SmallSEOTools

PLAGIARISM SCAN REPORT

Words	783	Date	June 21,2021
Characters	5094	Excluded URL	

0%	100%	0	45
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

6.4 METHODOLOGY
6.4.1 Image Color space
An RGB image is essentially a rank 3 tensor of height, width and color where the last axis contains the color data of our image. The data is represented in RGB color space which has 3 numbers for every pixel indicating the amount of Red, Green, and Blue values the pixel has. Figure 6.1 shows that in the left part of the “main image” has blue color so the blue channel of the image has higher values and turns dark.
In L*a*b color space, we have three numbers for each pixel but these numbers have different meanings. The first channel, L, encodes the Lightness of each pixel and when we visualize this channel it appears as a black and white image. The *a and *b channels encode how much green-red and yellow-blue each pixel is, respectively. In the following image you can see each channel of L*a*b color space separately.
In most of the papers listed in the Literature Survey, the L*a*b color space is widely used instead of RGB to train the models. Intuitively, to the train a model for colorization, we should give it a grayscale image and we hope that it colors it. In the L*a*b colorspace, the model is fed the L channel, which is essentially a grayscale image, and we perform computations to predict the other two channels (*a and *b). After the predictions, we concatenate the channels and get a colorful image. In case of RGB, there is a need to explicitly convert the three channels down to 1 to make it a grayscale image and then feed it to the network hoping that it predicts three numbers per pixel. This is an unstable task due to sheer increase in volume of combinations from two numbers to three.
We train models using both color spaces and empirically show the significance of the L*a*b colorspace.
6.4.2 Transferred learning and model tweaking
Isola et al. (2018) proposed a general solution to many image-to-image translation tasks. We propose a similar methodology as proposed in the paper with a few minor tweaks, so as to significantly reduce the amount of training data needed and minimize the training time to achieve similar results. Initially, we use a U-net for the generator. The encoder of our U-net is a pre-trained ResNet-18 network with half of it’s layers clipped off so as to get feature abstractions of the middle layers. Ledig et al. (2017) showed that training the generator separately in a supervised, deterministic manner helps it generalize the mapping from the input space to outputs. The idea solves the problem of ”The blind leading the blind” that is persistent in most image-to-image translation tasks to date.
We decide to use the Common Objects in Context (COCO) dataset to pre-train our generator independently using the imagenet weights. This training will be supervised and the loss function used is mean absolute error or the L1 Norm from the

target image. Even though trained deterministically,the problem of rectifying incorrect predictions still persists due to constraints over the convergence of loss metric.
To combat this, we use this trained generator and train it once again in an adversarial fashion to generalize it further. We hypothesize that re-training with an adversarial will further rectify the subtle color differences that mae couldn't solve.
We use a pre-trained ResNet-50 with imagenet weights as our discriminator with last few layers clipped off. The discriminative network used is something called a "Patch Discriminator" proposed by Isola et al. (2018). In a vanilla discriminator proposed by Goodfellow et al. (2014), the network produces a scalar output, representing the probability that the data x belongs to the input distribution and not the generator distribution pg. Isola et al. (2018) proposed a modification to the discriminative network so as to produce, instead of a single scalar, a vector of probabilities representing different localities of the input distribution (image) x. For instance, a discriminator producing a 3030 vector represents the probabilities every receptive field that is covered by the output vector. Thus, we can localize the corrections in the image that the generator should make.
Finally, we use the trained generator and trained discriminator and fine-tune it to fit it on our data which is rather small in size compared to the previous datasets. We train these networks in an adversarial fashion using the conditional GAN objective function (Isola et al.; 2018) and couple it with some noise introduced as the L1 norm of generated image tensor to the target image tensor. The reason behind this is to train the generator using an adversarial component while trying to minimize the Manhattan distance between the generator output and target vector space.

Sources	Similarity