Solution of 1. Assignment

Deadline: 12.11.17

Points: (maximum 80 Points)

Exercise 1	Exercise 2	Exercise 3	Exercise 4	Exercise 5	Exercise 6	Exercise 7
20	10	10	10	10	10	10
						Total
						80

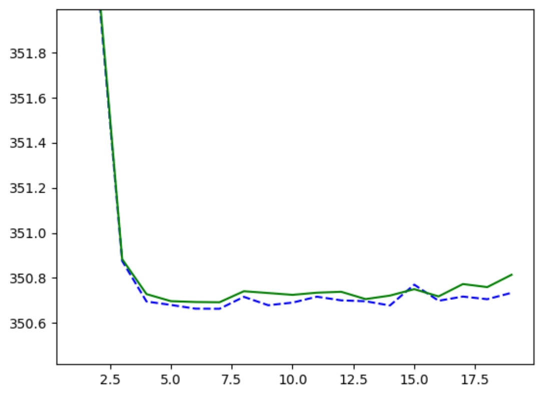
Exercise 1:

We decided to combine all training images in one design matrix of shape (number of pixels, 3) (Here 'number of pixels' includes all pixels of all 3 images). Accompanying the design matrix we also pass a tensor 'r' of shape (number of pixels), which holds the distances for every pixel. From 'r' we compute a tensor of shape(n+1, number of pixels), where n is the degree of the polynomial. From this and the parameters 'W' we compute the polynomial itself, which is then multiplied with the original pixels to construct the predicted pixels.
We compute the MSE as loss . The MSE, as shown in the lecture, also minimizes the ML-Estimator and is therefore a proper choice.The loss is optimzed with the AdamOptimizer(which performed much better than the GradientDescentOptimizer).

[ ] **of 20 Points**

Exercise 2:

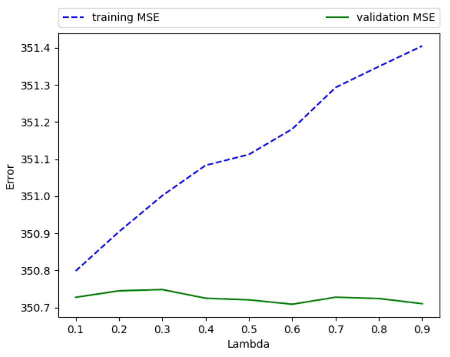
Results only got significantly better up to $n = 4$, after that the training error stayed more or less the same with the validation error increasing slightly so overfitting isn't a big problem.
After occam's razor we decided to view $n = 4$ as the optimal hyperparameter.

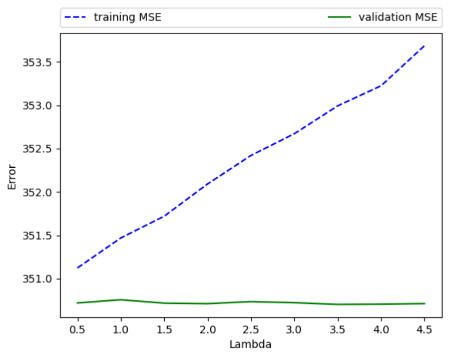


[ ] **of 10 Points**

Exercise 3:

Regularization is a method to deal with overfitting, which we don't really have in this case. We did not manage to obtain a better result than for $n = 4$, however were able to get a result as good as that with a significantly higher polynomial degrees. In our test changing λ didn't make too much of a difference on validation error. Using a polynomial of degree 10 we got best results with $\lambda \approx 0.6$.

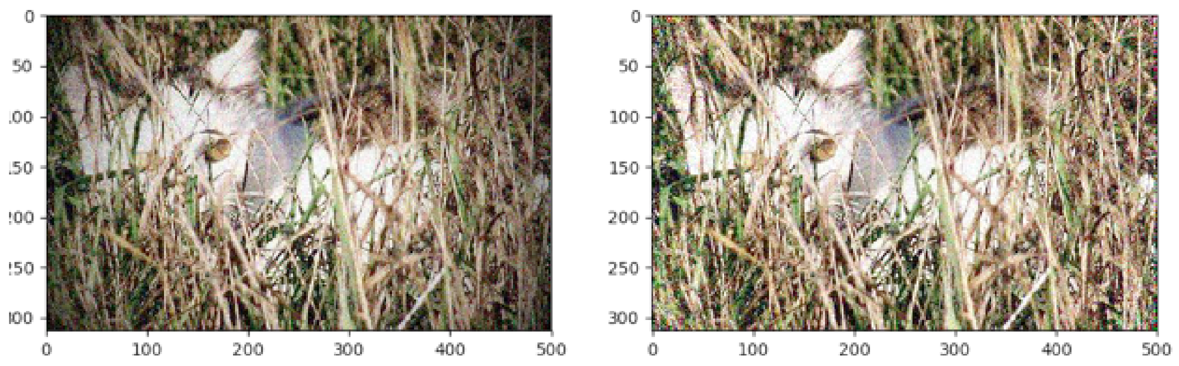


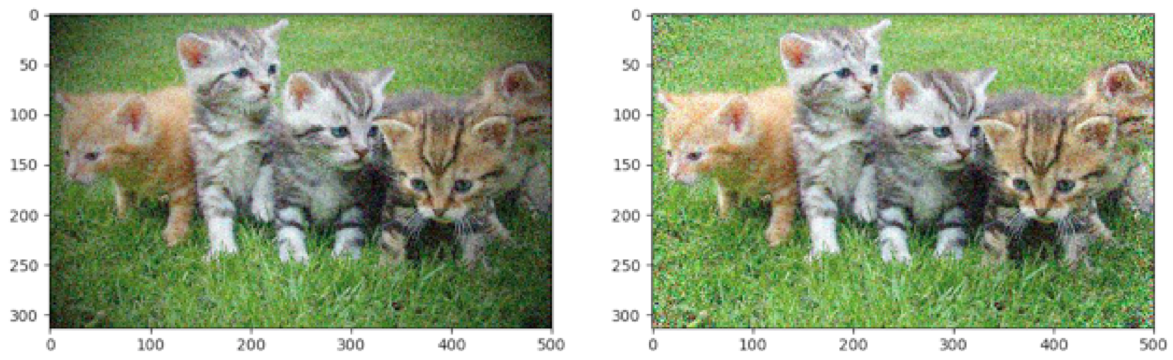


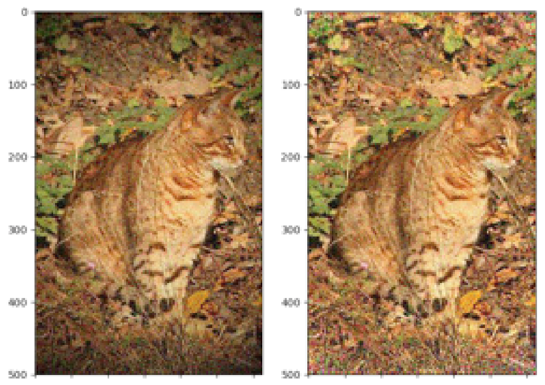
Our results were similar with a polynomial degree of 20 and lambdas ranging from 0.5 to 50 (the curve had a small incline, thus smaller lambdas achieved better results.)

[ ] **of 10 Points**

Exercise 4:







These images were obtained by our devignetting method provided with a fully trained polynomial of degree 4. The results are nice, but the noise in the corners seems more noticeable. We explain this by the following:

Let $p_n(r, \Theta) := a_0 \cdot r^0 + \cdots + a_n \cdot r^n$. Then $I'(x, y) = I(x, y) \cdot p_n(r, \Theta) + \text{noise}$. We obtained the devignetted pixels by the following equation:

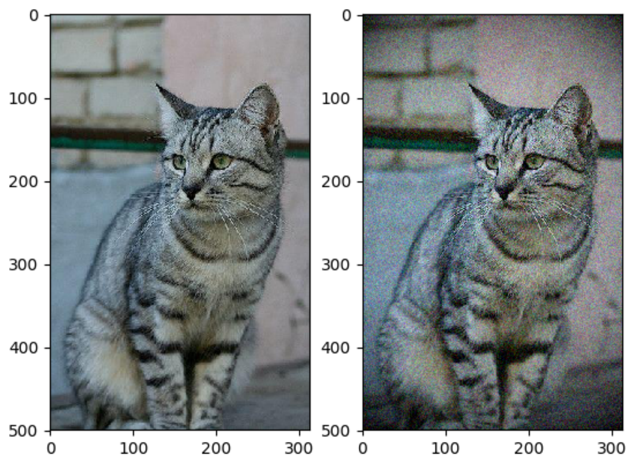
$I''(x, y) = \frac{I'(x,y)}{p_n(r,\Theta)} = \frac{I(x,y) \cdot p_n(r,\Theta) + \text{noise}}{p_n(r,\Theta)} = I(x, y) + \frac{\text{noise}}{p_n(r,\Theta)}$. And since $p_n(r, \Theta)$ tends to be smaller than 1 in the corners, the noise gets brighter and thereby more noticeable.

[ ] **of 10 Points**

Exercise 5:

The model does not help with removing the noise, since the noise is randomly distributed. Therefore it is impossible to learn something about the noise. Thus we cannot remove the noise with this model. (I do not really know what is meant by 'and look at what you observe in practise'. But the devignetting we applied did not affect the noise in the middle at all.)
If we assume, that our prediction model $\hat{f}$ approximates f such that there is only a small difference left, the following deduces a formula for σ:
$var(y - \hat{f}(x)) = E[(y - \hat{f}(x))^2] - E[y - \hat{f}(x)] = E[(y - \hat{f}(x))^2]$. The last equality holds, because the noise has mean $f(x)$ and we assume $\hat{f}(x)$ approximates $f(x)$ good enough. Therfore $\sigma^2 = var(y - \hat{f}(x)) = E[(y - \hat{f}(x))^2] = \text{MSE}(y - \hat{f}(x)) \approx 350$. Therefore $\sigma \approx 18.7$

This value seems plausible since, when adding this gaussian noise with $\sigma = 18.7$ to the vignetting_example.py-script we obtained the following image. The amount of noise seems very similar.



[ ] **of 10 Points**

Exercise 6:

Let $E : \mathbb{R}^3 \to \mathbb{R}, E(\Theta) = 2\Theta_1^2 + 4\Theta_2 + \max(0, \Theta_2 + \Theta_3)$ be a loss function. In order to perfrom the gradient descent, we first give the gradient of E (which only makes sense if $\Theta_2 + \Theta_3 \neq 0$):

$$\nabla E(\Theta) = \begin{cases} \begin{pmatrix} 4\Theta_1 \\ 5 \\ 1 \end{pmatrix} & \text{für } \Theta_2 + \Theta_3 > 0 \\ \begin{pmatrix} 4\Theta_1 \\ 1 \\ 0 \end{pmatrix} & \text{für } \Theta_2 + \Theta_3 < 0 \end{cases} \tag{1}$$

With this we compute the following:

$$\nabla E(\Theta^{[0]}) = \begin{pmatrix} 8 \\ 5 \\ 1 \end{pmatrix} \Rightarrow \Theta^{[1]} = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} - \frac{1}{2} \begin{pmatrix} 8 \\ 5 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ -1.5 \\ -0.5 \end{pmatrix} \text{ and}$$

$$\nabla E(\Theta^{[1]}) = \begin{pmatrix} -8 \\ 4 \\ 0 \end{pmatrix} \Rightarrow \Theta^{[2]} = \begin{pmatrix} -2 \\ -1.5 \\ -0.5 \end{pmatrix} - \frac{1}{2} \begin{pmatrix} -8 \\ 4 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ -3.5 \\ -0.5 \end{pmatrix}$$

[ ] **of 10 Points**

Exercise 7:

Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ be two differentiable functions. Consider the concatenation $h = g \circ f$. Let $p \in \mathbb{R}^n$ and let r_f, r_g denote the remainders of the linear approximation of f and g. Since both f and g are differentiable it holds, that $\lim_{\tau \to 0} \frac{r_f(\tau)}{\tau} = 0$ and $\lim_{\tau \to 0} \frac{r_g(\tau)}{\tau} = 0$

$$\begin{aligned} g(f(p+\tau h)) - g(f(p)) &= g'(f(p))(f(p+\tau h) - f(p)) + r_g(f(p+\tau h) - f(p)) && (2)\\ &= g'(f(p))(\tau \nabla f(p)^T h + r_f(\tau)) + r_g(f(p+\tau h) - f(p)) && (3)\\ &= g'(f(p))\tau \nabla f(p)^T h + g'(f(p)) r_f(\tau)) + r_g(f(p+\tau h) - f(p)) && (4) \end{aligned}$$

Now we have still have to show that $lim_{\tau \to 0} g'(f(p)) r_f(\tau)) + r_g(f(p+\tau h) - f(p)) = 0$, this holds because:

$$\lim_{\tau \to 0} \frac{r_f(\tau)}{\tau} = 0$$

and

$$\lim_{f(p+\tau h)-f(p) \to 0} \frac{r_g(f(p+\tau h) - f(p))}{f(p+\tau h) - f(p)} = \lim_{\tau \to 0} \frac{r_g(\tau)}{\tau} = 0$$

With this we can rewrite (3) to:

$$h(p+\tau h) - h(p) = g(f(p+\tau h)) - g(f(p)) = g'(f(p))\tau \nabla f(p)^T h + r_h(\tau)$$

with $\lim_{\tau \to 0} \frac{r_h(\tau)}{\tau} = 0$.
This matches exactly to the definiton(as a linear approximation) of the gradient/derivative.

[ ] of 10 Points