ESTIMATE PHOTOGRAPHIC SETTINGS FROM IMAGES

REPORT FOR CS229 FINAL PROJECT

Jun Wang
Department of Applied Physics
junwang9@stanford.edu

Jenny Hu
Department of Applied Physics
jingjinh@stanford.edu

May 7, 2021

ABSTRACT

This work focuses on estimating photographic parameters, including F-number, focal length and ISO, based on the resulting photos. We first train a vanilla neural network for each of the three parameters as the baseline, then apply convolutionary layers and Fourier transformation to improve the model, obtaining a reasonable prediction on F-number and focal length.

1 Introduction

Photographic parameters, e.g. aperture, ISO and focal length, are determining factors for the style and quality of resulting photos. While expert photographers may have the intuition to appropriately pick a lens and set up the parameters, it would be great for general photographers to know the settings of existing photos they want to take reference from. Although these information are stored in Exchangeable Image File Format (EXIF) that are recorded by most of contemporary digital cameras, they are not assumed when people come across an old photo or a compressed duplicate on social media. Thus estimating photographic settings merely based on images can be a helpful tool.

While application of machine learning algorithms has made significant advances in object recognition, capturing underlying information about styles and forms from images is a rather underexplored field. Distinguishing the form of an image from its contents is a major question in this field. Since photographic settings cannot affect what objects appear in the photo, its relevant information is confined in the form. Investigation on estimating them may also help people understand the distinction between form and contents.

The input to our algorithms is a bunch of images with three labels: F-number, ISO and focal length, and the outputs are neural networks. Our trained neural network takes an image as input and outputs the estimated values of the labels.

2 Related Work

Previous works on image style transferring with convolutional neural networks (CNN) has achieved impressive results Gatys et al. [2016, 2015]. Photographic settings are pertinent to the form of images as well, yet its target is more quantitative than style transferring. Previous works on estimating camera settings Gupta and Kannan [2017], Laurence and Murphy [2018] are mostly doing classifications after discretization, with fair results. The authors of Gupta and Kannan [2017] have a clear illustration of the effects of variation in exposure time, aperture and ISO. Besides aperture and ISO, we focus on focal length of the lens because it has fundamental connection with aberrations. We ignore exposure time because the effect also strongly depends on the speed of objects.

Based on the physics of imaging and aberrations, we notice that the effects of F-number and focal length have a primary dependence on the distance to the center of field of view (FoV). Thus we have been paying attention to methods that can incorporate coordinate information into training. A CoordConv layer Liu et al. [2018] add channels containing the coordinates to a usual CNN layer, and it is shown, for some tasks, to help the model learn translation dependence to varying extent. We intend to try applying a CoordConv layer in our model.

3 Dataset and Features

Our main dataset is downloaded from PixelPeeper, which categorizes full-size photos posted on Flickr by cameras and lenses. We downloaded around ten thousands photos that are shot with the top-50 most popular lenses , coarsely maintaining a uniform distribution on different lenses. The downloaded images are in Flickr's medium size, and we discarded those too far from the standard shape (640×427) and cropped them into a uniform 640×426 shape. Then we filtered out images missing any of the information we are interested in (focal length, F-number, ISO), obtaining our primary dataset that contains more than 6000 images. We randomly chose a training set of 5000 images and a test set of 1000 images and adhere to this split across the experiments.

To facilitate preliminary experiments with different feature transformations, we took a series of photos on the same object using aperture ranging from 1.7 to 14 with fixed 30mm focal length. On this toy dataset, we tried resizing to 75×75, resizing to 400×600 and doing 2D Fourier transformation. Two examples in this toy dataset are shown in Fig. 1 . Our preliminary experiments implied that compressing to 75×75 deteriorates resolution too much to support the algorithm, so we decided to train on 640×426 images in our primary experiments.

We preprocessed the images from RGB to grey scale, since the camera settings we are interested in are more pertinent to the shapes than colors. We normalized the grey scale images before feeding into the algorithms as input features.

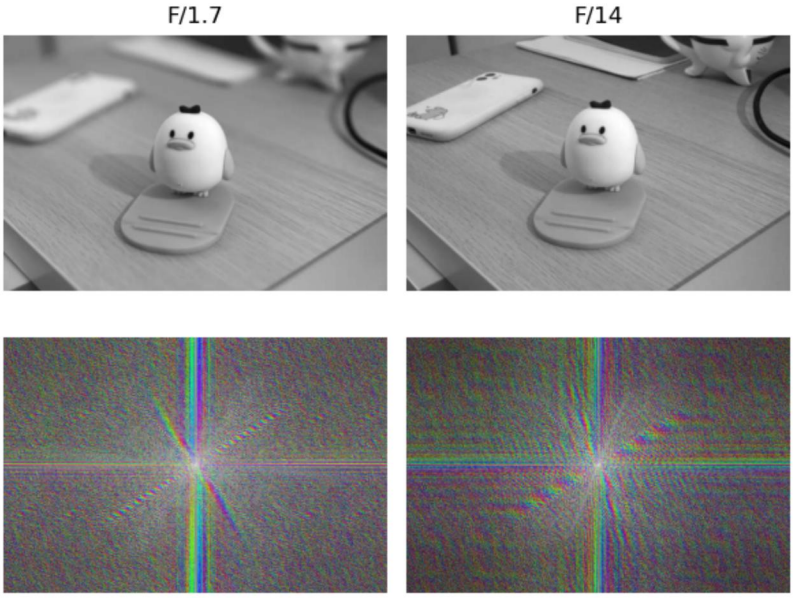

Figure 1: Visual illustration of variation in aperture, in both image and Fourier spectra. The Fourier spectra are visualized by mapping the amplitude to brightness and the argument of complex value to hue.

As for labels, we are particularly interested in the focal length, the aperture, and the ISO. We started with training three models independently for each of the parameter because intuitively these parameters are not correlated with each other. Later we also tried different combinations of the labels such as focal length + aperture. Labels are normalized when multiple labels are trained with one model.

4 Methods

4.1 Vanilla Neural Network

Considering to the high dimensionality of our training input and the continuity of our label/prediction values, we implement a neural network to do the regression. The idea of the neural network (NN) is that we first build a prediction model containing L layers of neurons:

In this model, σ is the activation function, common choices of which includes ReLU, Sigmoid and tanh. The nonlinearity of the activation function allows the NN to capture the behaviors of complicated data sets.

Algorithm 1 Neural Network

▷ X is the feature (d-dimensional), y is the label, W and b denote weight and bias of each layer.
Layer 1: Compute $a^{[1]} = \sigma(W^{[1]}X + b^{[1]})$
▷ $W^{[1]}$ has shape of $n_{h1} \times d$, where n_{h1} is the number of hidden neuron in layer 1.
Layer 2: Compute $a^{[2]} = \sigma(W^{[2]}a^{[1]} + b^{[2]})$
▷ $W^{[2]}$ has shape of $n_{h2} \times n_{h1}$, where n_{h2} is the number of hidden neuron in layer 2.
.....
Layer L: Compute $y = \sigma(W^{[d]}a^{[d]} + b^{[d]})$ ▷ $W^{[d]}$ has shape of $1 \times n_{h(L-1)}$.

$W^{[l]}$ and $b^{[l]}$ are the parameters we are trying to optimize. The most common optimizer used in a NN is minibatch stochastic gradient descent (SGD) and its variations. J is the task-specific loss function, and its gradients can be calculated with back propagation technique. In this project, we choose J as mean square loss (MSE) function because we are doing regression.

4.2 Convolutional Neural Network

A convolutional layer reduces the number of parameters by constraining the entries in weights $W^{[l]}$ and biases $b^{[l]}$ based on the spatial structure of the data. Specifically a convolution kernel for 2D data with c channels has c 2D filters. It takes an $H \times W \times c$ input and its 2D filters separately carries out convolutions on the c channels and then output a (biased) weighted sum of the c 2D convolutions. The output of these affine operations is then sent into an elementwise activation function for further forward propagation. Formally, the forward propagation of a 2D convolutional layer is $a^{[l+1]}_{k'} = \sigma(\sum_k F_{k'k} * a^{[l]}_k + b_{k'})$, where $g_1 * g_2$ denotes the 2D convolution between two 2D array g_1 and g_2, F, b are the weights and biases, σ the activation function, and $a^{[l]}, a^{[l+1]}$ the input and output of this layer respectively.

A CoordConv layer Liu et al. [2018] is a convolutional layer with additional channels containing the coordinates of each pixel. The depths of kernels will increase accordingly, while the number of output channels is not affected. In our case, we appended additional $\rho = \sqrt{(i - \frac{w}{2})^2 + (j - \frac{h}{2})^2}$ channels to some of the convolutional layers, where w, h are the layer-specific width and height, respectively.

5 Experiments

5.1 Vanilla Neural Network

We started with implementing an all-to-all connected NN with pytorch, and testing with a small set of sample (100 samples). We built a narrow and shallow NN with only 1 hidden layer/100 neurons, and experimented different optimizers (SGD, Adam), weight initialization methods (uniform, normal, Xavier, Kaiming), and activation functions (SELU, ReLU, Sigmoid, Tanh). The most stable combination that produce reasonable learning result is Adam optimizer + Xavier initialization + SELU, and we stick to them in the following exploration.

Next, we increased sample quantity, training with more neurons and layers as the model started to show underfit behavior. At the point that we can load our full training set (5000 samples), we further optimize the hyperparameters to get the baseline of this vanilla NN. During this process, we usually choose the largest learning rate that does not make the training loss oscillate drastically. We found out that the training loss is minimized when we used two hidden layers, with 1000 and 100 neurons respectively. We trained three NN models for focal length, aperture and ISO independently.

5.2 Convolutional Neural Network

We started with a plain, shallow CNN with small kernels and grey-scale images directly as input. We tuned the hyperparameters, including kernel size, strides, number of channels and number of layers to further improve our model in a trial and error manner, and we ended up using the architecture shown in Fig. 2 . We did not do cross validation considering our limited computing power and time.

During our experiments with the CNN, we found out that feature normalization and batch size are critical to successfully training a model. Without feature normalization, the model could hardly converge. With small batch size and large number of batches, the model tends to drastically fluctuate and converge very slowly. As a result, we chose the large batch size (1000) that is commensurable to the training set volume but does not raise memory error.

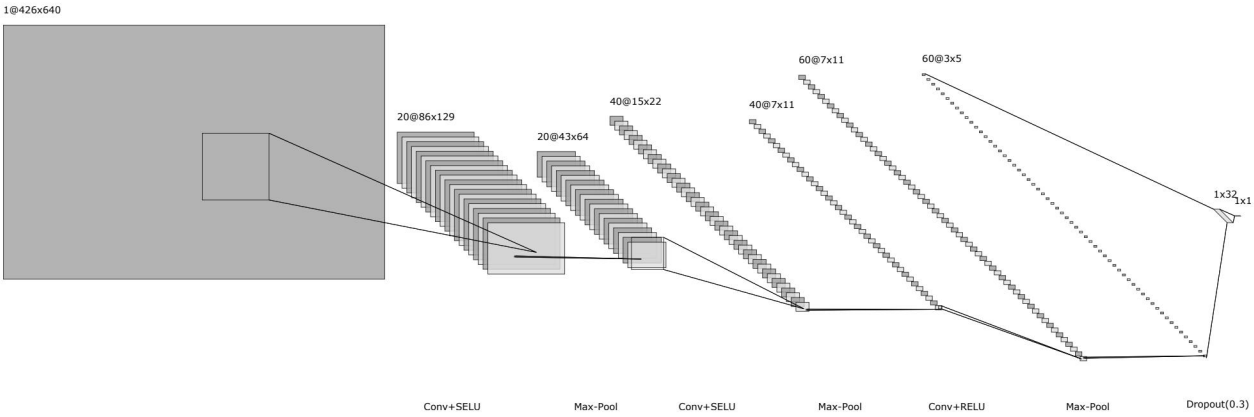

Figure 2: Architecture of the CNN for a single label.

For experiments with CoordConv, we used the same architecture but replaced the first and second convolution layers with CoordConv layers. For experiments with Fourier transformed features, we treated the real and imaginary parts of Fourier spectrum as two channels, rather than converting to a colored image as in Fig. 1 . We used the same hyperparameters as the plain CNN on the Fourier transformed features.

For each kind of CNN experiments: CNN, CNN with Fourier spectra as features (FT-CNN), and CNN with CoordConv layers (CoordCNN), we tried three combinations of labels: {Focal length, F-number, ISO}, {Focal length, F-number}, {F-number, ISO}, in addition to training on each label.

5.3 Metrics

The mean square error (MSE) is the objective of our optimization. We also used mean absolute error, R^2 score and a custom threshold-based accuracy to evaluate our model and results.

The mean absolute error is defined as $\text{MSE} = \sum |y_{\text{predict}} - y_{\text{true}}|/n$. Comparing with MSE, it emphasizes less on the large prediction errors.

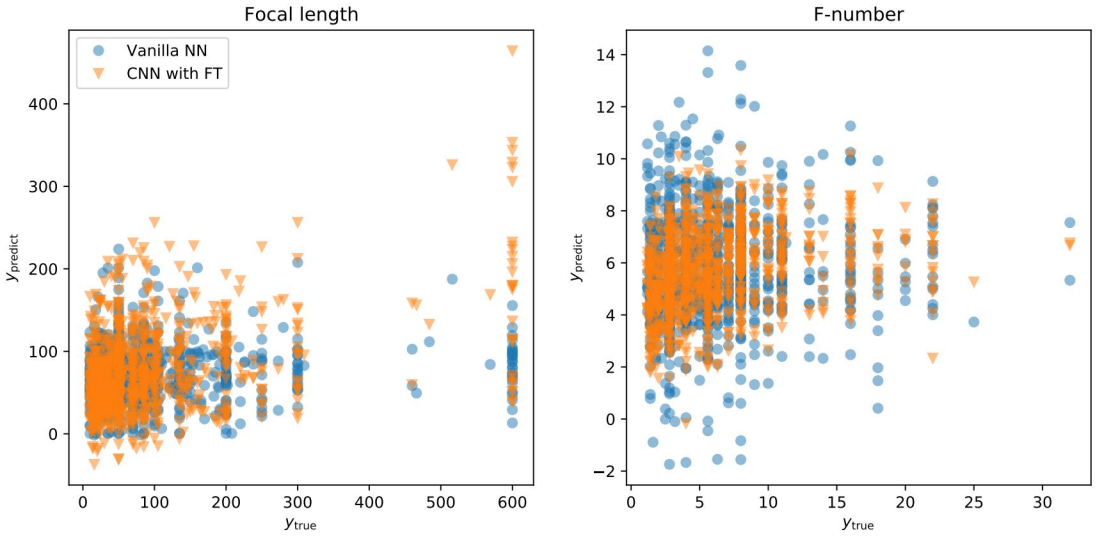

Figure 3: Prediction vs true values on the test set. The overall-best method, CNN with Fourier spectra as input, is compared to the baseline method.

The R_2 score, i.e. coefficient of determination, is defined as $R_2 = 1 - \frac{\sum(y_{\text{predict}} - y_{\text{true}})^2}{\sum(y_{\text{true}} - \bar{y}_{\text{true}})^2}$. It is a statistical measure of how well the regression predictions approximate the real data points. A perfect model would give $R_2 = 1$, and a model that always predict the expectation value $\bar{y}_{\text{true}}$ would give $R_2 = 0$. It would be negative when the model is worse.

Finally, to evaluate the how accurate our model is on individual examples, we defined accuracy as the percentage of examples that satisfy $|y_{\text{predict}} - y_{\text{true}}| <$ threshold. Considering the range of each label, for focal length, aperture and ISO, we chose 20, 1 and 100 as thresholds respectively.

6 Results and Discussion

We present the metrics evaluated on the test set in Table 1. For each method, we tried six setups of labels: {Focal length, F-number, ISO}, {Focal length, F-number}, {F-number, ISO}, {Focal length}, {F-number}, {ISO}, and the results in Table 1 are the best among these six. The best performance on focal length is usually trained with F-number, and the best performance on aperture is usually trained with ISO, for all three methods involving CNN. Additionally, we observe the test loss to start raising later when the training is on multiple labels than on individual labels. We speculate that the second label acts as regularization for the main label that can mitigate overfitting.

Comparing the four methods, CNN generally outperforms the vanilla NN, and replacing plain convolution layers with CoordConv layers showed very marginal improvement. FT-CNN seems to perform the best among the four methods. We try to give an intuitive speculation for the reason as follows. The photographic parameters contributes to the form, such as the high-frequency noise and large-scale aberrations, rather than the objects, and these form-relevant information are better represented as local patterns in the spatial-frequency space, as shown in Fig. 1. Convolutional layers is known to work well on recognizing the local patterns in the feature space, so if the input feature is the Fourier spectra, a CNN might capture the form of the image more easily.

Table 1: Summary of results on the test set with the four methods.

Label	Method	MAE loss	R_2 score	Accuracy (%)
Focal length / mm	Vanilla NN	66.4	-0.073	29
Focal length / mm	CNN	63.5	0.090	27
Focal length / mm	CoordCNN	63.3	0.093	25
Focal length / mm	FT-CNN	62.4	0.175	28
F-number	Vanilla NN	3.56	-0.258	19
F-number	CNN	3.10	0.021	20
F-number	CoordCNN	3.10	-0.008	22
F-number	FT-CNN	2.99	0.049	25
ISO	Vanilla NN	577	-0.020	10
ISO	CNN	597	-0.008	10
ISO	CoordCNN	544	-0.002	17
ISO	FT-CNN	570	-0.093	18

Overall the algorithms have not learned well about these labels, but in comparison on the R_2 scores, they have generally learned better about the focal length and F-number than about ISO. The reason might be the loss of high-resolution noise and details in the image size reduction in our preprocessing. In Fig. 3, we present the scatter plot of the predictions on the test set with FT-CNN. It is clear that FT-CNN outperforms the vanilla NN, and it can qualitatively predict a higher label for images with higher true labels, which are the successful aspects of the FT-CNN.

7 Conclusion and Outlook

In summary, we studied and trained a neural network to estimate the F-number, focal length and ISO of a photo.

Our current results are far from satisfactory, but there is a clear trend of improvement with our implementation of CNN and Fourier transformation. If we have access to larger computing power, we would like to train the model with larger image size and explore more image pre-processes to further develop our model.