able to learn an image enhancement style with unpaired images. That means they do not need a supervised dataset of images before and after editing them, but only the latter. [19],[20],[21], [22],[23] all belong to this category, with [19], [20] using Generative Adversarial Networks (GAN's) [24] and [20],[23] using Reinforcement Learning methods. These last approaches all belong to the category of predicting certain actions and the order in which to apply them, in order to reproduce an editing style.

*Image-to-Image Translation based Methods*: All approaches belonging to this category that we know of, make use of Deep Learning methods. [25],[26] and [27] use GAN's to show that image to image translation works for very different domains such as image colorization, map to satellite image translation etc. In [28] and [29] more general style transfer approaches are presented, for example to transform daytime images to nighttime images, switch out colors and surface patterns etc. Closely related, [30],[31],[32],[33],[34],[35] all use image-to-image translations for specific applications, with the last two making use of aesthetically labeled images. Other methods such as [36],[37] make use of GAN's to enhance low-quality images to look like DSLR camera images. Lastly, there is Chen et. al [1] who uses an adapted U-Net [32] architecture to show impressive results and beat previous methods when training in a supervised fashion on 2250 images. Their main contribution to the U-Net is the introduction of global features. Global features have already been explored in [30], however there they need to train a supervised network with explicit scene labels in the dataset. Here they directly use the U-Net to learn an implicit, global feature vector. Furthermore, they extend their architecture into a GAN to further provide very similar results by training in an unpaired fashion on a little more than 600 images. Note that we will use their Network Architecture and apply Meta-Learning to it.

Of all the aforementioned work, almost all ([18], [17], [16], [21], [23], [15], [22], [20], [19], [34], [30], [35], [1], [36], [37], [33], [31]) use approximately thousand to several thousand images to train their paired or unpaired models. The approaches not requiring a lot of data are either not specifically optimized for learning an image editing style or belong to the category of action/feature based methods. The data inefficiency apparent in image-to-image based methods is what our work tries to address. In some sense, we are inspired by Bychkovsky et al. [6], but try

to work directly in the pixel space instead of using handcrafted features. Also, we use Deep Learning Methods [1] instead of model-based approaches such as a GPR.

Chapter 3

# Model

## 3.1 Model Architecture

Our model architecture is taken from Chen et al. [1] (see Figure 3.1), who extended on the U-Net architecture [32] and call their model Deep Photo Enhancer. The U-Net architecture was initially proposed for bio-medical image segmentation but showed great results on other tasks as well. It is an architecture using Convolutional Neural networks to do pixel classification or in our case an image-to-image regression. Their architecture is optimized to work with few data points and relies on strong data augmentation. The U-Net consists of a contracting network which is used to capture context and an expanding network which yields accurate localization on the output. Initially, the U-Net was based on the idea of the Fully-Convolutional network [33]. However, [32] modified this architecture by adding successive layers to the contracting network and replacing their pooling operators with upsampling operators. This replacement generally increases the resolution of the output. To provide localization, they also combine the high-resolution features of the contracting network with the upsampled output. In order to let context information flow to the higher resolution levels, the U-Net further provides a large number of feature channels in the upsampling part. This

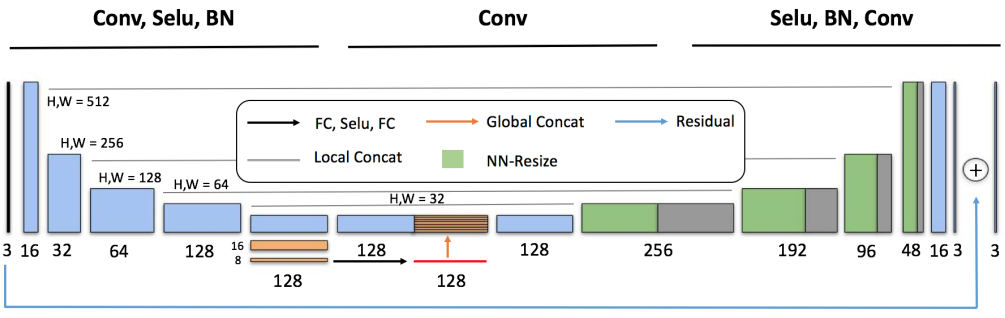



Figure 3.1: Our Model Architecture is used to train on a supervised dataset of edited and unedited image pairs. The red block marks the global features.

results in the contracting and expanding part being nearly symmetric and thus providing a U shaped architecture. The main extension of Chen et al. [1] to the U-Net, is the introduction of so-called global features. The idea here is that high-level context information about a scene, such as the lighting-condition, scene category or subject type is very relevant for our visual system to be able to adjust to it. This is also true for cameras, which have specific settings that have to be adapted to the current scene. Thus the conjecture is that these global features are relevant for output pixels to decide on their local value.
The contraction network of our architecture consists of a 5x5 filter with a stride of 2 which is then followed by a SELU [38] activation and batch normalization [39]. To make the architecture more efficient, the global feature extraction shares the contracting network of the U-Net with the extraction of local features for the first 5 layers. After the $32 \times 32 \times 128$ feature map of the fifth layer, we start extracting the global features. This is done by using the contraction steps to further reduce the feature map to $16 \times 16 \times 128$ and $8 \times 8 \times 128$. We then use a fully-connected layer with a SELU activation function, followed by another fully-connected layer to produce a $1 \times 1 \times 128$ feature map which are our aforementioned global features. We also make use of a Residual layer, i.e. by adding the input image to our predictions we are learning the difference between an unedited and edited image instead of having to predict an image directly.
Given a label $Y$, i.e. an edited image and a prediction $\tilde{Y}$ we optimize for the following loss function:

$$\arg\min_{\text{Model M}} \mathbb{E}[log_{10}(MSE(\tilde{Y}, Y))]$$

where

$$\text{MSE(,Y)} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} ||\tilde{Y}(i,j) - Y(i,j)||^2$$

and

$$\tilde{Y} = Model_M(i,j)$$

with m and n being the height and the width of the images. As the logarithm is a convex function, technically we are just optimizing for the mean-squared error (MSE). However, we need the logarithm of the MSE to calculate the Peak-Signal-to-Noise ratio, which we will show later.

## 3.2 Meta-Learning Models

### 3.2.1 MAML & FOMAML

In our work, we use Meta-Learning to address the data inefficiency of previous approaches. Firstly we use an algorithm called Model-Agnostic Meta-Learning (MAML) [2] which is completely model-free, i.e. it can be used on any model

**Algorithm 1:** MAML

**Input:** $p(\mathcal{T})$ distribution over tasks
**Input:** $\alpha$,$\beta$ step size hyperparameter
1 randomly initialize parameters $\theta$
2 **while** *not done* **do**
3  sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4  **forall** $\mathcal{T}_i$ **do**
5  evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K samples
6  Compute adapted parameters with gradient descent:
 $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7  **end**
8  Update $\theta \leftarrow \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9 **end**

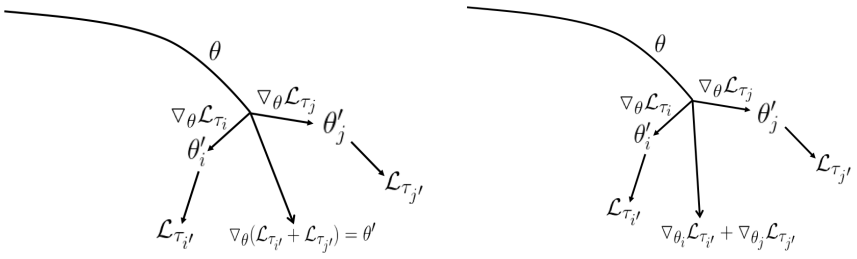
that is trainable by gradient descent. Thus it can be used for different Machine Learning paradigms such as a Regression, Classification and even Reinforcement Learning. On a high-level, its goal is to train a model on several individual tasks in order to learn a "good" initialization of the weights. "Good" means that we are then able to train our model on a new, similar task with very few data and get good results. In their work, they show that they are able to achieve state of the art results on two few-shot image classification benchmarks.

The Meta-Learning problem setting consists of a task distribution $p(\mathcal{T})$ that we want our model $f$ to be able to adapt to. In a K-shot learning setting the goal is to learn a new task $\mathcal{T}_i$ drawn from $p(\mathcal{T})$ from only K samples drawn from the data distribution of that task. First, we train on K samples from a specific task for a defined loss-function $\mathcal{L}_{\mathcal{T}_i}$ using Stochastic Gradient Descent (SGD). With the optimized parameters, we now evaluate newly drawn data from the data distribution of the same task on the loss function. We then optimize our original parameters based on how well we did on the evaluation loss across all tasks (see Algorithm 1). So finding optimal parameters requires the model to be able to quickly adapt to new tasks.

Unfortunately MAML is computationally very expensive due to the fact, that one needs to calculate second order derivatives when propagating the meta-gradient through the gradient operator in the meta objective (see Line 8 of Algorithm 1). [2] thus proposes a first order approximation which is also discussed in [40] and called FOMAML. The only thing that changes in algorithm 1 is line 8 which becomes:

$$\theta \leftarrow \beta \nabla_{\theta_i'} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

This effectively means that for each individual task we calculate the meta-




(a) MAML (b) FOMAML

Figure 3.2: A comparison of MAML and FOMAML

gradient with respect to the already optimized weights $\theta'_i$ of that specific task and not with respect to the original weights $\theta$(see Figure 3.2 for an illustration).

[2] have shown that using FOMAML yields nearly the same results and provides a 33% performance speedup. Additionally, it is also much simpler to implement [40]. These findings suggest that the advantages MAML provides, come from the task-specific optimization and not form the second order derivatives.

In our problem setting we have a dataset with original images that have been edited by several photographers. The goal in our case will be to train our Meta-Learning algorithm on several experts, such that when confronted with images of a new expert, the model is able to learn the editing style quickly. That means in our case, each expert is a task.

### 3.2.2 Supervised Learning$^2$ ($SL^2$)

Our second Meta-Learning algorithm is based on the work of [41] which has also been explored in the Reinforcement Learning domain by [3] ($RL^2$: Fast reinforcement Learning via Slow Reinforcement learning) and [42]. For clarity of presentation, we will use [3] as a reference. Their objective is to learn a prior model across different, but similar tasks, i.e. Markov Decision Processes (MDP), by learning a Reinforcement Learning (RL) algorithm with another RL algorithm (thus the name $RL^2$). The inner RL algorithm is modeled via the weights of a Recurrent Neural Network (RNN) that are learned through an outer, general purpose Reinforcement Learning algorithm. As the target is to learn across different tasks, they formulate the learning process in the following way. *Trials* are a specific MDP selected from a set with several different MDP's in it. The number of times the agent is then allowed to sample from a specific MDP is called the number of *episodes*. So one trial has several episodes in it. The goal of the algorithm is to maximize the expected total discounted reward during a whole trial and not, as one usually does, only of one episode. It is thus important for the agent to know in which MDP it currently is in, to be able to act accordingly. To this end, the agent will have to integrate all past information such as rewards

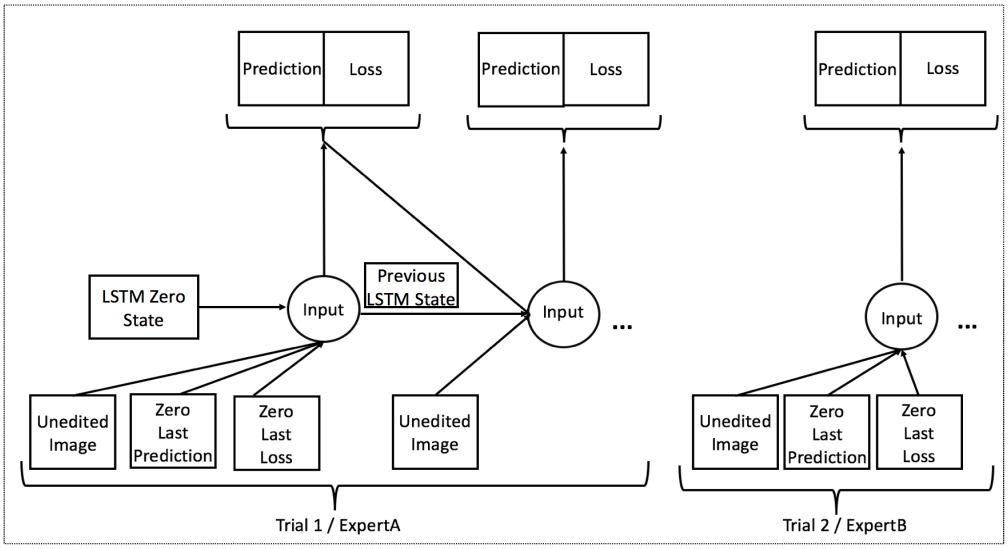



Figure 3.3: Our input always consists of the current image we have to predict the editing style for, the previous LSTM state and the previous prediction and loss.

and actions.

We adapt the $RL^2$ algorithm to the supervised learning domain as a way to do Meta-Learning and for further reference call it $SL^2$. In our work, the trials are the different experts that edit images in a specific way and an episode is just one image. That means one task with $n$ images of an expert in it, is one trial with $n$ episodes. The goal is that our model is able to recognize what expert it is currently training on. If it sees images form a new expert, it should also be able to adapt quickly to it. That's why we extend our architecture with a RNN. Specifically when our features have size $8x8x128$ (in Figure 3.1 just before the red, global feature block), we replace the fully-connected layer followed by a SELU activation and another fully-connected layer, with a fully-connected layer followed by a SELU activation and an LSTM [43]. The reason why we place the RNN at this exact spot is because there we calculate the global features. As we want our model to be able to differentiate in which task it is currently in, it makes sense to place the RNN where it might get the most general information about an image editing style and thus also knowledge about which expert it belongs to. Note that we train our model consecutively on each expert for a certain task size (similarly to MAML). Thus we teach the network to be able to adapt to new tasks quickly. Naturally just feeding the network with unedited images, will not let it learn what trial, i.e. expert it is currently learning on. This is why we also use similar inputs as in [3]. Namely the input to the network is the current, unedited image, the previous image prediction and the previous image loss (see Figure 3.3). Because we are using an LSTM, naturally we always have to feed

the previous state as well. Note that whenever we train on a new expert, we pass the zero state and the previous prediction and loss are also set to zero.

Chapter 4

# Experiments

## 4.1 Dataset

To evaluate different models we use the Adobe 5K dataset from Bychkovsky et al. [6] which has original images that are edited by 5 professional photographers (experts). Our training set for the DPE model (no Meta-Learning) consists of 2250 images and our test set consists of 498 images from Expert C (both are equivalent to the dataset used in Chen et al. [1]). When we do Meta-Learning, we train our meta-learner on all experts, except Expert C. Before evaluating on the test-set, we pre-train on K images of Expert C (K-Shot learning). From all expert-datasets, we remove the test-images of Expert C. This is to make sure, that the model has never seen the test-images before, not even if they were edited by another expert. Furthermore, we also remove 150 images which we use to pre-train on Expert C. That means for example when we pre-train on K = 30 and repeat the experiment for 5 times, we make sure that the model has never seen these images before. Note that for larger pre-training data sizes K, we do not make the guarantee that the model has never seen the image before, even if it is only from another expert. This then leaves us with a training set of about 4470 (depending on batch size) images per expert. Finally Chen et al. [1] use a data augmentation that multiplies the effective data size by 8, by using rotations and left-right shifts. Thus we also use data augmentation for our Meta-Learning algorithms. That means we train on 4 experts, with each one having a dataset of $8 \cdot 4470$, which leaves us with a final dataset of about 143'000 images. So even though we apply Meta-Learning, we initially still need a lot of data. But the important difference is, that firstly this data is already available, i.e. one does not need more data if one wants to Meta-Learn a new image editing style. Secondly, this Meta-Learning has to be done once, afterward one should be able to learn any image editing style with very few images.

## 4.2 Evaluation Metric

We use three different metrics to evaluate the performance of our models.

1. **PSNR** (Peak Signal to Noise Ratio):

$$20 \cdot log_{10}(\text{Max Pixel Value}) - 10 \cdot log_{10}(MSE(X, Y))$$

   In our case Max Pixel Value is 255, as we have encoded the images as 8-bit sRGB.

2. **SSIM** (Structural Similarity) [44]

3. **CIE-LAB Error**: This metric is calculated in the CIE-LAB color space in which each pixel is defined by a triplet $(L, a, b)$. The error is calculated as:

$$MSE(X_L, Y_L)$$

   i.e. the mean squared error of the lightness values of two images (the first element L of the triplet). We need this metric to be able to compare our results with Bychkovsky et al. [6]. As mentioned before, they are able to achieve good results after training only on few images.

## 4.3 Training

We trained all models for 50 Epochs. For the Deep Photo Enhancer Model, we use a learning rate of $10^{-5}$ from which we subtract $\frac{10^{-5}}{25}$ after every epoch, starting after the 25th epoch. We use the Adam Optimizer [4] and a batch size of 3 (equivalent to Chen et al. [1]). For our FOMAML implementation, we use an initial learning rate of $10^{-4}$ and no learning rate decay. For the inner optimization, we used Stochastic Gradient Descent and only for the outer optimization do we use Adam. Our task size was 30 images per expert. For our $SL^2$ implementation we use an initial learning rate of $10^{-3}$ which we change to $10^{-4}$ at epoch 17. We trained it with SGD and a batch size of 5 and equivalently a trial, i.e. task size of 5. As explained previously, the $SL^2$ algorithm always resets the state and the previous prediction and loss to zero for each new task that we train on. That means for the results we present, we actually don't make use of the idea of these special inputs. The reason is that our batch and task size are equivalent. So we always feed zero values for these inputs and additionally also feed zero values for the RNN state. We experimented with other settings but ran out of time to make a full analysis with different task sizes.

Chapter 5

# Results

## 5.1 Evaluation of Architecture

In this sub-section we review and extend the results of Chen et al. [1] and make a few pointedly observations which help us understand the need for Meta-Learning algorithms. We provide our own results for the Deep Photo Enhancer architecture. Furthermore, we also provide the initial loss of the original images and the edited images before training (see Table 5.1).

First, we have to note that Bychkovsky et al. [6] use a test set consisting of 2500 images, thus our test-datasets are not directly comparable[1]. Furthermore, we have to mention the fact that it might be possible that they decoded their images to another format than 8-bit sRGB, which might also be a factor in the comparison (although this seems unlikely given the results).

We also provide a comparison of the CIE-LAB Error of the Deep Photo Enhancer architecture with the Results of the GPR in Bychkovsky et al. [6]. They report that the average CIE-LAB Error in the test-set with Expert C (without training) is 16.3. This is higher than in our test-set where we report an average error of 11.94. Nevertheless, they show that after training their GPR, they are able to achieve an error of 4.7 on their test-set (after training on 2500 images). Compared to this the Deep-Photo Enhancer model performs quite well with an error of 5.42, even though it does not need any hand-crafted features. The dif-

[1] It was not possible for us to find out what images they used for their test-dataset.

| | PSNR | SSIM | CIE-LAB Error |
|---|---|---|---|
| **Deep-Photo Enhancer** | **23.87** | **0.8988** | **5.42** |
| Originals vs. Edited Images (no training) | 17.8 | 0.782 | 11.94 |

Table 5.1: Results of different models on the test set with 498 images of Expert C. For PSNR and SSIM, higher is better and for CIE-LAB Error lower is better.

| | PSNR | SSIM | CIE-LAB Error |
|---|---|---|---|
| Mean of Experts A,B,D,E vs Expert C | 21.35 | 0.8708 | 7.49 |
| Average Image B,C,D,E vs Expert A | 21.48 | 0.8689 | 8.59 |
| Average Image A,C,D,E vs Expert B | 24.97 | 0.9335 | 4.81 |
| **Average Image A,B,D,E vs. Expert C** | **23.74** | **0.9000** | **5.28** |
| Average Image A,B,C,E vs. Expert D | 23.21 | 0.9153 | 6.54 |
| Average Image A,B,C,D vs. Expert E | 22.20 | 0.9148 | 7.56 |
| **Mean of Average Images** | **23.12** | **0.9065** | **6.56** |

Table 5.2: Comparing test-datasets of different experts.

ference of 0.72 is much lower than the Just-noticeable-difference threshold of 2.3 [45] that one could perceive. Again we want to be precautious, we did not directly test our model on their test-dataset. Nevertheless, we do want to make the point that it seems like the model-free approach of Chen et al. [1], without using any features, yields competitive results to Bychkovsky et al.[6]. This is also the reason why we chose this architecture.
We present some further results to motivate the need and usefulness of Meta-Learning. As we will be training on all experts except Expert C (we will only pre-train on K data samples of Expert C in the end), we have to know how different the other experts are to Expert C. That's why we compute the average of evaluation metrics between the test-images of Expert C and all other Experts, i.e. we calculate the value of an evaluation metric of Expert C and A, Expert C and B etc. and take the average of that. We also compare the average test-images of a set of certain experts with the test-images of the remaining expert (see Table 5.2). Note that Bychkovsky et al. [6] reports that across the whole dataset of 5000 images, the average CIE-LAB Error between the edited images is 3.3 and the maximum error is 23.5.

Two things become apparent when looking at the results. Firstly, the *Mean of Average Images* shows a very high PSNR and SSIM (respectively a low CIE-LAB error). That means that generally speaking, taking the average of 4 Experts and evaluating on the remaining expert will yield good results. This implies that Meta-Learning an image editing style should be very useful. Secondly, we can see that taking the average of all experts, except Expert C, and then testing

| | K | Epochs | PSNR | SSIM | CIE-LAB Error |
|---|---|---|---|---|---|
| *BASELINE 1* | 5 | 1 | $17.80 \pm 0$ | $0.7820 \pm 0$ | $11.94 \pm 0$ |
| | 5 | 50 | $17.80 \pm 0$ | $0.7820 \pm 0$ | $11.91 \pm 0.02$ |
| | 30 | 1 | $17.80 \pm 0$ | $0.7820 \pm 0$ | $11.94 \pm 0$ |
| | 30 | 50 | $19.59 \pm 0.50$ | $0.8158 \pm 0.01$ | $9.81 \pm 0.73$ |
| *BASELINE 2* | 0 | 50 | 23.34 | 0.8848 | 5.56 |
| *BASELINE 3* | 5 | 1 | $23.37 \pm 0.04$ | $0.8853 \pm 0.0$ | $5.53 \pm 0.04$ |
| | 5 | 50 | $23.39 \pm 0.89$ | $0.8864 \pm 0.0$ | $5.50 \pm 0.05$ |
| | 30 | 1 | $23.37 \pm 0.04$ | $0.8864 \pm 0.0$ | $5.52 \pm 0.07$ |
| | 30 | 50 | $23.62 \pm 0.15$ | $0.8901 \pm 0.0$ | $5.43 \pm 0.03$ |
| *Bychkovsky et al. [6]* | 5 | - | - | - | $\sim 7.5$[2] |
| | 30 | - | - | - | $\sim 7.5$ |
| *FOMAML* | 0 | 50 | 22.71 | 0.8689 | 5.77 |
| | 5 | 1 | $22.86 \pm 0.16$ | $0.8712 \pm 0.0$ | $5.70 \pm 0.07$ |
| | 5 | 50 | $22.90 \pm 0.21$ | $0.8768 \pm 0.01$ | $5.69 \pm 0.08$ |
| | 30 | 1 | $22.88 \pm 0.26$ | $0.8752 \pm 0.0$ | $5.73 \pm 0.07$ |
| | 30 | 50 | $23.02 \pm 0.36$ | $0.8807 \pm 0.0$ | $5.69 \pm 0.11$ |
| $SL^2$ | 0 | 50 | 23.31 | 0.8800 | 5.54 |
| | 5 | 1 | $23.32 \pm 0.01$ | $0.8806 \pm 0.0$ | $5.54 \pm 0.0$ |
| | 5 | 50 | $23.32 \pm 0.01$ | $0.08817 \pm 0.0$ | $5.54 \pm 0.01$ |
| | 30 | 1 | $23.32 \pm 0.02$ | $0.8813 \pm 0.0$ | $5.53 \pm 0.01$ |
| | 5 | 50 | $23.36 \pm 0.12$ | $0.8904 \pm 0.0$ | $5.48 \pm 0.06$ |

Table 5.3: Comparing our Meta-Learning Algorithms with 3 *BASELINES*. When K=0 for the Meta-Learning algorithms, we did not pre-train on any data of Expert C. We provide 99% confidence intervals for pre-training on K=5 and K=30 for 5 experiments. Note that for *Baseline 2* we report the SSIM and CIE-LAB error that corresponded to the best PSNR value and not necessarily the best overall values. This is to make the *BASELINE* 2 and 3 comparable.

on Expert C yields a PSNR of 23.74 which is already competitive with the results of the Deep-Photo Enhancer architecture of 23.87. As our Meta-Learning algorithms will only train on images that are not edited by Expert C (naturally except for the pre-training process on Expert C), we will also learn some kind of "average", prior image editing style. Thus this result of 23.74 gives us a hint of our model capacity.

## 5.2 Evaluation of Meta-Learning Algorithms

In this section we will explore how our algorithms perform for different sizes K of the pre-training dataset of Expert C. To evaluate the effectiveness of our Meta-Learning algorithms, we propose 3 different baselines. Note that the evaluation process always happens on the test-dataset with 498 images.

1. *BASELINE 1*: Train the Deep Photo Enhancer model on K=30 and K=5 images of Expert C.
2. *BASELINE 2*: Train the Deep Photo Enhancer model on all data of all 4 Experts (except Expert C).
3. *BASELINE 3*: Same as *BASELINE 2* but also pre-train the model on K=30 and K=5 images of Expert C.

Note that when we pre-train on few images such as K=5 or K=30, the data that we randomly selected can have a high influence on the results. Thus whenever we pre-train on K=5 or K=30 images, we select 5 sets of K datapoints and report the average performance for all 5 pre-trained models using 99% confidence intervals. For K that are larger than 30 we only performed the experiment once. We did not use any data augmentation, so K is the actual number of images that we trained on. But we do want to mention that data augmentation experimentally improved the results and thus would be useful for a real application. We also need to mention again that when we pre-train our $SL^2$ algorithm on Expert C, we always set all inputs except the current image, i.e. previous prediction and previous loss to zero for each batch. That means that we also set the state for the RNN to zero. The reason is that we want the pre-train process to be equivalent to the training process. Lastly one is usually interested in how quickly a model trained with a Meta-Learning algorithm is able to adapt to a new task. Thus we always report the evaluation metrics after 1 and after 50 Epochs.

First of all, we can see that the Deep Photo Enhancer, trained on K images of Expert C performs very poorly (*BASELINE 1* in Table 5.3) and is beaten by every other experiment. But training the Deep Photo Enhancer model (without Meta-Learning) on all experts except Expert C (*BASELINE 2* in Table 5.3) already yields very good results. One can notice that it is still worse than only training on 2250 images of Expert C (Table 5.1) and it also does not reach the full model capacity that has been indicated by the average image of these other experts (Table 5.2). Nevertheless, such a good performance was expected after having seen how well an average image already performs. Taking the model of *BASELINE 2* and pre-training it on Expert C yields the results of *BASELINE 3* in Table 5.3 which are very competitive with the original model that was trained on 2250 images of Expert C. This is very impressive considering the fact that we

[2]We had to visually interpret these results from a graphic of their paper.

use only a fraction of the data of Expert C. We can further see that increasing the pre-train data size yields results that significantly outperform Chen et al. [1] (see Table 5.4).

We would again like to make a comparison to Bychkovsky et al. [6] which yielded very good results when training on few images (see Table 5.3). We can see that all our methods are competitive with their results. We mention again, that they use a different test and train set. The point we would like to make here is that Deep Learning methods can achieve competitive results with feature-based methods when training on only few data points.

The results further show, that our *FOMAML* algorithm performs much worse than the other methods (see Table 5.3 and 5.4). We believe that this is due to the fact that the model receives only sparse gradient updates. The reason is that for each expert we need 30 images to optimize and then another 30 to evaluate the loss function to be able to calculate the Meta gradient. We do this process for 4 experts to finally get one final gradient update. We tried to train the model for longer than 50 epochs, but it did not improve the results. Another

| | K | Epochs | PSNR | SSIM | CIE-LAB Error |
|---|---|---|---|---|---|
| *BASELINE 3* | 100 | 1 | 23.34 | 0.8838 | 5.49 |
| | 100 | 50 | 23.74 | 0.8925 | 5.30 |
| | 1000 | 1 | 23.91 | 0.8910 | 5.31 |
| | 1000 | 50 | 24.30 | 0.9010 | 5.1477 |
| | 2250 | 1 | 23.77 | 0.8911 | 5.5067 |
| | 2250 | 50 | 24.19 | 0.9017 | 5.1878 |
| *FOMAML* | 100 | 1 | 23.32 | 0.8842 | 5.53 |
| | 100 | 50 | 23.39 | 0.8858 | 5.50 |
| | 1000 | 1 | 23.11 | 0.8825 | 5.77 |
| | 1000 | 50 | 23.11 | 0.8834 | 5.77 |
| | 2250 | 1 | 23.21 | 0.8842 | 5.67 |
| | 2250 | 50 | 23.44 | 0.8882 | 5.55 |
| $SL^2$ | 100 | 1 | 23.40 | 0.8842 | 5.46 |
| | 100 | 50 | 23.72 | 0.8948 | 5.32 |
| | 1000 | 1 | 23.79 | 0.8953 | 5.34 |
| | 1000 | 50 | 24.01 | 0.8981 | 5.29 |
| | 2250 | 1 | 23.74 | 0.8923 | 5.45 |
| | 2250 | 50 | 23.81 | 0.8926 | 5.43 |

Table 5.4: All models have been trained with 50 Epochs on K images of Expert C. Note that for K = 2250 the dataset is identical to the training dataset of the Deep Photo Enhancer Model in 5.1.

reason why this algorithm might not work, is that it has been shown to work on image classification tasks. In such a scenario, images between classes might have much in common, but a classification task still has to differentiate between clearly distinctive images. The task of predicting an image editing style, where the changes to an image are only subtle and the content of the image stays the same, has only small changes between different styles. Thus it might be, that this algorithm is not optimally suited for such a task.

This is the reason why we turned to the $SL^2$ algorithm which should be more suited for similar tasks. As one can see, the results (see Table 5.3) are much better than for *FOMAML*. Unfortunately $SL^2$ is not able to outperform a normal optimization with Adam (*BASELINE 2*), with or without pre-training on images of Expert C. But although it is not able to outperform these baselines, it is still able to improve on Chen et al. [1] when pre-training on larger data sizes of Expert C (see Table 5.4). It might be possible that using larger Meta task sizes, i.e. actually using the previous loss and prediction as an input, could improve the performance. We will leave this up to future work. Lastly looking at Figure 5.1 we can see that *BASELINE 3*, *FOMAML* and $SL^2$ visually yield quite similar results to DPE trained on 2250 images.

(a) Original Image

(b) Label Edited by Expert C

(c) DPE trained on 2250 images of Expert C

(d) Average Image of Experts A,B,D,E

(e) DPE trained on all experts A,B,D,E + pretrained on 30 images of Expert C (no Meta-Learning)

(f) Ours: FOMAML trained on all experts A,B,D,E + pretrained on 30 images of Expert C

(g) Ours: $SL^2$ trained on all experts A,B,D,E + pretrained on 30 images of Expert C

Figure 5.1: Visual comparison of different algorithms

Chapter 6

# Conclusion

In our work we have evaluated different Meta-Learning algorithms such as *FO-MAML* and $SL^2$. Both approaches have not been able to outperform an optimization with a standard Adam optimizer. Nevertheless, we were able to significantly improve on the results of Chen et al. [1], even when using notably less data. This was achieved by learning a prior image editing style from other experts, before training on the expert of interest. Bychkovsky et al. [6] showed that learning a Covariance Function on Expert E (which is supposedly similar to expert C) improves the results of training on few data of Expert C. Akin to them, we show that learning a prior editing style on all experts also improves the results in the domain of Deep Learning.

Chapter 7

# Future Work

In future work in this area, we would first like to do a further analysis of the $SL^2$ algorithm, especially to find out if other parameter settings such as the learning rate, the optimizer and the task size have a positive influence on the results. We could also try out other Meta-Learning algorithms on the problem of learning an image editing style. For example, one could directly learn the inner learning rate through meta-gradient steps. Another line of thought would be to combine our work with [10], which would allow us to make a selection of few images with maximal information about the editing style. One could also try a totally different approach to learn an image-editing style, by creating a new dataset based on the dataset of Bychkovsky et al.[6] which contains the individual editing steps that were made for a specific image. This would then lead to a feature based supervised learning problem where one would try to predict the editing-actions given an input image, instead of directly predicting an image. Even though it would be feature based, the dataset would contain all possible editing steps that are possible and thus one might achieve better results without having to sacrifice the model capacity. We also recognize the usefulness of unpaired approaches, thus it would be interesting to see if one could combine Meta-Learning with one of these approaches.
Generally speaking, we would be interested in evaluating our and others work with more loss-functions. Specifically, we would like to understand how image editing affects a loss function and how well different loss functions capture the representation of a style. This has been a long-standing problem in the Computer Vision community and is especially important when trying to evaluate the prediction of an image editing style.

# Bibliography


[1] Y.-S. Chen, Y.-C. Wang, M.-H. Kao, and Y.-Y. Chuang, "Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 6306–6314.

[2] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[3] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl$^2$: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[5] T. Flanders, "Speakers give sound advice," *Syracuse Post Standard*, p. 18, March 1911.

[6] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, "Learning photographic global tonal adjustment with a database of input/output image pairs," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 97–104.

[7] S. B. Kang, A. Kapoor, and D. Lischinski, "Personalization of image enhancement," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1799–1806.

[8] J. C. Caicedo, A. Kapoor, and S. B. Kang, "Collaborative personalization of image enhancement," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 249–256.

[9] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu, "Automatic photo adjustment using deep neural networks," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 2, p. 11, 2016.

[10] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

[11] F. Berthouzoz, W. Li, M. Dontcheva, and M. Agrawala, "A framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations." *ACM Trans. Graph.*, vol. 30, no. 5, pp. 120–1, 2011.

[12] C. E. Rasmussen and C. K. Williams, "Gaussian process for machine learning," p. 100.

[13] L. Kaufman, D. Lischinski, and M. Werman, "Content-aware automatic photo enhancement," in *Computer Graphics Forum*, vol. 31, no. 8. Wiley Online Library, 2012, pp. 2528–2540.

[14] L. Yuan and J. Sun, "Automatic exposure correction of consumer photographs," in *European Conference on Computer Vision*. Springer, 2012, pp. 771–785.

[15] J. Yan, S. Lin, S. Bing Kang, and X. Tang, "A learning-to-rank approach for image color enhancement," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2987–2994.

[16] P. S. Chandakkar and B. Li, "Joint regression and ranking for image enhancement," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 235–243.

[17] R. Yu, W. Liu, Y. Zhang, Z. Qu, D. Zhao, and B. Zhang, "Deepexposure: Learning to expose photos with asynchronously reinforced adversarial learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 2153–2163.

[18] H. Yang, B. Wang, N. Vesdapunt, M. Guo, and S. B. Kang, "Personalized exposure control using adaptive metering and reinforcement learning," *IEEE transactions on visualization and computer graphics*, 2018.

[19] H. Fang and M. Zhang, "Creatism: A deep-learning photographer capable of creating professional work," *arXiv preprint arXiv:1707.03491*, 2017.

[20] Y. Hu, H. He, C. Xu, B. Wang, and S. Lin, "Exposure: A white-box photo post-processing framework," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 2, p. 26, 2018.

[21] A. Nazemi, S. Kamyab, Z. Azimifar, and P. Fieguth, "Human perception-based image enhancement using a deep generative model," *Journal of Computational Vision and Imaging Systems*, vol. 4, no. 1, pp. 3–3, 2018.

[22] J. Park, J.-Y. Lee, D. Yoo, and I. So Kweon, "Distort-and-recover: Color enhancement using deep reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5928–5936.

[23] M. Omiya, E. Simo-Serra, S. Iizuka, and H. Ishikawa, "Learning photo enhancement by black-box model optimization data generation," in *SIGGRAPH Asia 2018 Technical Briefs*. ACM, 2018, p. 7.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[25] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint*, 2017.

[26] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," *arXiv preprint arXiv:1703.05192*, 2017.

[27] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 700–708.

[28] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," *CoRR, abs/1703.07511*, vol. 2, 2017.

[29] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, "A closed-form solution to photorealistic image stylization," *arXiv preprint arXiv:1802.06474*, 2018.

[30] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 110, 2016.

[31] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1511–1520.

[32] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[33] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[34] H. Talebi and P. Milanfar, "Learned perceptual image enhancement," in *Computational Photography (ICCP), 2018 IEEE International Conference on*. IEEE, 2018, pp. 1–13.

[35] Y. Deng, C. C. Loy, and X. Tang, "Aesthetic-driven image enhancement by adversarial learning," in *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 2018, pp. 870–878.

[36] A. Ignatov, N. Kobyshev, R. Timofte, K. Vanhoey, and L. Van Gool, "Dslr-quality photos on mobile devices with deep convolutional networks," in *the IEEE Int. Conf. on Computer Vision (ICCV)*, 2017.

[37] A. Ignatov, N. Kobyshev, R. Timofte, K. Vanhoey, and L. Van Gool, "Wespe: weakly supervised photo enhancer for digital cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 691–700.

[38] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 971–980.

[39] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[40] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *CoRR, abs/1803.02999*, vol. 2, 2018.

[41] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent," in *International Conference on Artificial Neural Networks*. Springer, 2001, pp. 87–94.

[42] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763*, 2016.

[43] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[44] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[45] G. Sharma and R. Bala, *Digital color imaging handbook*. CRC press, 2002.