

Image Colorization with Deep Convolutional Neural Networks

Jeff Hwang

jhwang89@stanford.edu

You Zhou

youzhou@stanford.edu

Abstract

We present a convolutional-neural-network-based system that faithfully colorizes black and white photographic images without direct human assistance. We explore various network architectures, objectives, color spaces, and problem formulations. The final classification-based model we build generates colorized images that are significantly more aesthetically-pleasing than those created by the baseline regression-based model, demonstrating the viability of our methodology and revealing promising avenues for future work.

1. Introduction

Automated colorization of black and white images has been subject to much research within the computer vision and machine learning communities. Beyond simply being fascinating from an aesthetics and artificial intelligence perspective, such capability has broad practical applications ranging from video restoration to image enhancement for improved interpretability.

Here, we take a statistical-learning-driven approach towards solving this problem. We design and build a convolutional neural network (CNN) that accepts a black-and-white image as an input and generates a colorized version of the image as its output; Figure 1 shows an example of such a pair of input and output images. The system generates its output based solely on images it has “learned from” in the past, with no further human intervention.

In recent years, CNNs have emerged as the de facto standard for solving image classification problems, achieving error rates lower than 4% in the ImageNet challenge [12]. CNNs owe much of their success to their ability to learn and discern colors, patterns, and shapes within images and associate them with object classes. We believe that these characteristics naturally lend themselves well to colorizing images since object classes, patterns, and shapes generally correlate with color choice.



Figure 1. Sample input image (left) and output image (right).

2. Related work

Our project was inspired in part by Ryan Dahl’s CNN-based system for automatically colorizing images [2]. Dahl’s system relies on several ImageNet-trained layers from VGG16 [13], integrating them with an autoencoder-like system with residual connections that merge intermediate outputs produced by the encoding portion of the network comprising the VGG16 layers with those produced by the latter decoding portion of the network. The residual connections are inspired by those existing in the ResNet system built by He et al that won the 2015 ImageNet challenge [5]. Since the connections link downstream network edges with upstream network edges, they purportedly allow for more rapid propagation of gradients through the system, which reduces training convergence time and enables training deeper networks more reliably. Indeed, Dahl reports much larger decreases in training loss on each training iteration with his most recent system compared with an earlier variant that did not utilize residual connections.

In terms of results, Dahl’s system performs extremely well in realistically colorizing foliage, skies, and skin. We, however, notice that in numerous cases, the images generated by the system are predominantly sepia-toned and muted in color. We note that Dahl formulates image colorization as a regression problem wherein the training objective to be minimized is a sum of Euclidean distances between each pixel’s blurred color channel values in the target image and predicted image. Although regression does seem to be well-suited to the task due to the continuous nature of color spaces, in practice, a classification-based approach may work better. To understand why, consider a pixel that

exists in a flower petal across multiple images that are identical, save for the color of the flower petals. Depending on the picture, this pixel can take on various tones of red, yellow, blue, and more. With a regression-based system that uses an ℓ_2 loss function, the predicted pixel value that minimizes the loss for this particular pixel is the mean pixel value. Accordingly, the predicted pixel ends up being an unattractive, subdued mixture of the possible colors. Generalizing this scenario, we hypothesize that a regression-based system would tend to generate images that are desaturated and impure in color tonality, particularly for objects that take on many colors in the real world, which may explain the lack of punchiness in color in the sample images colorized by Dahl’s system.

3. Approach

We build a learning pipeline that comprises a neural network and an image pre-processing front-end.

3.1. General pipeline

During training time, our program reads images of pixel dimension 224×224 and 3 channels corresponding to red, green, and blue in the RGB color space. The images are converted to *CIELUV* color space. The black and white luminance *L* channel is fed to the model as input. The *U* and *V* channels are extracted as the target values.

During test time, the model accepts a $224 \times 224 \times 1$ black and white image. It generates two arrays, each of dimension $224 \times 224 \times 1$, corresponding to the *U* and *V* channels of the *CIELUV* color space. The three channels are then concatenated together to form the *CIELUV* representation of the predicted image.

3.2. Transfer learning

We initialized parts of model with a VGG16 instance that has been pretrained on the ImageNet dataset. Since image subject matter often implies color palette, we reason that a network that has demonstrated prowess in discriminating amongst the many classes present in the ImageNet dataset would serve well as the basis for our network. This motivates our decision to apply transfer learning in this manner.

3.3. Activation function

We use the rectified linear unit as the nonlinearity that follows each of our convolutional and dense layers. Mathematically, the rectified linear unit is defined as

$$f(x) = \max(0, x)$$

The rectified linear unit has been empirically shown to greatly accelerate training convergence [9]. Moreover, it is much simpler to compute than many other conventional activation functions. For these reasons, the rectified linear unit has become standard for convolutional neural networks.

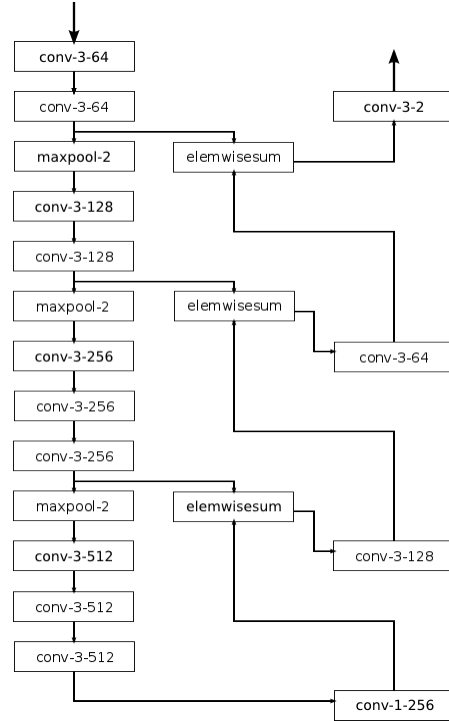


Figure 2. Regression network schematic.

One downside of using the rectified linear unit as the activation function in a neural network is that the model parameters can be updated in such a way that the function’s active region is always in the zero-gradient section. In this scenario, subsequent backpropagated gradients will always be zero, hence rendering the corresponding neurons permanently inactive. In practice, this has not been an issue for us.

3.4. Batch normalization

Ioffe et al introduced batch normalization as a means of dramatically reducing training convergence time and improving accuracy [7]. For our networks, we place a batch normalization layer before every non-linearity layer apart from the last few layers before the output. In our trials, we have found that doing so does improve the training rate of the systems.

3.5. Baseline regression model

We used a regression-based model similar to the model described in [2] as our baseline. Figure 2 shows the structure of this baseline model.

We describe this architecture as comprising a “summarizing”, encoding process on the left side followed by a “creating”, decoding process on the right side

The architecture of the leftmost column of layers is inherited from a portion of the VGG16 network. During this “summarizing” process, the size (height and width) of the

feature map shrinks while the depth increases. As the model forwards its input deeper into the network, it learns a rich collection of higher-order abstract features

The “creating” process on the right column is a modified version of the “residual encoder” structure described in [2]. Here, the network successively upscales the preceding layer output, merges the result with an intermediate output from the VGG16 layers via an elementwise sum, and performs a two-dimensional convolution on the result. The progressive, decoder-like upscaling of layers from an encoded representation of the input allows for the propagation of global spatial features to more-local image regions. This trick enables the network to realize the more abstract concepts with the knowledge of the more concrete features so that the creating process will be both creative and down to earth to suit the input images.

For the objective function in our system, we considered several loss functions. We began by using the vanilla ℓ_2 loss function. Later, we moved onto deriving a loss function from the Huber penalty function, which is defined as

$$L(u) = \begin{cases} u^2 & |u| < M \\ M(2|u| - M) & |u| > M \end{cases}$$

Intuitively, the function is defined piecewise in terms of a quadratic function and two affine functions. For residuals u that are smaller than the threshold M , it follows the ℓ_2 penalty function; for residuals that are larger than M , it reverts to the ℓ_1 penalty function. This feature of the Huber penalty function allows it to extract the best of both worlds between the ℓ_2 and ℓ_1 norms; it can be more robust to outliers while de-emphasizing points the system has fit closely enough to. For our particular use case, this behavior is ideal, since we expect there to be many outliers for colors that correspond to a particular shape or pattern.

3.6. Final classification model

Figure 3 depicts a schematic of our final classification model. The regression model suffers from a dimming problem because it minimizes some variant of the ℓ_p norm, which motivates the model to choose an average or intermediate color when multiple distinct color choices are possible. To address this issue, we remodeled our problem as a classification problem.

In order to perform classification on continuous data, we must discretize the domain. The targets U and V from the $CIELUV$ color space take on values in the interval $[-100, 100]$. We implicitly discretize this space into 50 equi-width bins by applying a binning function (denoted $\text{bin}()$) to each input image prior to feeding it to the input of the network. The function returns an array of the same shape as the original image with each U and V value mapped to some value in the interval $[0, 49]$. Then, instead

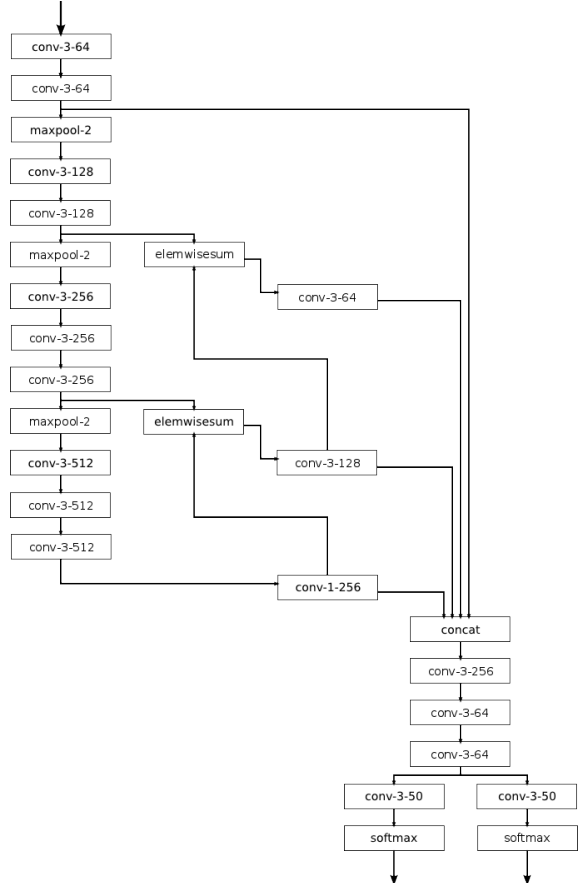


Figure 3. Classification network schematic.

of directly predicting numeric values for U and V , the network outputs two separate sets of the most probable bin numbers for the pixels, one for each channel. We used the sum of cross-entropy loss on the two channels as our minimization objective.

In terms of the architecture, we introduced a concatenation layer `concat`, which is inspired by segmentation methods. Combining multiple intermediate feature maps in this fashion has been shown to increase prediction quality in segmentation problems, producing finer details and cleaner edges[4]. Although there is no explicit segmentation step in our setup, this approximate approach allows our system to minimize the amount of visual noise that is generated along object edges in the output image.

We experimented with placing various model structures between the concatenation layer and output. In our final model, the concatenation layer is followed by three 3×3 convolutional layers, which are in turn followed by the final two parallel 1×1 convolutional layers corresponding to the U and V channels. These 1×1 convolutional layers act as the fully-connected layers to produce 50 class scores for each channel for each pixel of the image. The classes with the largest scores on each channel are then selected as the

Dataset	Training	Test
McGill	896	150
MIT CVCL	361	50
ILSVRC 2015 CLS-LOC	12486	494
MIRFLICKR	7500	1000

Table 1. Number of training and test images in datasets.



Figure 4. Sample images from the MIT CVCL Open Country dataset.

predicted bin numbers. Via an un-binning function, we then convert the predicted bins back to numerical U and V values using the means of the selected bins.

4. Dataset

We tested our system on several datasets; Table 1 provides a summary of the datasets we considered.

The MIT CVCL Urban and Natural Scene Categories dataset contains several thousand images partitioned into eight categories [10]. We experimented with 411 images in the ‘‘Open Country’’ category to measure our system’s ability to generate images pertaining to a specific class of images; Figure 4 shows some sample images from the dataset.

To gauge how well our system generalizes to diverse images, we experimented with larger datasets encompassing broader classes of photos. The McGill Calibrated Colour Image Database contains more than a thousand images of natural scenes organized by categories [11]. We chose to experiment with samples from each of the categories. The ILSVRC 2015 CLS-LOC dataset is the dataset used for the ImageNet challenge in 2015 [12]. We sampled images from the following categories: spatula, school bus, bear, book shelf, armor, kangaroo, spider, sweater, hair dryer, and bird. The MIRFLICKR dataset comprises 25000 Creative Commons images downloaded from the community photo-sharing website Flickr [6]. The images span a vast range of categories, artistic styles, and subject matter.

We preprocess each image in our dataset prior to forwarding it to our network. We scale each image to dimensions of $224 \times 224 \times 3$ and generate a grayscale version of the image of dimensions $224 \times 224 \times 1$. Since the input of our network is the input of the ImageNet-trained VGG16, which expects its input images to be zero-centered and of dimensions $224 \times 224 \times 3$, we duplicate the grayscale im-

age three times to form a $(224 \times 224 \times 3)$ -sized image and subtract the mean R, G, and B value across all the pictures in the ImageNet dataset. The resulting final image serves as the black-and-white input image for the network.

5. Experiments

5.1. Evaluation metrics

For regression, we quantify the closeness of the generated image to the actual image as the sum of the ℓ_2 norms of the difference of the generated image pixels and actual image pixels in the U and V channels:

$$L_{reg.} = \|U_p - U_a\|_2^2 + \|V_p - V_a\|_2^2$$

Likewise, for classification, we measure the closeness of the generated image to the actual image by the percent of binned pixel values that match between the generated image and actual image for each channel U and V :

$$\text{Acc.}_U = \frac{1}{N^2} \sum_{(i,j)}^{(N,N)} \mathbf{1}\{\text{bin}(U_p) = \text{bin}(U_a)\}$$

$$\text{Acc.}_V = \frac{1}{N^2} \sum_{(i,j)}^{(N,N)} \mathbf{1}\{\text{bin}(V_p) = \text{bin}(V_a)\}$$

, where $\text{bin} : \mathbb{R} \rightarrow \mathbb{Z}_{50}$ is the color binning function described in Section 3.6. We emphasize that classification accuracy alone is not the ideal metric to judge our system on, since the accuracy of color matching against target images does not directly relate with the aesthetic quality of an image. For example, for a still-life painting, it may be the case that virtually none of the colors actually match the corresponding real-life scene. Nevertheless, the painting may still be regarded as being artistically impressive. We, however, report it as one possible measure because we do believe that there exists some correlation between the two.

We can also apply these formulae to the regression results to compare with the classification results.

Finally, we track the percent deviation in average color saturation between pixels in the generated image and in the actual image:

$$\text{Sat. diff.} = \frac{\left| \sum_{(i,j)}^{(N,N)} S_{p_{ij}} - \sum_{(i,j)}^{(N,N)} S_{a_{ij}} \right|}{\sum_{(i,j)}^{(N,N)} S_{a_{ij}}}$$

Generally speaking, the degree of color saturation in a given image strongly influences its aesthetic appeal. Ideally, then, the saturation levels present in the training images should be replicated at the system’s output, even when the exact hues and tones are not matched perfectly. This metric allows us to quantify the faithfulness of this replication.

5.2. Experiment setup and alternative structures

Our networks were implemented with Lasagne [1] and were trained on an AWS instance running a NVIDIA GRID K520 GPU.

We started by trying to overfit our model on a 270-image random subset of ImageNet data.

To determine a suitable learning rate, we ran multiple trials of training with minibatch updates to see which learning rate yielded faster convergence behavior over a fixed number of iterations. Within the set of learning rates sampled on a logarithmic scale, we found that a learning rate of 0.001 achieved one of the largest per-iteration decreases in training loss as well as the lowest training loss of the learning rates sampled. Using that as a starting point, we moved to with the entire training set. With a hold-out proportion of 10% as the validation set, we observed fastest convergence with a learning rate of 0.0003

We also experimented with different update rules, namely Adam [8] and Nesterov momentum [14]. We followed the recommended $\beta_1 = 0.9$ and $\beta_2 = 0.99, 0.999$. For Nesterov Momentum, we used a momentum of 0.9. Among these options, the Adam update rule with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ produced slightly faster convergence than the others, so we used the Adam update rule with these hyperparameters for our final model.

In terms of minibatch sizes, we experimented with batches of four, six, eight and twelve images based on network architecture. Some alternative structures we tried required less memory usage, so we tested those with all four options. The model shown in Figure 3, however, is memory-intensive. Due to the limited access of computational resource, we were only able to test it with batch sizes of four and six with the GPU instance. Nevertheless, this model with a batch size of six demonstrated faster and stabler convergence than the other combinations.

For weight initialization, since our model uses the rectified linear unit as its activation function, we followed the Xavier Initialization scheme proposed by [3] for our original trainable layers in the decoding, “creating” phase of the network.

We also developed several alternative network structures before we arrived at our final classification model. The following are some design elements and decisions we weighed:

1. Multilayer aggregation – elementwise sum versus concatenation: we experimented with performing layer aggregation using an elementwise sum layer in place of the concatenation layer. An elementwise sum layer reduces memory usage, but in our experiments, it turned out to harm training and prediction performance.
2. Presence or absence of residual encoder units: a

residual encoder unit refers to a joint convolution-elementwise-sum step on a feature map in the “summarizing” process and an upscaled feature map in the “creating” process, as described in Section 3. We experimented with trimming away the residual encoder units and applying aggregation layers directly on top of the maxpooling layers inherited from VGG16. However, the capacity of the resulting model is much smaller, and it showed poorer quality of results when overfitting to the 300-image subset.

3. The final sequence of convolutional layers before the network output: we experimented with one and two convolutional layers with various depths, but the three-layer structure with the current choice of depths yielded the best results.
4. Color space: initially, we experimented with the HSV color space to address the under-saturation problem. In HSV, saturation is explicitly modeled as the individual S channel. Unfortunately, the results were not satisfying. Its main issue lies in its exact potential merit: since saturation is directly estimated by the model, any prediction error became extremely noticeable, making the images noisy.

5.3. Results and discussion

Figure 5 depicts two sets of regression and classification network outputs along with their associated black-and-white input images. The model that generated these images was trained on the MIT CVCL Open Country dataset.



Figure 5. Test set input images (left column), regression network output (center column), and classification network output (right column).

The regression network outputs are somewhat reasonable. Green tones are restricted to areas of the image with foliage, and there seems to be a slight amount of color tinting in the sky. We, however, note that the images are severely desaturated and generally unattractive. These results are expected given their similarity to Dahl’s sample outputs and our hypothesis.

In contrast, the classification network outputs are amazingly colorful yet realistic. Colors are lively, nicely saturated, and generally tightly restricted to the regions they correspond to. For example, for the top image, the system managed to infer the reflection of the sky in the water and colored both with bright swaths of blue. The foliage on the mountain is colored with deep tones of green. Overall, the output is highly aesthetically pleasing.

Note, however, that there exists a noticeable amount of noise in the classification results, with blobs of various colors interspersed throughout the image. This may result from several factors. It may be the case that the system discretizes the U and V channels into bins that are too large, which means that image regions that contain color gradients may appear choppy. Also, the system performs classification on a per-pixel basis without explicitly accounting for the color values of surrounding pixels. Finally, it may simply be the case that a particular patch containing a shape or pattern has many possible color matches in the real world, and the system does not have the capacity to choose a particular color consistently. For instance, considering the second classification-network-generated image, we see that the brush in the foreground takes on varying shades of green, yellow, and brown. In the wild, grass does take on many colors depending on seasonality and geography. With the little context present in the corresponding black and white image, it can be difficult for even a human to discern the most probable color for the grass.

System	Acc. (U)	Acc. (V)	Sat. diff.
Classification	34.64%	24.14%	6.5%
Regression	12.92%	19.02%	85.8%

Table 2. Performances on MIT CVCL Open Country test set.

Despite the aforementioned flaws in its output, all in all the classification system performs very well. Via our evaluation metrics, we can quantify the superiority of the classification network over the regression network for this particular dataset (Table 2). For the landscape data test set, using the classification network, we find that the U and V channel prediction accuracies are 0.3464% and 0.2414%, respectively, and that the average percent difference in saturation is 6.5%. In comparison, using the regression network on the same test set, we find that the U and V channel prediction accuracies are 0.1292% and 0.1902%, and that the average percent difference in saturation regression is a whopping 85.8%. Evidently, not only is the classification network able to correctly classify the color of a particular pixel more effectively, but it is also much more likely to predict colors that match the saturation levels of images in the training set.

In Figure 6, we present sample colored test images along with the black and white input. The model generally yields



Figure 6. Sample test set input and output from ImageNet.

more convincing images on nature themes. It is the case because the candidate colors for objects in nature are more well defined than some man-made objects. For example, the sky might be blue, grey or even pink at the a sunset, but it is almost never green. However, chair cushions, for example, may bear almost any color. The model needs to see more cushion images than sky images before it learns to color it nicely. On the bottom right image, for example, the table is colored partially red. Since we did not sample the table class, there is not many images with tables in the training set for the model to learn how to color table objects properly.



Figure 7. Sample outputs exhibiting color inconsistency issue.

The main challenge our model faces is inconsistency in colors within individual objects. Figure 7 shows two test input and output pairs that suffer from this issue. For the first example, the model colored parts of the sweatshirt red and other parts of it grey. As human beings, we can imagine a sweatshirt being red or grey as a whole. Our current system-on the other hand-makes one color prediction on each pixel, and hopefully the close-by pixels have similar color assignment. However, it is not always the case. Even though local regions of small sizes are examined together given the nature of convolutional layers, there is no explicit enforcement on the object level. We experimented with applying a Gaussian smoothing on the class scores to address this issue. This kind of smoothing performed only slightly better. Unfortunately, it introduced another issue: it significantly increased visual noise along object edges. Accordingly, we left out the smoothing in our final model.

Figure 8 shows two under-colored images. As we observed earlier, man-made objects with a large intra-domain color variation are generally more challenging. The model was not able to give a good prediction for the sweater, most likely because of the wide range of color choices. Upon



Figure 8. Under-colored output examples.

close examination, we noticed that the model even painted part of it slightly green. Similarly for the crowd picture, the model did not provide much color to non-white clothes.

6. Conclusion and future work

Through our experiments, we have demonstrated the efficacy and potential of using deep convolutional neural networks to colorize black and white images. In particular, we have empirically shown that formulating the task as a classification problem can yield colorized images that are arguably much more aesthetically-pleasing than those generated by a baseline regression-based model, and thus shows much promise for further development.

Our work therefore lays a solid foundation for future work. Moving forward, we have identified several avenues for improving our current system. To address the issue of color inconsistency, we can consider incorporating segmentation to enforce uniformity in color within segments. We can also utilize post-processing schemes such as total variation minimization and conditional random fields to achieve a similar end. Finally, redesigning the system around an adversarial network may yield improved results, since instead of focusing on minimizing the cross-entropy loss on a per-pixel basis, the system would learn to generate pictures that compare well with real-world images. Based on the quality of results we have produced, the network we have designed and built would be a prime candidate for being the generator in such an adversarial network.

References

- [1] Lasagne. <https://github.com/Lasagne>, 2015.
- [2] R. Dahl. Automatic colorization. <http://tinyclouds.org/colorize/>, 2016.
- [3] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [4] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–456, 2015.
- [5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
- [6] M. J. Huiskes and M. S. Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43. ACM, 2008.
- [7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [8] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [10] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.
- [11] A. Olmos et al. A biologically inspired algorithm for the recovery of shading and reflectance images. *Perception*, 33(12):1463–1473, 2004.
- [12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.
- [13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [14] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147, 2013.