Small-Scale Automatic Image Colorization

Arin Champati, Brendan Houle, Byron Zhang

Princeton University

Abstract. In recent years, convolutional neural networks (CNN) have been utilized frequently to solve the task of colorizing grayscale images. However, past approaches have relied on large image datasets to produce plausible results. In this paper, we survey several different approaches that leverage deep learning on a small dataset to re-implement different colorization CNNs, examine the effects of different color spaces, loss functions, and network architectures, then assess their ability to generalize.

1 Introduction

Given a 1-channel grayscale image, only a small fraction of information on the true coloring of the 3-channel, colored image remains. This makes reconstruction of the colorized image particularly challenging. However, when humans see a black-and-white photo, they can usually infer what color it should be. For example, skies are most often blue, golden retrievers are often yellow, and leaves are often green. This can be inferred even if these objects are grayscaled in images. While a lot of information related to color may have been lost in the grayscaling, a lot of the semantic information still remains, which offers us a path torward to retrieving the original color of the image.

Since learning semantic information from natural images can greatly reduce the difficulty and complexity of image colorization, convolutional neural networks have become a popular tool to deal with this task. However, training such neural networks requires optimal design choices and careful selection of hyperparameters in order to yield the best results. Previous works on this matter have relied on very large, complex databases such as ImageNet [3] with over one million images. Because of the high amount of data, examining different design choices would likely be extremely time consuming.

In this paper, we perform the task of image colorization on a small dataset (CIFAR-10 [6]) and give an overview on the effects of a broad set of design choices, such as color spaces, loss functions, network architectures, transfer learning, and various hyperparameters associated with these choices. Although we expect to not obtain optimal results comparable to the state-of-art, we hope to show that expected trends observed in previous approaches are reproducible on a small dataset, which may facilitate future experimentation in this area.

2 Related Work

Earlier methods such as [9], [5] use example-based approaches, which learn the correspondence between a user-provided example (reference) image and the actual grayscale input. Although these methods generate impressive results on images of common natural landscapes, the quality of results depends highly on the similarity between the reference color image and the input image. Because example-based approaches suffers from requiring careful selection of reference input, more recent methods proposed to use deep convolutional neural network to allow full automation in the image coloring process.

2.1 Regression-based Colorization

One benefit of using convolutional neural networks for colorization is their ability to provide high-level understanding of images. To leverage higher-level object semantics, [2] proposed to use a CNN to extract features with increasing receptive fields, concatenating them to predict per-pixel colors on a fully connected neural network; [4] trains an object classification CNN along a colorization network, fusing them to produce final colored output. Since color spaces are continuous, a natural selection for the objective function in [2], [4] are regression-based. However, [1] has shown that standard regression loss tends to favor conservative color predictions. This leads to poorly saturated output images, leaving a need for a more robust colorization method favoring colors that are properly saturated.

2.2 Classification-based Colorization

Novel approaches, such as [1], have proposed to quantize continuous color spaces into bins of equal sizes. With this method, the colorization problem becomes a classification problem. A CNN trained with the cross-entropy loss function predicts a probability distribution of the quantized color bins for each pixel, which is then used to interpolate a final image in the continuous color spaces. [7], [10] was able to obtain more plausible results with the classification approach by training with a larger dataset (ImageNet [3] with 1.2 million images). However, these approaches still produce noisy predictions when processing complex images with multiple foreground objects, since its convolutional layers have shown to retain only single-object semantics.

3 Methods

3.1 Data

For this paper, we decided to explore coloring images on a small dataset. At first, we thought to use a subset of ImageNet [3] rescaled to 64x64 images, however, due to computational limitations, we needed a dataset with smaller images. So we decided to use CIFAR-10 [6] (See Figure 1), which consists of 60000 32x32

color images spanning 10 different classes (6000 images per class). Because there are only 10 classes and comparably many samples per class, CIFAR-10 provided itself as a good dataset for the purpose and demonstration of our system.

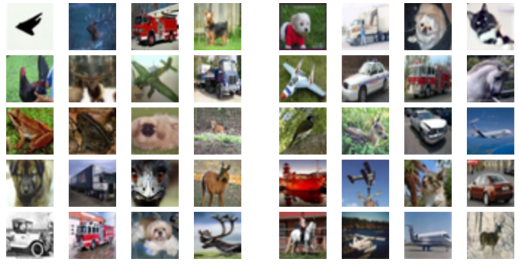

Fig. 1: CIFAR-10 Example Images

3.2 Color Space

We cast the colorization problem into the RGB, $CIELab$, Yuv color spaces and compare the results from each space's outputs. For the RGB color space, the task is to output values of the three channels for every pixel of an image given its grayscale representation.

For the Lab and Yuv color space, the task is to predict only values of the ab and uv channels, respectively, because the L (Luminance) and Y (Luma) channels can already be represented directly by grayscale values from the input. These two-channel predictions are then concatenated onto a modified grayscale image to produce the final prediction for image colorization.

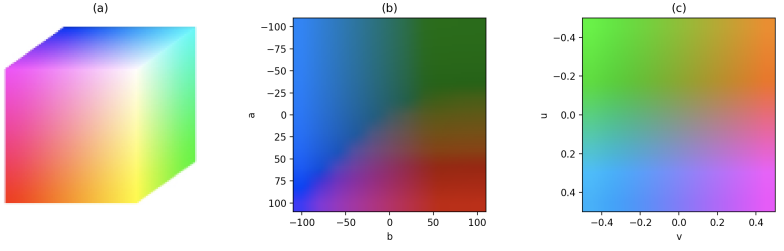

Fig. 2:
(a) Cube illustrating the RGB color space $R \in [0, 255]$, $b \in [0, 255]$, $B \in [0, 255]$.
(b) Visualization of CIELab color space, $a \in [-110, 110]$, $b \in [-110, 110]$, $L = 30$.
(c) Visualization of Yuv color space, $u \in [-0.5, 0.5]$, $v \in [-0.5, 0.5]$, $Y = 0.5$.

3.3 Loss Function

As mentioned earlier, there are two common loss functions proposed by recent research on image colorization: regression loss and categorical cross-entropy loss. Here, we experiment with minimizing both objectives.

Regression Loss We train a CNN on the average L2 regression loss, also known as Mean Squared Error (MSE) in all color spaces from Figure 2, which can be modeled by the equation,

$$MSE(X, \hat{X}) = \frac{1}{hw} \sum_{h,w} ||X_{hw} - \hat{X}_{hw}||^2 \tag{1}$$

where X is the ground truth image, and $\hat{X}$ is the output prediction, h, w are the coordinates of a particular pixel. Previous work has shown that the use of regression-based methods in color prediction often leads to poorly saturated and potentially unrealistic results.

An intuitive explanation for this failure is offered by [1]. Many similar objects often have very different colorizations. For example, two similarly shaped birds might have very different colorizations; two nearly identical balloons might be colored completely differently. A regression-based approach such as mean squared error would try to minimize the Euclidean distance between prediction and ground trugh, choosing the mean color of objects with the same descriptors. In the case of balloons, the mean color of all of the balloons might be gray or brown, a coloring that does not seem plausible to human eye. Thus, we expect the classification-based approach to provide us with more reasonable colorings.

Cross-entropy Loss To train a CNN on cross-entropy loss, we take the approach as suggested by [1], [10], which quantizes the ab and uv color spaces into Q bins of equal sizes. We then make the network accept an $H \times W \times 1$ grayscale image, and outputs a $H \times W \times Q$ block, where $H = W = 32$ according to the dimensions of images in CIFAR-10. Each pixel (h, w) in the output containing a vector $p \in \mathbb{R}^Q$ that denotes the probability distribution of quantized color classes at that pixel. To obtain this binned ground truth image, we label each one of the uv and ab pixels with the it closest bin out of the Q different computed bins. The loss function $L_{cl}(\hat{Z}, Z)$ is given by:

$$L_{cl}(\hat{Z}, Z) = -\sum_{h,w} \sum_{q} Z_{h,w,q} \log(\hat{Z}_{h,w,q}) \tag{2}$$

where $\hat{Z}$ represents the predicted probability distribution for a pixel, Z is a one-hot encoding vector of the ground truth color bin, and $0 <= q < Q$. We expect the cross-entropy loss function to produce more visually plausible color predictions than regression loss functions, because it sacrifices exact Euclidean distance between predicted pixel and ground truth to capture the multi-modal

nature of the colorization problem. A coloring that classifies a balloon as red will look more realistic than a model that classifies a balloon as gray due to the failures of regression. Note that while these colorizations might be more visually plausible, they won't necessarily match the ground truth.

We chose not to implement neural networks with cross-entropy loss in the RGB color space for the classification task. Predicting in RGB requires quantization of a 3-dimensional color space, while predicting in Lab and Yuv requires quantization of 2-dimensional color spaces because the L and Y channels are already known from the input. The same number of bins would capture more detailed information in a 2-dimensional space than a 3-dimensional one, which is why we chose to experiment with the cross-entropy loss function on only the Lab and Yuv color spaces.

3.4 Class Rebalancing

Even classification-based methods can sometimes lead to desaturated results due to a class-imbalance problem in the quantized color spaces. [10] pointed out that the distribution of ab values in natural images is strongly biased toward low ab values due to the presence of clouds, dirt, and walls. Because lower ab values tend to produce desaturated colors, which appear at frequencies orders of magnitudes higher than other saturated values, both L2 regression and cross-entropy loss functions are strongly dominated by desaturated ab values values. This phenomenon is especially magnified on CIFAR-10 because it contains many images with white background, and white has $a = b = 0$, as shown in 3. Similar phenomenon can be observed in the uv color gamut because lower uv values also bring desaturated colors.

To account for class-imbalance, [10] proposed to add in a class-rebalancing term in the cross-entropy loss function, making it

$$L_{cl}(\hat{Z}, Z) = -\sum_{h,w} v(Z_{h,w}) \sum_{q} Z_{h,w,q} \log(\hat{Z}_{h,w,q}) \tag{3}$$

where $v(Z_{h,w})$ is the class-rebalancing weight for prediction at pixel location (h, w) based on its closest ab or uv bin, and $\hat{Z}_{h,w,q}$ is the predicted probability of the pixel belonging to bin q, so

$$v(Z_{h,w}) = \mathbf{w}_{q*} \qquad q* = \arg\max_{q}(Z_{h,w,q}) \tag{4}$$

To compute the weights $\mathbf{w}_q$ for each quantized bin q in the quantized ab and uv space, we first obtain the empirical probability distribution in the quantized color spaces from colors of every pixel in the CIFAR-10 dataset, then smoothen this distribution using a Gaussian kernel with $\sigma = 3$. We then mix this probability distribution with a uniform distribution to form a new distribution, take its reciprocal as the weights for class-rebalancing, finally renormalize the weights such that the expected value of the weights equals to one. The generation process

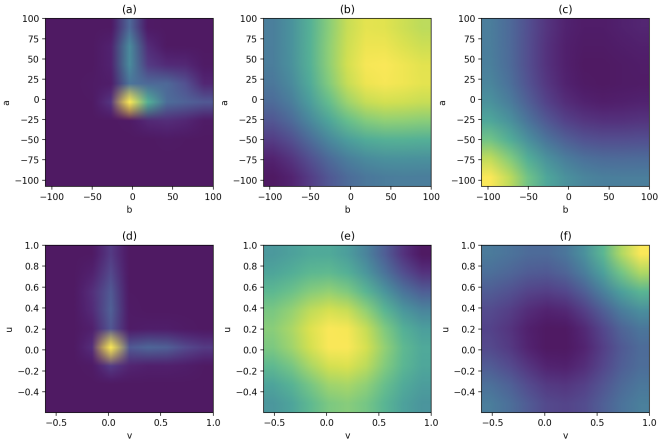

Fig. 3: Brighter regions denote higher probabilities & weights.
(a): Empirical probability distribution of ab values in CIFAR-10, bin size = 81.
(b): Smoothened probability distribution of (a), $\sigma = 3$.
(c): Weights obtained for rebalancing ab space using distribution from (b).
(d): Empirical probability distribution of uv values in CIFAR-10, bin size = 81.
(e): Smoothened probability distribution of (d), $\sigma = 3$.
(f): Weights obtained for rebalancing uv space using distribution from (e)

mentioned above is modeled by:

$$\mathbf{w}_q \propto ((1-\lambda)p_q + \frac{\lambda}{Q})^{-1} \qquad \mathbb{E}[w] = \sum_q p_q w_q = 1 \tag{5}$$

where p_q is the smoothened probability of bin q in ab or uv space. λ and Q are hyperparameters, which we found to work best at $\lambda = 0.2$, $Q = 81$. This result will be further discussed in section 4. With the class-rebalancing procedure proposed by [10], we expect to observe more vibrant outputs because wrong predictions on rarer, saturated colors would be penalized more than wrong predictions on desaturated colors in the new loss function.

3.5 Network Architecture

Throughout our experimentation, we utilized three architectures as baselines, with modifications for certain tasks.

Regression Both figures 4 and 5 outline the models we used for regression purposes. Note that the model diagrams both display the architectures for predicting in the Lab or Yuv space, where the model only predicts two channels (ab or uv). The input (L or Y channel) is concatenated to the output to provide

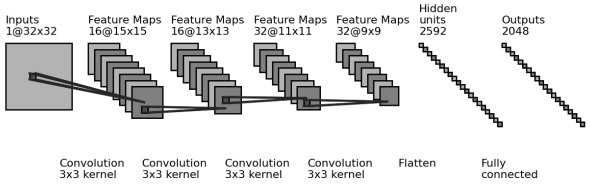

Fig. 4: Baseline Regression Model

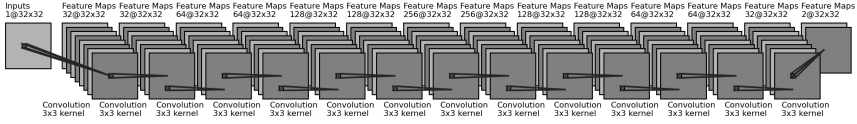

Fig. 5: Convolution Only Regression Model

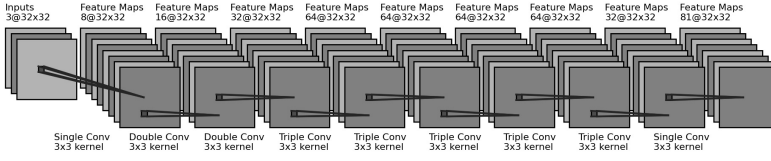

Fig. 6: Convolution Only Classification Model

a full three channel prediction. When predicting in *RGB* space, the model will output predictions for all three channels (details below). Both models also learn to reduce the mean squared L2 distances between the ground truth images and predicted images.

Figure 4 refers to the baseline model utilized for regression purposes. This model learns multiple small feature maps, and the neural network learns a flattened representation of the output image. If predicting within the *Lab* or *Yuv* colorspace, the output vector is reshaped to be the desired image shape ($32 \times 32 \times 3$). Note that between each convolution is a batch normalization layer preceded by ReLU activation.

Figure 5 refers to a more complex, deeper model utilized for regression purposes. This model stacks convolutions in blocks, similar to the VGG16 [8] architecture. After every two or three convolutions (the specific block sizes are captioned in the diagram), there is a batch normalization layer preceded by ReLU activation. Note that there is no neural network to rescale the image size, instead, there is padding added such that the spatial dimensions are preserved.

Classification Figure 6 refers to the model utilized for classification purposes. This architecture bases itself off of VGG16 [8] and the one proposed by "Colorful Image Colorization" [10]. Our dataset is smaller and has less variation than in "Colorful Image Colorization" (which was trained on ImageNet), so this architecture includes less parameters in total (due to less convolutional blocks and a

less amount of filters). Additionally, there is no downsampling or upsampling in this model, due to the fact that the size of the input is relatively small (32×32). As in Figure 5, after every two or three convolutions (the specific block sizes are captioned in the diagram), there is a batch normalization layer preceded by ReLU activation. The last two 64 depth blocks contain a trous (Dilated) convolutions - convolutions that expand the receptive field without loss of resolution. The idea is that, not only do local features contribute to image colorization, but the global composition of an image contributes as well. Finally, the model outputs an image whose depth correlates to the number of bins, where each $1 \times 1 \times Q$ vector represents a probability distribution for each of the Q bins for that pixel.

Transfer Learning In some experiments, we decided to use pretrained features from VGG16 trained on ImageNet [8], [3] as the first two layers of the network. VGG16 downsamples with Pooling Layers, which appear from the third layer onwards, so we decided to extract features from only the first two convolutional layers of VGG. Pooling layers reduce dimensionality, and as we will explore in the section 4 of this paper, dimensional reduction is not desired for the colorization task. We used the first two layers of VGG16 as either a feature extractor or a weight initializer, more details to follow.

3.6 Training

Each model was trained with the Adam optimizer with learning rates ranging from 1e-3 to 5e-3. The models were trained on a Google Colaboratory TPU instance, with a batch size of 1024 (as this is optimal for TPUs), and each epoch took 7 seconds to train. For classification, the models generally converged within 70 epochs, and for regression, the models converged within 100 epochs. Note that the models were all trained on an 80/20/20 train/test/validation split. In order to increase the variation in the data for better generalizability, we utilized a simple data augmentation by performing left-right mirroring on all of the training images and appending these augmented images to the dataset.

3.7 Output Retrieval

Regression If operating in the RGB color space, the output of the CNN are the desired channels. If operating in Lab or Yuv color spaces, we first normalize the input grayscale images to fit the range of the L and the Y channels, then stack the $H \times W \times 1$ input with the $H \times W \times 2$ output ab or uv channels to form the final three-channel image.

Classification Because the output of the classification network is a probability distribution, we need a mapping $F(\hat{Z})$ that maps the $H \times W \times Q$ output to $\hat{Y}$, the final $H \times W \times 2$ output values for the learned ab and uv channels. A natural choice of generating results in classification problems is simply taking the mode of the probability distribution as output. While this produces more vibrant colors, the

Fig. 7: As expected, as T increases, the color of the image becomes more desaturated with less noise.

output is often noisy and spatially inconsistent due to the quantization of the color space, since each bin can include a wide range of potential colors. Another possible approach would be taking the expected value of the distribution, though this would average the colors to produce a desaturated, sepia tone.

Therefore, as proposed by [10], we use a function f_T inspired by simulated-annealing that shifts the output softmax distribution with a temperature parameter T, which allows us to weigh in on both mode and expected value of the distribution, defined by the following:

$$f_T(z) = \frac{exp(log(z)/T)}{\sum_q exp(log(z_q)/T)} = \frac{z^{1/T}}{\sum_q z_q^{1/T}} \tag{6}$$

Finally, we map the probability distribution to a final output by taking the expected value of the annealed distribution:

$$F(\hat{Z}_{h,w}) = \mathbb{E}[Y] = f_T(\hat{Z}_{h,w}) \cdot Y_q \tag{7}$$

where $f_T(\hat{Z}_{h,w})$ is the annealed distribution of $\hat{Z}_{h,w}$ and Y_q is the $[a, b]$ values corresponding to bin q. Observe that at $T = 1$, the distribution $\hat{Z}$ is unchanged, and as $T \to 0$, $\hat{Z}$ becomes a one-hot encoding vector with the mode of the distribution having probability 1. We take the suggestion offered by [10] and found that $T = 0.38$ gives decent results that are not too noisy, but also not too dull. See Figure 7 for a demonstration.

4 Experiments

4.1 Evaluation Metrics

In the below subsections, we will explore the evaluation metrics used to quantify image colorization performance between different experiments.

Threshold Accuracy To provide a low level metric that compares pixel values to ground truth, we provide a threshold accuracy metric. Similar to "Colorful Image Colorization"[10], we compute the percentage of predicted pixel values that fall within some threshold distance from the ground truth for every example in the testing set. We then create a cumulative mass function across every

discrete pixel value possibility and calculate the normalized integral. However, unlike [10], we decided to evaluate the distances in the RGB color space as opposed to the ab channels in the Lab color space. We chose to do this because in our classification models, we trained our models in the Lab and Yuv color space, and thought to use RGB to evaluate generalizability and reduce bias in our evaluation metric.

Regression on L2 loss is expected to perform the best on this metric due to its nature of reducing prediction error. Classification and providing plausible image colorization by predicting a probability distribution is expected to perform slightly below the regression model due to its nature - it is not attempting to reduce the loss on pixel accuracy, instead, it attempts to learn the distribution of color (which often will correlate to the ground truth values). Note that simply predicting gray will result in a high value for this metric (see Table 1). This is likely due to the fact that the natural color distribution of RGB space in natural photos is very unsaturated.

Additionally, note that while this metric is a low-level 'brute-force', the patterns observed within this metric are the same patterns that visually follow, at least for this dataset. Usually, the classification threshold accuracies would be much lower, due to the fact that they are *plausible* colors. However, in the CIFAR-10 dataset, because of the lack of color and structural variation, the 'plausible colors' are more likely than not also the 'regressed' colors.

Table 1: Baseline threshold performance on CIFAR-10 testing set. Note that the grayscale accuracy is high.

Baseline Threshold Accuracy on CIFAR-10 Testing Set	
CIFAR-10 Ground Truth Images	CIFAR-10 Grayscale Images
100%	91.05%

Feature Preservation An interesting and quantifiable metric for the success of our colorization models is the evaluation of our model predictions on a pretrained classification model. The evaluation accuracy on the model predictions provides valuable information regarding feature preservation along the color space. Namely, if the outputted model predictions contain similar semantic information as ground truth, then it should perform similarly well on the classification model that was trained on ground truth.

First, we trained a simple classifier on ground truth images in the RGB space. The baseline accuracy and grayscale performance on this model can be found in Table 2 (note that the grayscale performance is high as much of the classification relies on spatial information). Then, for each model, we converted its outputs to RGB and passed it as input to this pretrained model. We then report the classification accuracy.

This is a more robust evaluation of our models as it does not directly rely on pixel similarity. Instead, it relies on the preservation of feature encodings. As we will see, even though regression on average predicts pixel values closer to ground truth, the classification models encode more important semantic information within the color channels. Thus, even though there might be an expectation that outputs with smaller euclidean distance from ground truth would contain the same encodings, this perception is false. The classification approach exceeds the regression approach in preserving the original integrity of the image, which may also explain why the results are more visually convincing.

Table 2: Baseline feature preservation performance on CIFAR-10 testing set. Note that the grayscale accuracy is high.

Baseline Feature Preservation Metric on CIFAR-10 Testing Set	
CIFAR-10 Ground Truth Images	CIFAR-10 Grayscale Images
81.44%	73.40%

4.2 Regression

Color Spaces As previously mentioned in this paper, we regressed on the RGB, Lab, and Yuv color spaces. In order to gauge the performance of these color spaces in the context of our model architectures, we trained the Baseline Regression Model as seen in Figure 4 as well as the Convolution Only Regression Model as seen in Figure 5. The quantitative results for these trained models can be found in Table 3, and the qualitative results (example outputted images) can be found in Figure 8.

The RGB color space unsurprisingly performed the worst. A large majority of this poorer performance can be attributed to the model being required to predict 3 output channels as opposed to 2 (as with Lab and Yuv). Additionally, measuring distance within a color space is somewhat human-perception oriented (in the eye of the beholder, so to say). Both the Lab and Yuv color spaces were created with color *perception* in thought, i.e. the distances within these color spaces are more perceptible. So even qualitatively, the results are better for the Lab and Yuv color spaces as the model is optimizing a color distance that is more perceptibly plausible to the human eye.

We eventually decided that the Lab color space performs the best for image colorization with regression due to the performance combination between our evaluation metrics (see Tables 3 and 4). The Lab color space outperforms the Yuv color space in *every* quantifiable task, and the results for Lab on random samples across the entire dataset looked more visually plausible. Thus, for classification purposes, we decided to build our architecture around this color space in our subsequent experiments.

Additionally, by viewing the outputs in Figure 8, we can see that regression on the *Lab* color space makes the least mistakes and still accurately colorizes the images. For example, multiple frogs in the Yuv color space were incorrectly colored red.

Architectures We also attempted two main differing model architectures: one that contains purely convolutional layers, and one that included fully connected layers. We explored transfer learning by utilizing the first two convolutional layers from VGG16 trained on ImageNet as feature extractors [8] [3]. These features were evaluated as both initializers (i.e. learnable) and feature extractors (i.e. unlearnable). We only attempted transfer learning on the *Lab* color space, as we concluded that it outperformed models without transfer learning for the colorization task.

In general, any architecture we attempted to use that needed to upscale the learned features performed much worse in the colorization task, and the outputs were blurry and noisy (see "Baseline Regression" in Figure 8).

We found that purely using convolutional layers without dimensionality reduction gives the best results, both quantitatively and qualitatively. Additionally, for transfer learning, using unlearnable VGG layers (i.e. VGG as a feature extractor) performed worse than using learnable VGG layers (i.e. VGG as a weight initializer). This is likely due to the fact that VGG was trained for a classification task and we are using it for regression, so some fine tuning is indeed required.

Table 3: Performance of all trained regression models across different color spaces and architectures on threshold accuracy. Note: diagrams of the model architectures can be found in Figures 4 and 5.

Regression Threshold Accuracy Performance Across Color Spaces for Multiple Model Architectures			
Models	**RGB**	**Yuv**	**Lab**
Baseline Regression Model	91.22%	94.08%	95.58%
Convolution Only Regression Model	92.88%	95.68%	95.82%
Convolution Model w/Unlearnable VGG Features	N/A	N/A	95.71%
Convolution Model w/Learnable VGG Features	N/A	N/A	**95.85%**

4.3 Classification

Leveraging Regression Findings for Classification From the regression experiments, we learned that *Lab* is the preferred color space for image colorization (this will be further explored within the classification experiments) and that

Fig. 8: Example outputs from regression across different color spaces and multiple model architectures. Note: diagrams of the model architectures can be found in Figures 4 and 5. From this figure, one can see that the model trained with learnable VGG16 in the *Lab* color space features performs the best.

Table 4: Performance of all trained regression models across different color spaces and architectures on feature preservation (accuracy of the regression outputs on a pretrained classification model). Note: diagrams of the model architectures can be found in Figures 4 and 5.

Regression Feature Preservation Performance Across Color Spaces for Multiple Model Architectures			
Models	**RGB**	**Yuv**	**Lab**
Baseline Regression Model	66.68%	40.76%	74.66%
Convolution Only Regression Model	74.41%	74.65%	76.46%
Convolution Model w/Unlearnable VGG Features	N/A	N/A	75.62%
Convolution Model w/Learnable VGG Features	N/A	N/A	**76.65%**

utilizing VGG16 trained on ImageNet [3] [8] is beneficial for the model's performance due to the feature encodings. In our classification experiments, we build upon these findings to obtain quality colorizations for CIFAR-10 images [6].

Note: we attempted to quantize the *uv* color space, however, it produced results that were mostly gray and worse than quantizing the *ab* color space. An example output quantized on the most optimal bin size (81) can be found in Figure 9.

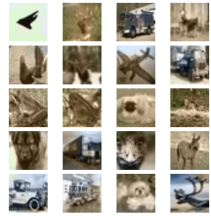

Fig. 9: Results when quantizing *uv* subspace into 81 bins without rebalancing.

Bin Sizes At first, we attempted to train the model with the same number of bins in the quantized *ab* space as in "Colorful Image Colorization" (313 bins) [10]. However, we quickly found that the model overfitted even on small networks with just 3-4 layers: the model did not learn anything meaningful and the results were almost grayscale. This behavior could stem from the following reasons: the CIFAR-10 dataset has less intra-class variation than ImageNet (which [10] was trained on) and CIFAR-10 has very few examples when compared to ImageNet. However, we also found that quantizing the *ab* space into 64 and 100 bins behaved in the same manner (i.e. overfitting). Quantizing into 49 bins and 81 bins performed the best, and specifically 81 bins was the superior quantization. We

suspect that using 49 and 81 bins divides the *ab* space in the CIFAR-10 dataset more meaningfully than 64 and 100 bins.

Quantizing the *ab* space into 81 bins performs better on every comparable quantifiable task (See Tables 5 and 6). Additionally, although qualitatively, the 49 bin quantization looks similar to the 81 bin quantization in the examples provided, 81 bins outperforms on the entire CIFAR-10 dataset by providing more vibrant, more plausible/accurate colors (see Figure 10).

Rebalancing We also decided to explore class rebalancing on the color space. As explained in section 3.4, placing the rebalancing term $v(Z_{h,w})$ in the cross-entropy loss function penalizes wrong predictions on saturated colors more than desaturated colors, encouraging more vibrant and visually plausible colors in the output. However, we did not notice any stark differences both qualitatively or quantitatively. The rebalanced images were only slightly more saturated than the non-rebalanced images. We believe that due to the lack of intra-class color variation in CIFAR-10 (for example, cars are usually black, blue, or grey), the rebalancing method could not promote rare colors. This is because if there is only a small selection of colors to choose from, then there indeed are no 'rare' colors that rebalancing attempts to promote (as shown in Figure 2, the colors are densely concentrated within the subspace).

However, we did notice some differences, especially in the predicted colors for cars: more of the cars were colored red when rebalanced, and rebalancing produces more vibrant blue and green colorings. This suggests that rebalancing is working as intended, but the CIFAR-10 dataset is restricted to a color distribution that has less variance and an even distribution of colors per class.

Transfer Learning Following the experiments with regression models, we decided to use VGG16 layers pretrained on ImageNet used as either a feature extractor (layers were unlearnable) or a weight initializer (layers were learnable). The architecture followed the regression methods, where we extracted the first two layers of VGG16 in order to avoid dimensional reduction in the feature space.

The quantitative results for the VGG architectures all outperform models trained explicitly and completely from scratch in almost every metric. However, any model that used VGG performed very similarly in the quantitative evaluation meaures, and the determining factor for the best model was ultimately subjective.

When performing the subjective evaluation, it was clear to see that VGG as a feature extractor outperformed it as a weight initializer (while the opposite relationship held true for regression). The model converged to a lower loss, and this was observable as the model made less 'mistakes' in the test dataset. This is likely due to the nature of the loss function. In the regression experiments, the loss was drastically different (regression) than what VGG16 was originally trained on (classification), so we needed to allow the VGG features to learn. However, in this case, the models are both attempting a classification problem

in some regard. Forcing the features to maintain the semantic information lead to more saturated results, as seen in Figure 10.

Table 5: Performance of all trained classification models across different bin sizes and architectures on threshold accuracy. Note: diagrams of the model architectures can be found in Figure 6.

Classification Threshold Accuracy Performance Across Bin Sizes for Multiple Model Architectures	
Models	**Threshold Accuracy**
49 Bins No Rebalancing	94.60%
64 Bins No Rebalancing	94.39%
100 Bins No Rebalancing	94.26%
81 Bins No Rebalancing	95.58%
81 Bins w/ Rebalancing	95.56%
81 Bins w/Unlearnable VGG Features No Rebalancing	95.40%
81 Bins w/Unlearnable VGG Features w/Rebalancing	95.40%
81 Bins w/Learnable VGG Features No Rebalancing	95.25%
81 Bins w/Learnable VGG Features w/Rebalancing	**95.57**%

Table 6: Performance of all trained classification models across different bin sizes and architectures on feature preservation (accuracy of the regression outputs on a pretrained classification model). Note: diagrams of the model architectures can be found in Figure 6.

Classification Feature Preservation Performance Across Bin Sizes for Multiple Model Architectures	
Models	**Classification Accuracy**
49 Bins No Rebalancing	74.79%
64 Bins No Rebalancing	73.57%
100 Bins No Rebalancing	74.92%
81 Bins No Rebalancing	76.74%
81 Bins w/ Rebalancing	75.97%
81 Bins w/Unlearnable VGG Features No Rebalancing	**76.94**%
81 Bins w/Unlearnable VGG Features w/Rebalancing	76.77%
81 Bins w/Learnable VGG Features No Rebalancing	76.45%
81 Bins w/Learnable VGG Features w/Rebalancing	76.23%

Fig. 10: Example outputs from classification across different bin quantizations and multiple model architectures. Note: diagrams of the model architectures can be found in Figure 6. From this figure, one can see that the model trained with unlearnable VGG16 features with rebalancing performs the best.

*Model was trained with 81 bins.

4.4 Best Model

After conducting our experimentation, we concluded that the class rebalanced classification model trained with unlearnable VGG16 features with an 81 bin quantized *ab* color space gave the most convincing plausible image colorizations. This model performs better than the best regression model (model trained with learnable VGG16 features) in feature preservation, but not in the threshold accuracy (this is as expected, consistent with the explanations given in the previous sections). However, simply looking at the outputted images, we have accomplished more saturated, less noisy images with the classification model. See Table 7 and Figure 11 for comparative results.

Table 7: Comparing threshold accuracy (TA) and feature preservation (FP) from the best regression model and best classification model.

Best Regression vs. Best Classification Model		
	Regression: VGG Learnable	Classification: VGG Unlearnable w/Rebal
TA	**95.85%**	95.57%
FP	76.65%	**76.77%**

Successes Our system often accurately provides plausible image colorizations on many different classes of CIFAR-10, particularly classes containing birds, horses, and natural scenes. The system is particularly robust in detecting skylines, grasses, and water due to the high frequency of natural backgrounds in the CIFAR-10 dataset. While the colorizations are somewhat noisy, they are nicely saturated and similar to the ground truth after the class-rebalancing procedure. In summary, we successfully created a system that can colorize images with generally high plausibility when trained on a small dataset. Please find ground truth images next to our model's successful predictions in Figure 12.

Failures We noticed that our best model struggled when the input image contains a white background. Our model tends to incorrectly color these backgrounds with a mint-green tone. This issue is particularly apparent after class rebalancing, which is explainable because white is the most frequent color in the CIFAR-10 dataset (low *ab* and *uv* values, as shown in Figure 3). Consequently, incorrectly classifying white isn't penalized harshly by our loss function, which allowed for more errors. We also observed a phenomenon present in the network of [10] in our CNNs initialized with VGG layers, where the model tended to generate a red patch around the dog's nose and mouth area. We suspect that this occurs because the first few VGG layers look for lower-level features such as a tongue that is sticking out in dogs, which tricked the network into hallucinating a red patch on dogs that don't have their tongues sticking out. See Figure 13.

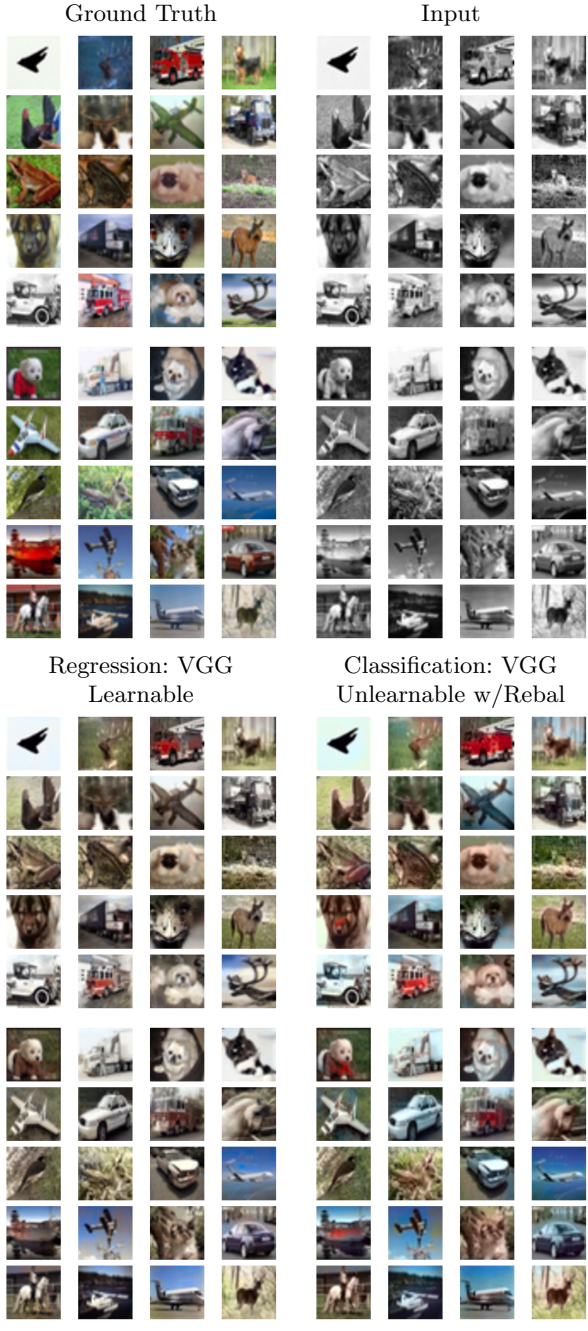

Fig. 11: Comparing outputs from the best regression model (left) and the best classification model (right).

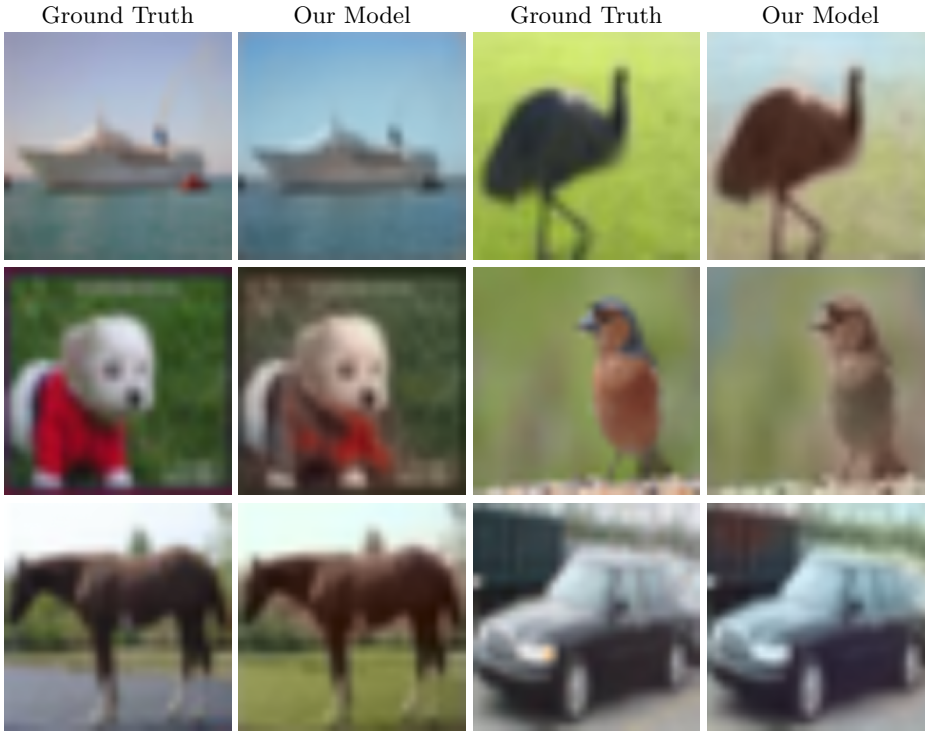

Fig. 12: Comparing ground truth images to successful predictions from our classification model.

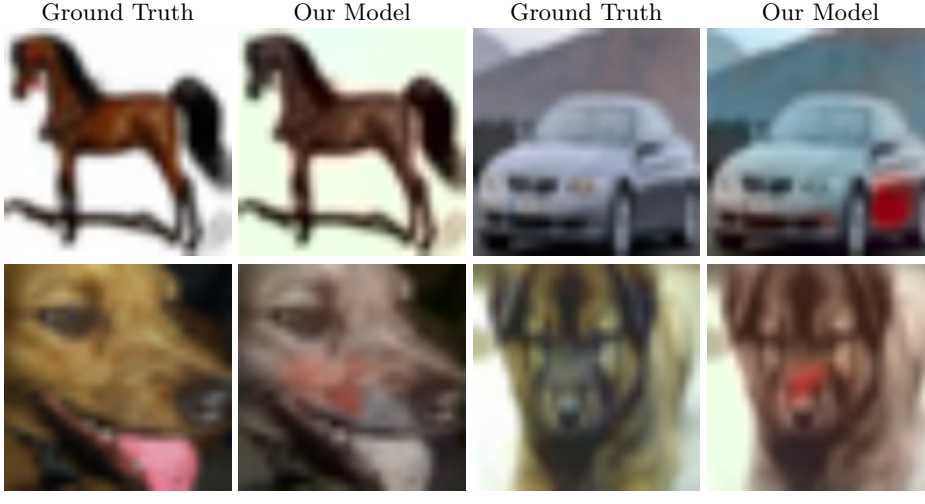

Fig. 13: Comparing ground truth images to unsuccessful predictions from our classification model.

5 Conclusion

Colorization of grayscale images is a complex task in computer vision that often requires great computational resources and large data size. However, we were able to produce decently plausible results and reproduce the expected trends observed in previous approaches. Our work has shown that through devised methods of classification, class rebalancing, and transfer learning, a relatively strong model can be built and tuned to achieve this difficult task even with a small dataset and limited computational resources.

One possible topic that requires future research is the method used to quantize different color spaces. We did not observe a clear correlation between the number of bins used in quantization and the quality of our outputs. In fact, a small difference in bin size resulted in a huge gap in performance of our colorization systems. Future research on colorspace quantization may yield promising improvements to current state-of-art methods used for image colorization.

6 Acknowledgements

We thank the course staff of COS429 for a great semester. Special thanks to Jia Deng and Angelina Wang for providing guidance on our final project.

7 Appendix

Please find the implementation of this paper on:
https://github.com/arin-champati/429-image-colorization

We pledge our honour that this paper represents our own work in accordance with University regulations.

/s/ Arin Champati
/s/ Brendan Houle
/s/ Byron Zhang

References

1. Charpiat, G., Hofmann, M., Schölkopf, B.: Automatic image colorization via multimodal predictions. Computer Vision–ECCV pp. 126–139 (2008)
2. Cheng, Z., Yang, Q., Sheng, B.: Deep colorization. CoRR **abs/1605.00075** (2016), http://arxiv.org/abs/1605.00075
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
4. Iizuka, S., Simo-Serra, E., Ishikawa, H.: 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 (Proc. of SIGGRAPH 2016) **35**(4), 110:1–110:11 (2016)
5. Irony, R., Cohen-Or, D., Lischinski, D.: Colorization by Example. In: Proceedings of Eurographics Symposium on Rendering 2005 (EGSR'05, June 29–July 1, 2005, Konstanz, Germany). pp. 201–210 (2005). https://doi.org/http://dx.doi.org/10.2312/EGWR/EGSR05/201-210
6. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (canadian institute for advanced research) http://www.cs.toronto.edu/ kriz/cifar.html
7. Larsson, G., Maire, M., Shakhnarovich, G.: Learning representations for automatic colorization. CoRR **abs/1603.06668** (2016), http://arxiv.org/abs/1603.06668
8. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014), http://arxiv.org/abs/1409.1556
9. Welsh, T., Ashikhmin, M., Mueller, K.: Transferring color to greyscale images. ACM Trans. Graph. **21**, 277–280 (07 2002). https://doi.org/10.1145/566570.566576
10. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. CoRR **abs/1603.08511** (2016), http://arxiv.org/abs/1603.08511