Colorizing grayscale images

Nolan Bock, Sai Nikhil Thirandas, Naveen Verma

bock.n@northeastern.edu, thirandas.s@northeastern.edu, verma.na@northeastern.edu

Objectives and significance

The goal of the project was to implement a series of Neural Networks that probabilistically colorize grayscale images and compare their performance to each other. We first attempted to establish a baseline with the Convolutional Neural Network model outlined in Zhang et al [1]. From there, we changed the layers in that Convolutional Neural Network to analyze how each component contributes to accurate colorization. Slightly altering the Zhang implementation also allowed us to reduce the problem scope to fit within the bounds of the semester. We then began to compare our model with a Fusion-Encoder-Decoder model [2] and analyzed its performance vs. all other Convolutional Neural Networks that we implemented. Additionally, we experimented heavily with methods of evaluation since evaluating the probabilistic coloring of an image is a challenging problem.

Our motivation for the project was to explore Convolutional Neural Networks and their ability to place colors in certain regions. By the end of the project, we began to understand the problem space (colorization) and the general architecture of a CNN model. We also gained experience in analyzing and plotting the model's accuracy with respect to the component/layer of a CNN model and also with respect to certain hyperparameters and various loss functions.

Background

Summary

Our project deals with Convolutional Neural Networks that map grayscale input to a distribution over quantized color value outputs [1]. The function of our trained CNN is to output 2 channels of colors 'a' and 'b', that can be concatenated with 'L' channels to get the colored image, but not necessarily the correct colorization. These colors are the ones nearest to the actual image when the model is well trained. For example, apples are likely red or green, but distinguishing between one versus the other in a specific image will be a probability question - which color is more probable for the apple given the other features in the image. A very improbable color for an apple would be black. Given that, our trained CNN will have inherent inaccuracy from the baseline truth of some image, but the goal of our project will be to output a probable image. Zhang et al proposed an evaluation strategy by using Amazon Mechanical Turk to pay

individuals to pick the “real” image out of one colorized and one with true color, but that is out of the scope of this project.

We also implemented a Fusion-Encoder-Decoder model which uses a deep CNN fused with high level features extracted from the pretrained Inception-ResNet-v2 model [2]. The model for our Fusion-Encoder-Decoder is discussed in detail in the methodology section. Broadly, it will supplement the existing CNN to do the same colorization that we expect from the CNN, but with additional feature extraction and layers. Additionally, our Fusion Encoder Decoder model formulates the problem in a slightly different manner than the formulation in Zhang et al. This was of particular interest to our team because it allowed us to explore the efficacy of colorization when colorizing was formulated using several different types of loss function and as both a classification and regression problem. Further details are discussed in the next subsection and the methodology section.

Previous work

Our approach differed slightly from our proposed approach, which is highlighted below. The next section (methods) will highlight the differences between our proposed approach and the approach that we used. Our initial goal was to build upon the CNN outlined in Zhang et al. The specific architecture of Zhang et al’s implementation is shown below and further explored throughout this section.

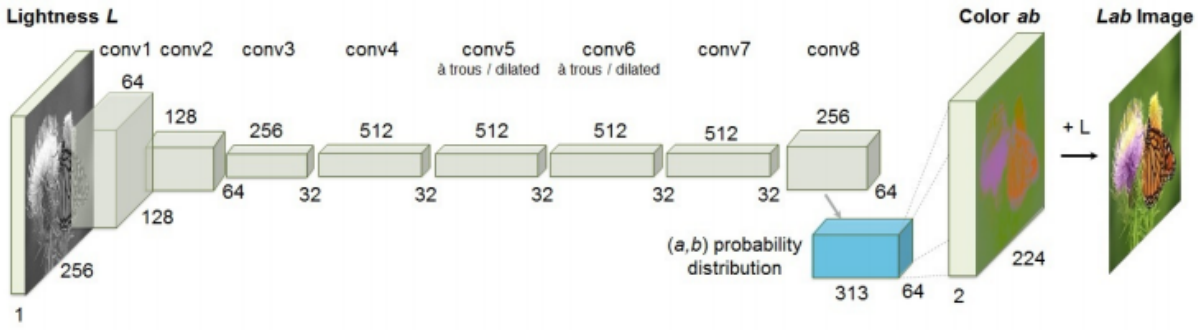

Fig. 2. The network architecture used by Zhang et al. Each convolutional layer refers to a block of 2 or 3 repeated convolutions and ReLU layers followed by a BatchNorm layer. There are no pool layers [7].

Broadly, the CNN architecture maps grayscale input to color as shown in the image above. Zhang et al uses two channels: the L channel, which has light information but no color information, and the ab channel, which has color information. The *a* channel shows you the values on the red to green axis, while the *b* channel shows the values on the blue to yellow axis. Perhaps a reasonable assumption would be to define a loss function that measures loss by the square difference between the true and estimated *ab* pixel

values for each pixel in an image, but this loss function will find no difference between the following two images: one which misclassified color in the foreground and one that misclassified color in the background on an unimportant object.

Instead, Zhang et al formulated the problem as a problem of multinomial classification. Given that, the *ab* space was translated into a discrete space. Figure 3 shows the translation from continuous colors in *ab* space (a regression problem) and discrete colors in *ab* space (multinomial classification).

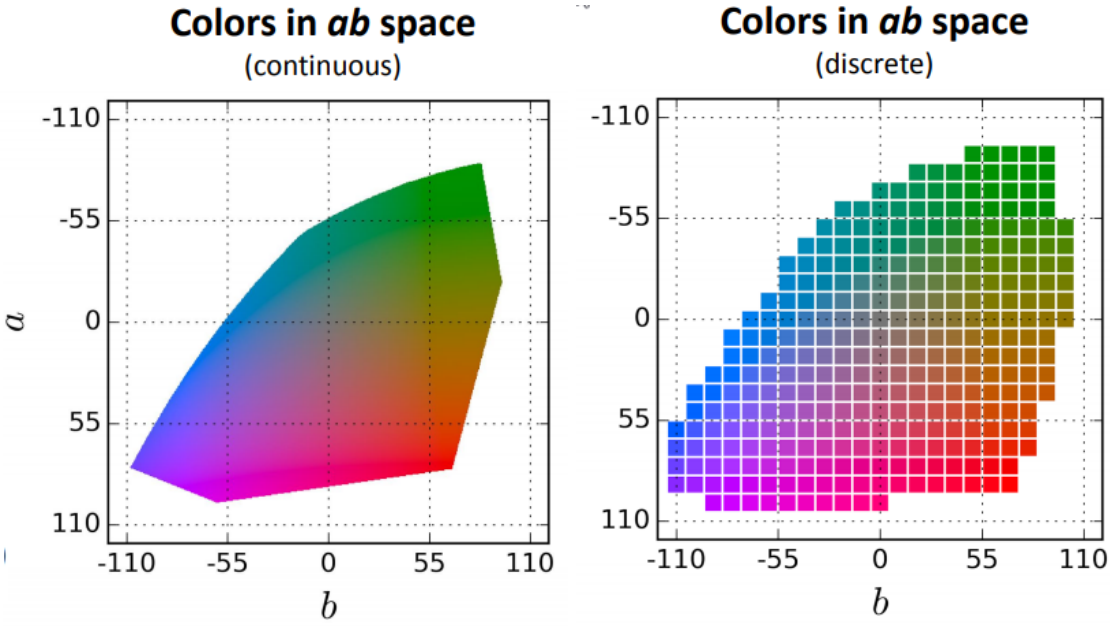

Fig. 3. Quantized *ab* color spaces. The discrete space was used for the CNN trained in Zhang et al [7].

The colors in ab space were broken into bins of size 10, which means there are a total of 313 ab pairs. An example classification is shown in Figure 4, which highlights that the foreground object (the bird) is blue, red, or purple and the background object is green, yellow, or brown. The heavier the coloring, the more likely that color is the true color according to our model.

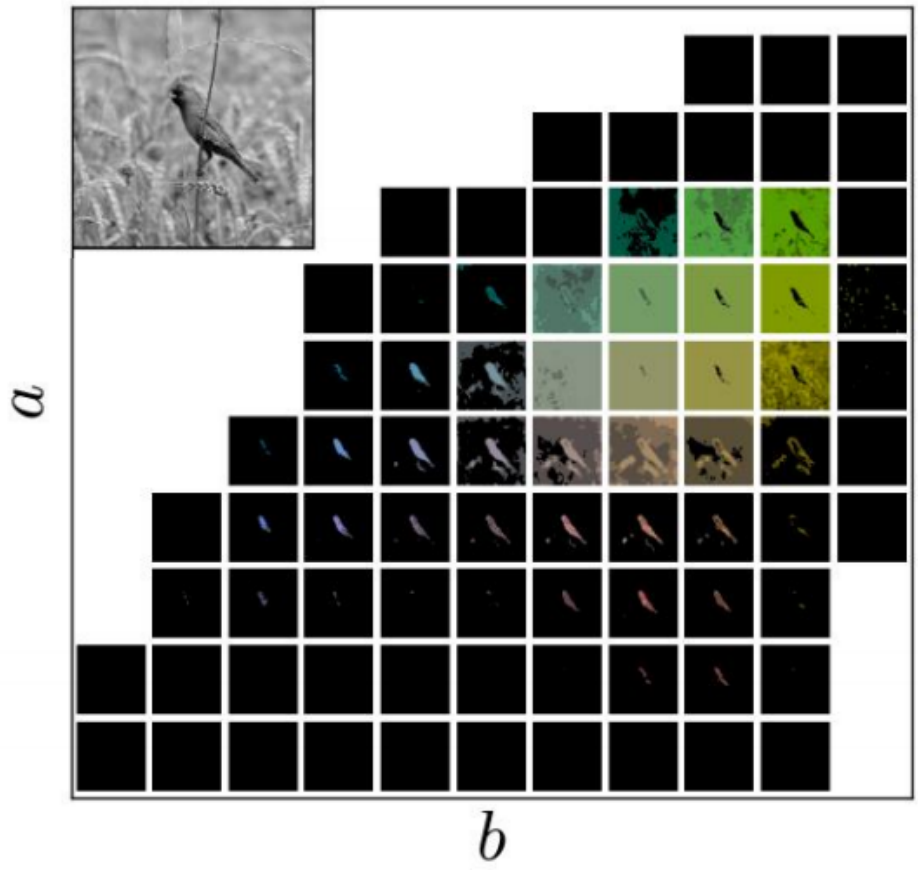

Fig. 4. Bird classification with foreground and background differential classification [7].

Here we can see that the bird image is classified by the discrete *ab* color space. Zhang et al also considered that the vast majority of pixels in their training set were focused around (0,0) of the *ab* space (dark red), which created images that were desaturated. Thus, a bias towards desaturated images was created because the CNN predicted unnaturally bland images. To account for this, Zhang et al added a class rebalancing term to resample the rarest colors in a given dataset. Thus, their loss function becomes a cross entropy loss function with class rebalancing, which will have the final form as shown in Figure 5.

$$\mathrm{L}(\widehat{\mathbf{Z}}, \mathbf{Z}) = -\frac{1}{HW} \sum_{h,w} v(\mathbf{Z}_{h,w}) \sum_{q} \mathbf{Z}_{h,w,q} \log(\widehat{\mathbf{Z}}_{h,w,q})$$

Fig. 5. Class rebalancing in the loss function to encourage learning rare colors. The variables h,w are the image dimensions and q is the number of quantized *ab* values (313 in our case) [7].

The loss function uses several terms that we need to define further. $v(\dot{\bullet})$ is a weighting term that can be used to rebalance the loss based on color-class rarity. The $\hat{Z}$ term is the predicted probability distribution for some image and Z is the true probability distribution. The H and W terms are the dimensions of the image. Further, the equation for the weighting term is defined in figure 5.1, below.

$$v(\mathbf{Z}_{h,w}) = \mathbf{w}_{q^*}, \text{ where } q^* = \arg\max_q \mathbf{Z}_{h,w,q}$$

$$\mathbf{w} \propto \left((1-\lambda)\widetilde{\mathbf{p}} + \frac{\lambda}{Q}\right)^{-1}, \quad \mathbb{E}[\mathbf{w}] = \sum_q \widetilde{\mathbf{p}}_q \mathbf{w}_q = 1$$

Fig. 5.1. Weighting term definition and expected value [1].

$$\mathbf{w} = k\left((1-\lambda)\widetilde{\mathbf{p}} + \frac{\lambda}{Q}\right)^{-1}$$

Fig. 5.2. Weighting term with a proportionality constant.

Again, there are several terms in figure 5.1 and 5.2 that we must define, which we will do below:

- *h* represents a vertical location and varies from 0 to H (height of image).
- *w* represents a horizontal location and varies from 0 to W (width of image).
- *q* represents class in the *ab* color gamut and varies from 1 to 313.
- Q is total classes, which is 313 in our case.
- q* is the class value between 1 to 313 in which the pixel has highest posterior (**z**).
- k represents the proportionality constant term that creates an equality between w and the rest of the right side.
- **w** is the rarity_weights for the 313 classes.
- **p** is the class priors over the entire dataset for the 313 classes.

These terms and the values they took in our implementation are further discussed in the Methodology section, but are necessary to highlight here as they form the basis of the Zhang et al. implementation [1].

Finally, the Zhang et al. implementation uses an annealed mean to convert the output of the model, a probability distribution, to the *ab* plane.

$$\mathcal{H}(\mathbf{Z}_{h,w}) = \mathbb{E}\big[f_T(\mathbf{Z}_{h,w})\big], \qquad f_T(\mathbf{z}) = \frac{\exp(\log(\mathbf{z})/T)}{\sum_q \exp(\log(\mathbf{z}_q)/T)}$$

Fig. 5.3. Annealed mean definition - converts probability distribution to the *ab* plane [1].

In the equation shown in figure 5.3, **z** is defined above figure 5.1 and has the same representation, and T represents the temperature, a user variable that can be varied during implementation. Zhang et al defines T to be 0.38 in their model [1].

The final layer of Zhang et al's CNN is to convert from the 64x64x313 probability distribution to a 224x224x2 layer that represents the *ab* channel of the image. As a result of the loss function favoring rare colors, the images produced will be more colorful. Our initial implementation goal was to implement the simple SSE (sum of squared errors) loss function as well as both cross entropy loss function and a weighted cross entropy loss as described above.

The final goal of our project was to implement a Fusion Encoder Decoder model, which has an architecture that is summarized below both in the figure and paragraph below.

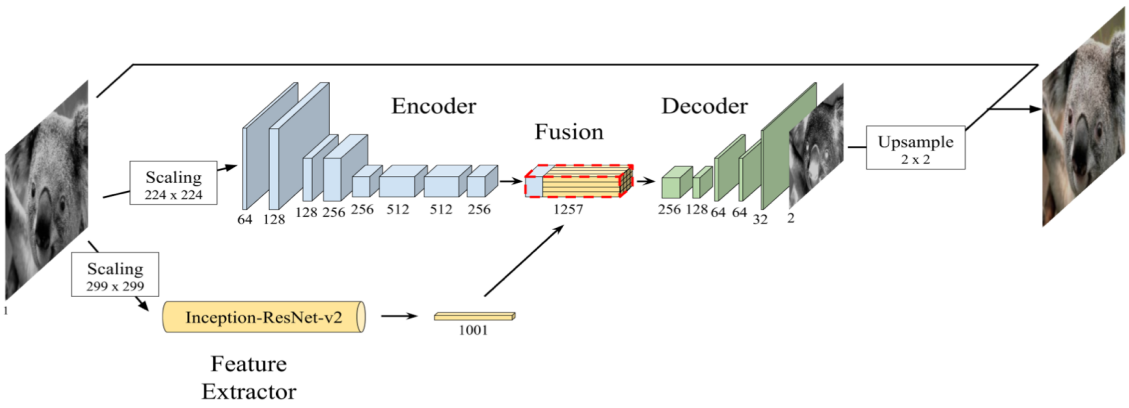

Fig. 6. Fusion Encoder Decoder model.

Instead of training a feature extraction branch from scratch, the Fusion Encoder Decoder model makes use of an Inception-ResNet-v2 network and retrieves an embedding of the gray-scale image from the last

layer of the decoder. As summarized by Baldassarree et al, “the network is logically divided into four main components. The encoding and the feature extraction components obtain mid and high-level features, respectively, which are then merged in the fusion layer. Finally, the decoder uses these features to estimate the output” [2]. The encoder uses 8 convolutional layers to output a 512 depth feature initial representation. Feature extraction scales the image and stacks it upon itself separated by the three channels (l, a, and b from the l and *ab* channels). Fusion performs several additional convolutional layers that eventually perform 256 convolutional kernels of size 1 x 1 that are used to generate a H/8 x W/8 x 256 volume. Finally, the decoder takes the H/8 x W/8 x 256 dimension input, upsamples using the nearest neighbor approach, and outputs height and width at twice the height of the input [2]. A full architectural overview and the associated code is provided in Baldassarre et al, which we will reference throughout our implementation. One important distinction between the Fusion Encoder Decoder and the Zhang implementation is that the Fusion Encoder Decoder does not formulate the problem as a classification problem and instead formulates it as the more straightforward regression problem. That is, the Fusion Encoder Decoder maps an input image in grayscale to a color image; whereas, the Zhang implementation maps a grayscale image to the *ab* space probability distribution described above. Thus, the evaluation methods for both models vary slightly. Similarly, the Fusion Encoder Decoder loss function differs from the loss function implemented in the Zhang et al implementation and is defined in figure 6.1, below.

$$C(\mathbf{X}, \boldsymbol{\theta}) = \frac{1}{2HW} \sum_{k \in \{a,b\}} \sum_{i=1}^{H} \sum_{j=1}^{W} (X_{k_{i,j}} - \tilde{X}_{k_{i,j}})^2;$$

Fig. 6.1. The Fusion Encoder Decoder loss function as implemented in Deep Koalarization [2].

In figure 6.1, the equation C is the cost for some image and the image is represented by the variable **X**. The cost is defined as the sum of squared errors for each pixel in each channel between the true and predicted 128x128 images. Similarly, H and W are the height and width of the image (i.e. 128 and 128). Finally, a and b are the *ab* channels and X_{kij} is pixel value for channel k at location (i, j) in the true image. Similarly, X-hat is the estimated pixel value. Our initial goal was to implement this exact loss function for our Fusion Encoder Decoder, as well.

Methods

Data

Initially, our goal was to implement the Zhang and Fusion Encoder Decoder models exactly as they were implemented in their respective papers (described above in the Previous Work section). We began by implementing the Zhang model on Google Colab and using the GPU runtime to train in 12 hour increments. This required saving the weights and restarting the training loop each time the GPU shut down, which happens automatically after 12 hours of training. Unfortunately, these implementations led to colorizations that were very far from the ground truth.

Fig. 7. Ground truth cat vs. colorized cat on initial model

As figure 7 illustrates, our initial implementations of the Zhang model showed drastic miscolorizations. In this case, the cat is not only missing the vibrance in its color, but also has a blue stripe across a large portion of its body. After experimentation, we attributed this error to our dataset, which we will now explain. The initial goal of our project was to train on a very large dataset (the full Places 365 dataset), which has 365 classes of 256x256 images with 4000 to 5000 images in each class - a total of about 1.8 million images [4]. Some classes within that dataset varied significantly in their colorizations. For example, figure 8 illustrates that the Florist Shops class has drastically different colorizations depending on the type of flower included in the image or if the image had a flower in it at all.

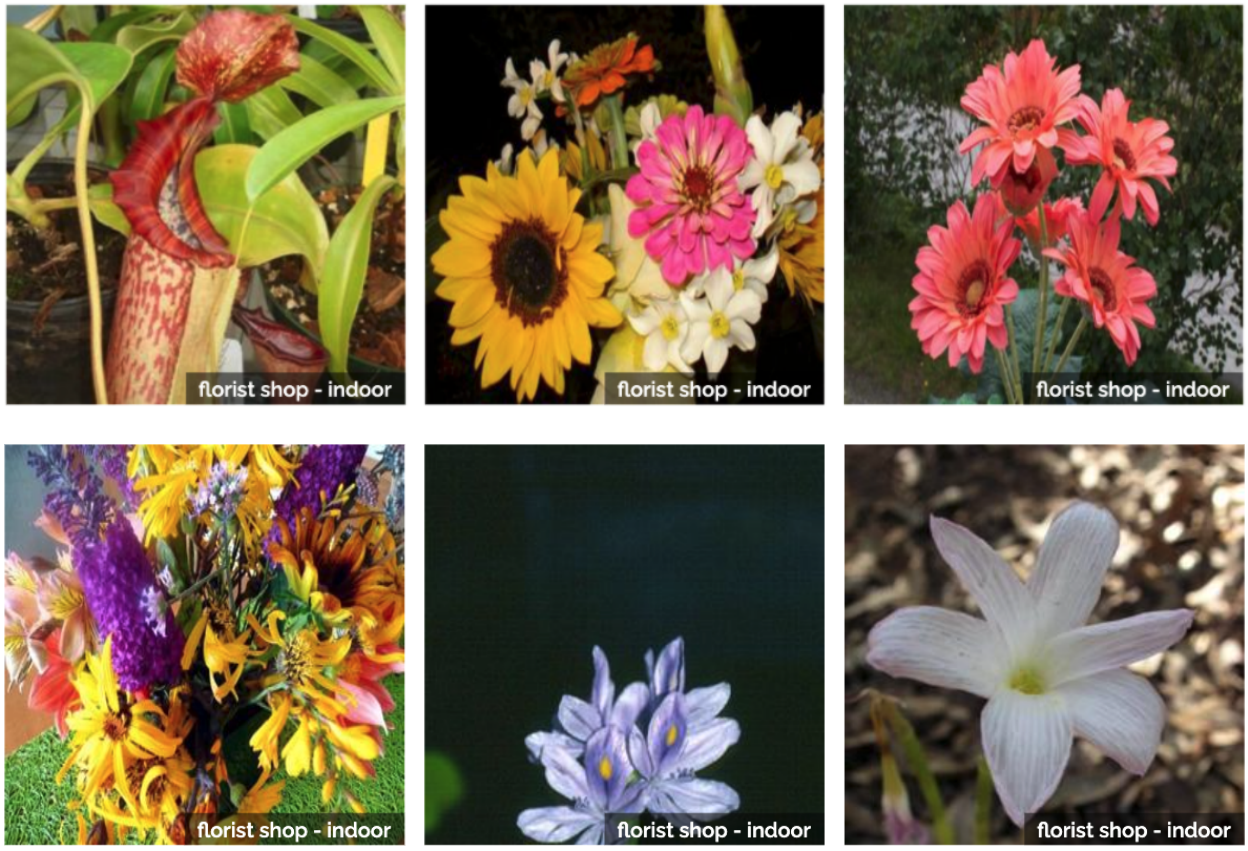

Fig. 8. Flower images from the Places 365 dataset.

We realized that our neural network was not only being trained to colorize an image probabilistically, but also the model would first need to implicitly classify the image (e.g. as a cat or a flower) in order to assign probable colorizations to different regions of the image. We believe this method was successful for the Zhang implementation because of its access to millions of images and Zhang et al's ability to train on very long increments - training loops that run several days. After discussion with Professor Radivojac, we decided to reduce the scope of the colorization problem in order to more quickly iterate on models. Specifically, we decided to train our models on specific classes within the Places 365 dataset and test our models ability to colorize images from the same class that were not included in the train set. The reduction meant that the task of implicitly classifying the image while colorizing it would be removed and resources, both time and computation, used would be greatly reduced. We chose to use the classes Butte, Farm, Pizzeria, and Forest Roads from the Places 365 dataset, as well as the Celeb class from the CelebA dataset [9] in order to test how well our models could generalize to other datasets besides Places365. Our full training, validation, and test datasets are available on google drive [10].

Our models also use data augmentation, which occurs during training of both the Zhang et al and the Fusion Encoder Decoder models. This process makes sure that the model sees different augmented versions of the same image in different epochs. That is, images may be shifted slightly in their frame from epoch to epoch. Specifically, the operations for augmentation are zooming, rotating, shearing and horizontal flipping. This also serves to prevent overfitting and increases variance of the model. It should be noted that data augmentation in our models is a runtime operation. Thus, we don't store the augmented images anywhere.

Methodology

As described above, our problem statement changed significantly from the alterations in the dataset. Specifically, the decision to remove the initial image classification from the problem meant that our data would be trained on some class (e.g Butte or Pizzeria) and that model would colorize images from only that class. Thus, some complexity of both the Zhang et al and Fusion Encoder Decoder model was unnecessary since our problem statement was reduced. The implication of this in our models was that we changed the convolutional layers as described below.

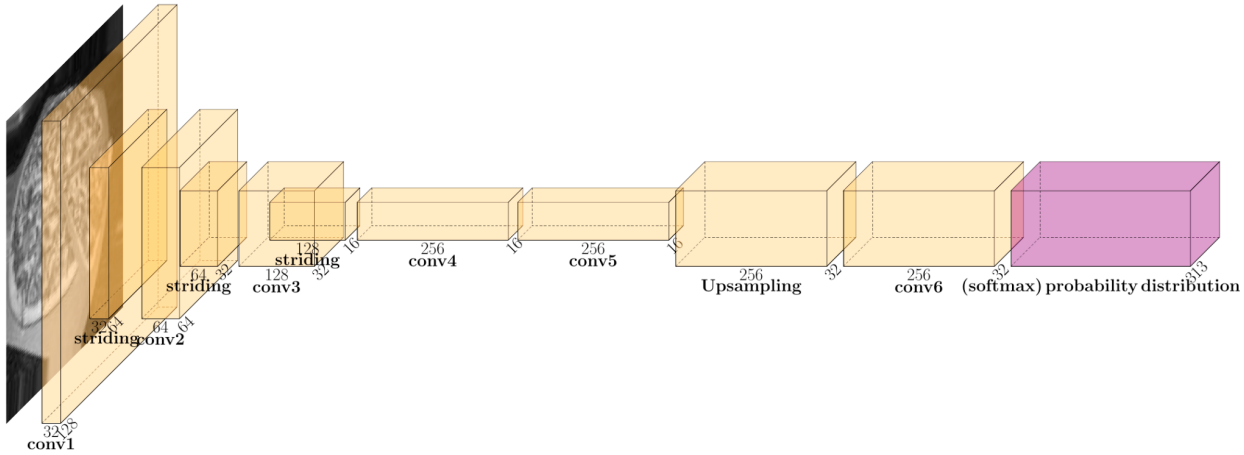

Fig. 9. Our model architecture (generated using PlotNeuraNet [14]), which is based on Zhang et al. 's implementation, excluding the probability distribution to image mapping. This is the model that is trained.

Figure 9 shows our model, which is essentially a scaled down version of Zhang et al's (shown in figure 2). Still, due to the reduction in our problem statement, we deviated from the Zhang model in several ways after experimenting with hyperparameters in order to balance the training time and efficacy of our colorings. Below we summarize our changes and the reasoning behind each change.

First, we changed the input image size to 128x128 instead of 256x256. We originally trained and tested our model on the Cifar-10 dataset, which contained 32x32 images, but 32x32 seemed to have too low resolution to identify colorization even though it might be accurate empirically. Therefore, we chose

to use 128x128 images in order to see the details in the image, but still have faster training loops due to smaller image size.

Further, we reduced the kernel layers to half depth. That is, a 64 input dimension (depth) in some convolutional layer in the Zhang model becomes a 32 input dimension convolutional layer in our model. Additionally, we completely removed two of the 512x32 convolution layers from the middle of our model - originally there were four in Zhang, we scaled down to two and further scaled the dimensions to 256x16.

Our rationale for reducing the dimensionality of the parameters is that Zhang et al's original model was trained on the whole imagenet dataset (Places 365) of 1.8 million images and we chose to reduce that number greatly in order to speed up training - this choice is discussed in detail in the data subsection and will be briefly summarized as it relates to our methodology. In general, we kept the layer ordering and ratio of dimensions the same as Zhang et al's model to simulate the same behavior, but on a smaller dataset. We believe this decreased our training time and prevented overfitting. Additionally, we chose to shrink the model because the imagenet database (Places 365) has 1.8 million images with 1000 classes and, in order to do multinomial classification of this magnitude, a complex convolutional neural network model is required. Further, Zhang et al had to colorize *after* their model appropriately classified an image, which may have required more complexity. Thus, as previously discussed, we moved to our reduced problem statement and reduced the model to implement, as well. That is, instead of trying to solve a classification problem to classify images and then colorize those images, we are trying to solve the colorization problem only. As discussed in the data subsection, we train each model on its own class and test on that same class, as well. For example, if we train a model on the pizza class, we only use that model to colorize the pizza test images.

Besides designing the neural network architecture, we also had to calculate the class priors and rarity weights in order to use the class rebalancing discussed in the previous work subsection. Essentially, calculating these priors is giving more weight in the loss function (figure 5) to the rare pixels. In order to do so, we must calculate all aspects of figures 5.1 and 5.2, which we show again below for clarity. All variables are defined in the previous work section, we omit their redefinition for the sake of brevity. We first count pixels in all images in our dataset and get a count of all 313 classes in the color gamut (figure 3). From this information, we calculate the class priors for all the classes in the entire dataset, which is the **p** term in the figure 5.1 - we use **p** throughout this formulation to represent the **p** tilde variable. We then create a uniform distribution over all classes with probability 1/Q where Q=313, the number of discretized *ab* classifications. Next, we take a weighted mixture of these two distributions with lambda as the weight of uniform distribution and 1-lambda as weight of **p** (class priors). Now the rarity of each class should be inversely proportional to its own probability, so it is also inversely proportional to the weighted mixture described above. Now, **w** in figure 5.1 is proportional to the inverse of the distribution, but we do not

know the constant term k in figure 5.2. We find this constant term by using the last equation in figure 5.1, which says that expectation of **w** over **p** must be 1.

$$v(\mathbf{Z}_{h,w}) = \mathbf{w}_{q^*}, \text{ where } q^* = \arg\max_q \mathbf{Z}_{h,w,q}$$

$$\mathbf{w} \propto \left((1-\lambda)\widetilde{\mathbf{p}} + \frac{\lambda}{Q}\right)^{-1}, \quad \mathbb{E}[\mathbf{w}] = \sum_q \widetilde{\mathbf{p}}_q \mathbf{w}_q = 1$$

Fig. 5.1. The same figure as in previous work, repeated here for the formulations in this section. Weighting term definition and expected value [1].

$$\mathbf{w} = k\left((1-\lambda)\,\widetilde{\mathbf{p}} + \frac{\lambda}{Q}\right)^{-1}$$

Fig. 5.2. The same figure as in previous work, repeated here for the formulations in this section. Weighting term with a proportionality constant.

Now, we have calculated the rarity_weights (**w**) for all 313 classes. Since our true class labels are soft encoded, each pixel has 5 non zero class scores between 0 to 1, which are the class posteriors for 5 nearest neighbours of the pixel in the 313 ab gamut of classes smoothed using a gaussian distribution. The actual rarity_weight used for the pixel is the rarity_weight of the class in which the pixel has highest posterior (nearest 1 *ab* class in the gamut).

Next, we must prepare the true class labels for use. As mentioned in Zhang et al, the Zhang based model is not end to end trainable. That is, the output of a Zhang based model is a probability distribution, which must be used to probabilistically create an image [1]. Thus, we learn the probability distribution, represented by the pink block in figure 9 above, during training. The probability distribution is the softmaxed class probabilities for the 313 *ab* classes. First, we downsample the original L channel image from 128x128 to 32x32 and then for each pixel we find the 5 nearest neighbours of it in the 2D gamut of the *a* and *b* channels. We then take the distances to these neighbours and use it to create a gaussian distribution, which tells us the 313 *ab* class scores for that pixel. We call this the soft encoding process. This process gives us the 32x32x313 class probabilities for all the pixels (i.e soft encoded class label not

one-hot vector). This is the **y** for the input image, which we hope our model can learn. Thus, we are trying to teach our model the probability distribution of each pixel in the downsampled 32x32 image.

During training we use these rarity_weights exactly as mentioned in figure 5 for our custom loss function. The advantage of using rarity_weights in the loss function is that our loss for rare pixels will be greater than the loss generated from non-rare pixels. So, in order to reduce the loss, our model learns the true classes of rare pixels with high priority.

Finally, there are a few remaining implementation details that are important to mention. Namely, the input to our model is a 128x128x1 L channel of some image that represents the grayscale of an image, which will be colorized. Accordingly, the output of our Zhang based model is a 32x32x313 probability distribution that is colorized by the process described below (in figure 10 and beyond). During training, our final implementation used a batch size of 100 and the custom loss function defined by figure 5. We also used Adam optimizator when training our model and a validation set that was 10% of the total class that was being trained (e.g. a class with 1000 images had 100 test images that were assigned randomly). Similarly, 80% of the full class set was used for training and the remaining 10% was used for the test set. That is, there was an 80/10/10 split between training, validation, and test sets respectively. We trained the Zhang based model for 40 epochs in order to avoid overfitting and saved the best models (as evaluated by our loss function) during the training. The models that performed the best against our loss function were the models used for testing. Similarly, we used early stopping when our epoch to epoch loss did not change by more than 0.001 on 7 consecutive epochs.

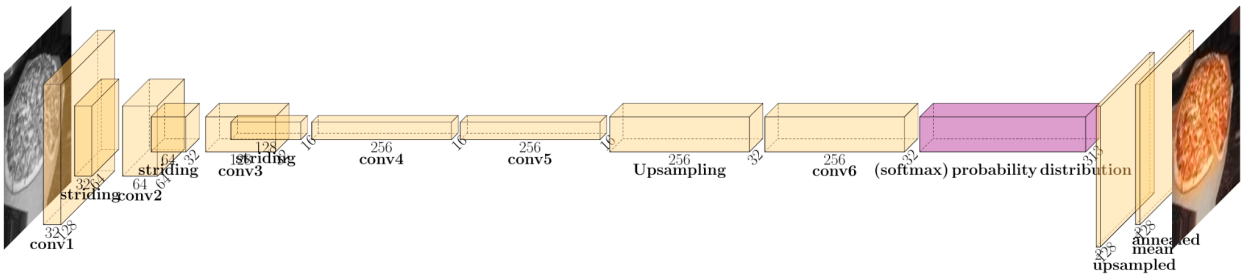

Fig. 10. Our model architecture (generated using PlotNeuraNet [14]), which is based on Zhang et al's implementation, including the probability distribution to image mapping, which is what produces results for testing.

Figure 10 shows our Zhang based models full scope from input image to output image. Thus, it is the same as figure 9 in the trainable portion of the model - everything up to and including the pink probability distribution. However, figure 10 shows the layers used to convert the model's output probability distributions to an actual image using the annealed-mean as calculated in figure 5.3. First we take an annealed mean of the probability distribution given by **f(z)** in figure 5.3. as defined in Zhang et al [1]. That is, we set the temperature, **T**, to 0.38. For each pixel, these 313 values are posterior of that class

given the pixel. That is, the 313 values represent p(z|pixel) where z ranges from 1 - 313. Thus, we take a product of the actual *a* channel values from our gamut for these classes with the class posteriors and then take a sum of the products throughout the class axis (i.e. the horizontal axis in the pink probability distribution shown in figure 10). After doing a similar operation on the *b* channel, we have two 32x32 matrices which are the *a* and *b* channels image converted from the 32x32x313 probability distribution output of the model. Next, we upsample the two 32x32 matrices 4 times to get the same size output as the input (128x128). Finally, we add the L channel to our 128x128 model to get the full 3 channel *Lab* image and save the rgb version of it for evaluation.

Next, we discuss our network architecture for the Fusion Encoder Decoder model based on the implementation in Deep Koalarization [2]. Figure 11, below, shows our architecture in full.

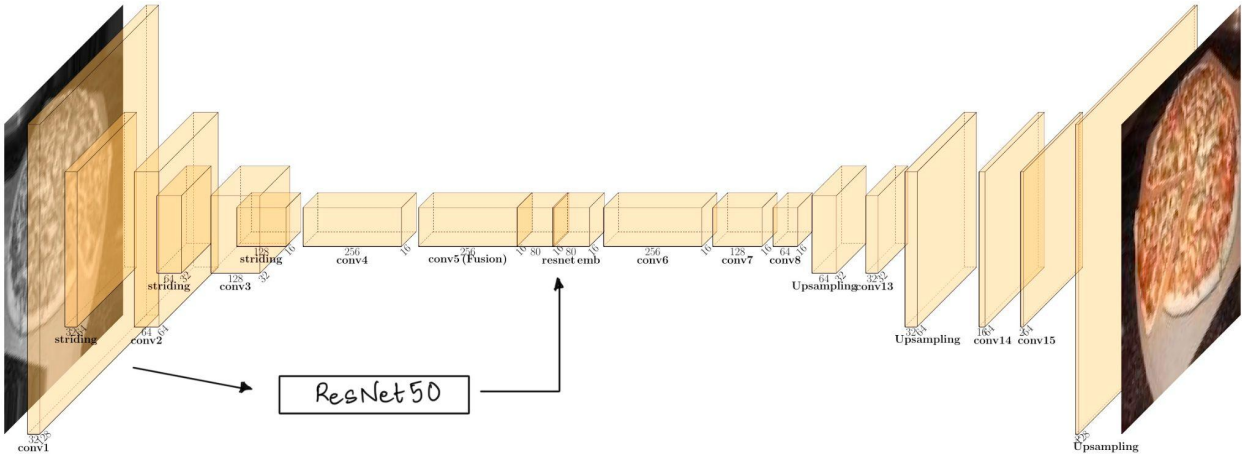

Fig. 11. Our model architecture (generated using PlotNeuraNet [14]), which is based on the Fusion Encoder decoder model.

The model shown in figure 11 is a fusion encoder decoder based upon the original architecture described in figure 6. Again, we made some changes due to the reduction in the problem size, which we will summarize below. We have the same reasoning as with the Zhang model for downscaling the size and depth of some of the layers, so we will omit that justification as we have already discussed it in depth. One distinction to note is that we used the ResNet50 model instead of InceptionResnetV2, which had 500+ layers. We chose to do this because the inception model is too large to be used at runtime during our training. Thus, we chose to run all of our images through the ResNet50 model and save the features in files to be used for training later in order to reduce training time.

Another change we implemented was to use the second to last layer of the ResNet50 model instead of the last layer. We did this in order to get more features from the ResNet50 model. The original implementation copies the embedding layer (the 1000x1 layer in figure 6) many times and stacks it with

one of the convolution layers. Instead of this process, we attempt to get a more meaningful shape (16x16x80) from the second to last layer and stack the embeddings onto the convolution layer. These differences are highlighted in the juxtaposition of figures 6 (the original implementation) and figure 11 (our implementation).

The major difference between our Zhang based model and our Fusion Encoder Decoder based model is that, with the Fusion Encoder Decoder, we are trying to minimise the euclidean distance between pixel intensities of *a* and *b* channels; whereas, in the Zhang based model we were classifying pixels among 313 *ab* classes. So, Zhang et al solves the colorization problem with multinomial classification whereas the Fusion Encoder Decoder uses regression [1]. Thus, we used the loss function exactly as it is defined in figure 6.1 in order to train our model.

There are several implementation specific details that we will highlight below. First, the model accepts two inputs, one is the L channel image sized at 128x128. Each pixel in the L channel input ranges from 0 to 100. The second input that the model expects is the output of the Resnet50 model, which took the same grayscale image as its input. The output of the Resnet50 model has dimensions 16x16x80. The true output of our model is the *a* and *b* channels normalized from [-128,128] to [-1,1]. The model's output is shaped as 128x128x2 in which the final 2-dimensional layer represents the two *a* and *b* channels using the tanh activation function, which is in the same range as the true values. During training, we used the mean squared error as our loss function, as discussed above. The batch size was 100. Again, we used the Adam optimizer during training and split between the training, validation, and test sets are 80%, 10%, and 10%, respectively. We again used 40 epochs and the best models were saved with best loss and best validation_loss so far. Similarly, early stopping was again used when validation loss stops improving by more than 0.001 for more than 7 epochs.

Evaluation strategy

The main hurdle to our evaluation was that the goal of our project is to colorize images probabilistically. That is, we hoped to create probable colorings of images, but not necessarily images that are a perfect recreation of their ground truth input. In practice, the most effective mode of evaluation for this model was human evaluation on images in a result set based on if they were passable as valid colorizations - it is relatively simple to evaluate if a batch of images is colorized appropriately or not. Obviously, this was not a suitable form for the final evaluation strategy, but it should be noted that throughout our experimentation, the most frequent method of evaluation was comparing results from ground truth images to some model’s predicted colorization. Still, we attempted to evaluate empirically, which is summarized below.

We began evaluating success by measuring the sum of the entire test set's misclassified pixel's magnitude: the amount one predicted pixel varies from the truth. However, this process immediately resulted in poor evaluation. For example, consider Figure 12, which shows an image predicted from the Zhang et al model (left) and the ground truth of that image (right). We may consider that image very close to the truth if a human was classifying success and failure since the main issue is the background and tie color (not very important) but our evaluator will classify this image as fairly incorrect since the entire background will be incorrect by a large margin. We considered clustering pixels and evaluating the clusters, but the same type of problem was likely to happen and that approach would not account for "insignificant" clusters (like the tie or background) vs. "significant" clusters.

Fig. 12. Inherent ambiguity in the model evaluation.

Ultimately, we decided to evaluate our performance using classification accuracy on the multiclass classification problem described in the Background section. That is, classification accuracy on our model will represent the accuracy of classification in the discretized *ab* space, which has 313 possible *ab* values. While we knew that classification accuracy wouldn't capture the holistic success of our model, it gives us an evaluation metric for comparing model to model performance while we make slight tweaks to the hyperparameters. Given the inherent error in evaluating probabilistic images, we expected our results to show low classification accuracy, but we do not necessarily expect that to predict a poorly colored image or a failure in the model, but rather a failure in evaluating success without human interaction. For example, the Zhang et al. implementation used Amazon Mechanical Turk to have humans rate which image some person thought was the ground truth when presented with ground truth and their model's output. For context on our model's performance, in the results section we will provide several images from our model and their ground truth to illustrate that relatively low classification accuracy does not necessarily correspond to poor colorings. As mentioned above (in the Background section), the Fusion

Encoder Decoder model formulates the problem not as a multinomial classification problem but as a regression problem. Thus, our evaluation technique for the Fusion Encoder Decoder cannot use classification accuracy on the probability distribution and instead will be trained to minimize mean squared error (MSE) and will be evaluated on the MSE and the coefficient of determination (R^2) in the test data.

Results

Evaluation of results

Below we summarize the results of our two models: our implementation based on Zhang et al and our implementation based on the Fusion Encoder Decoder. We will then highlight important and interesting aspects of each. Finally, the comparisons subsection will show a subset of images from all classes that we trained and tested. Due to the large number of train and test images, we cannot show all the images in this paper, but we have included the full results on our github [8].

Our final model based upon the Zhang et al implementation, which is summarized above in the Methodology section, was evaluated on classification accuracy and cross entropy loss. As described in the evaluation strategy section, since the output of all models based on the Zhang implementation are probability distributions, there is inherent ambiguity in any evaluation of that probability distribution. That is, two extremely similar probability distributions, which may produce image colorizations that are virtually identical to the human eye, still have different probability distributions. Further, if our model classifies some image section as one class and the ground truth for that area is a neighboring class, it is very likely that no human could tell the difference in color, but the evaluation metric captures that as a misclassification. Thus, the classification accuracy will very likely be lower than what seems to be valid for a human looking at the image. Figure 13 shows our classification accuracy and cross entropy loss for our Zhang et al based model. All classes are shown with the train and test sets used - to reiterate, our training set size is 80% of the full class size, which ranged from 4000 to 5000 images. Accordingly, the validation and test sets are both 10% of the full class size. We included cross entropy loss because the neural net was trained using the cross entropy loss function as the function to minimize.

Class	Classification accuracy	Cross entropy loss
Pizza (train)	0.19892676174640656	3.1655542850494385
Pizza (test)	0.1650390625	3.500483512878418
Farm (train)	0.22935253381729126	2.953244686126709
Farm (test)	0.0146484375	4.311539173126221
Forest road (train)	0.34410154819488525	2.6128673763275145
Forest road (test)	0.216796875	2.1972098350524902
Butte (train)	0.2139248102903366	2.9827488651275633
Butte (test)	0.1884765625	2.6482996940612793
Celeb (train)	0.2130419909954071	2.9751222400665283
Celeb (test)	0.197265625	2.725433349609375
Average (train)	0.23986929	2.93790749054
Average (test)	0.1564453125	3.07659311295

Fig. 13. Zhang based model's classification accuracy and cross entropy loss on all classes.

To illustrate the ambiguity, consider the lowest classification accuracy we had throughout our test sets, which was the Farm class with a classification accuracy of 0.0146484375. Figure 14 shows the first 6 images in the test set with model output on the left side and ground truth on the right side.

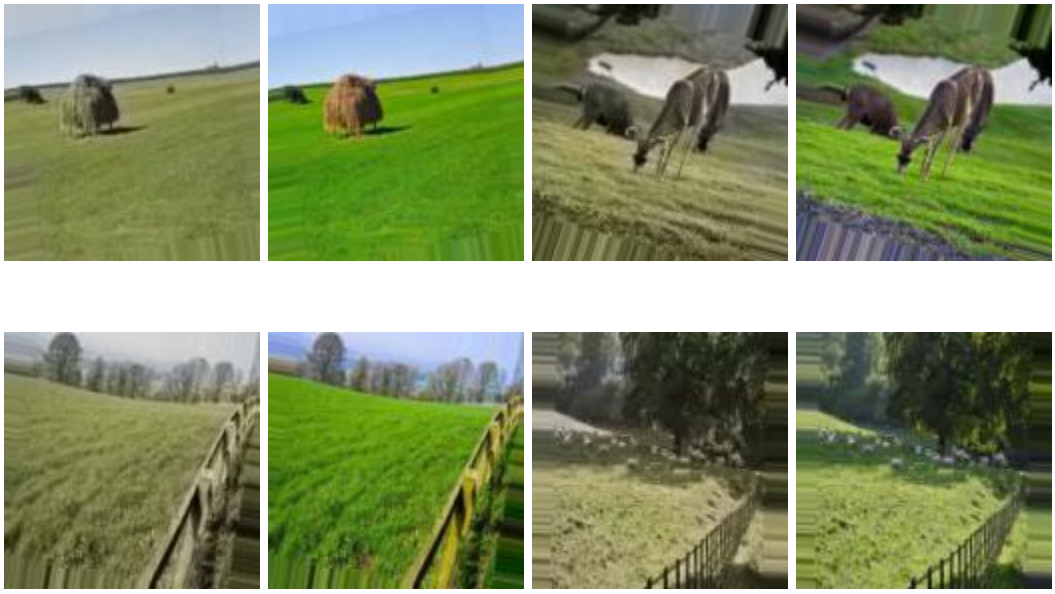

Fig. 14. Comparison of first 6 results vs. ground truth in the Farm class.

If we were to reduce the colorizations into grouping of images and assign those groups a color (e.g. groupings were one of green, red, blue, etc…) then our classification on the images would be very high. However, because the shade of the colors is darker in almost every output image when compared to the ground truth, the overall classification accuracy is very low. This specific evaluation issue is also highlighted below in the mean squared error evaluation of similar colorizations in the Fusion Encoder Decoder - the mean squared error is extremely low in all cases for the Fusion Encoder Decoder but a human looking at the images would certainly notice the images favor darker shades of colors. This discrepancy will be discussed further in the comparisons subsection.

The best performance when evaluated using classification accuracy in our Zhang based implementation was on the forest road class. Figure 15 shows the first 6 images in the test set of the Forest road class. Again, the left image is our model output and the right image is the ground truth.

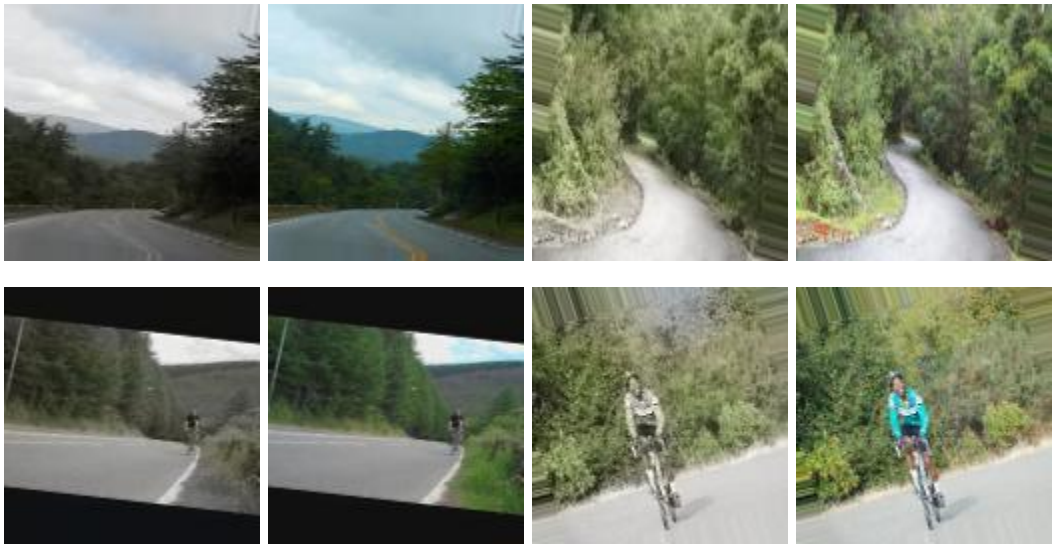

Fig. 15. Comparison of first 6 results vs. ground truth in the Forest road class.

We attribute the relative strength of our model's evaluation of the forest road class to our model's strength in colorizing images that have darker colors. In general, the forest road class has a high density of roads, which are a uniform color in general, and shadows, which are dark colors. We noticed, and Zhang et al [1] highlighted, that image colorizors tend to favor darker colorizations since those colorizations are much more frequent than vibrant colors. Thus, we expected darker ground truth images to have better colorizations and the forest road class seems to confirm that hypothesis.

Interestingly, the Celeb class illustrated that our final model has significant bias against colorizing red hair, which also again highlights the evaluation error. Our goal was to produce a probable image colorization, which was certainly done in the images in figure 16; however, the true distribution of colors and the true colors varied significantly from our output and that variance was for a significant aspect of the picture.

Fig. 16. Comparison of our model's results vs. ground truth to illustrate bias.

In each case in figure 16, the output of our model was a probable image, but favored skin color that was less pale and hair that was brown instead of red. This highlights that the model tends to predict colorizations that are closer to the average of the dataset, which seems to be brown hair and slightly darker skin. Figure 16 also highlights an issue with Zhang et al's evaluation strategy which was to use Amazon Mechanical Turk to pay humans to pick between the model output and the ground truth for which was the real image. I believe the model output in figure 16 looks more realistic, but is obviously the model output and not real. Thus, there is a bias in the interpretation of what a real output would be because we expect hair to typically be some shade close to brown instead of red or bright red (as it is in

both images). Still, we believe this highlights both the efficacy of our model to produce plausible colorizations and the bias associated with producing the most probable image since features of any image can have extremely improbable colorizations in reality.

Next, we discuss our final model based upon the Fusion Encoder Decoder implementation, which is summarized above in the Methodology section. Since the Fusion Encoder Decoder maps an input grayscale image to a colorized image instead of to an *ab* probability distribution, we evaluate the Fusion Encoder Decoder using mean squared error (MSE) and the coefficient of determination (R^2) in figure 17. Given that this is a regression problem, we included the R^2 to illustrate another mode of evaluation instead of only including mean squared error. In our model, we found that the lower the MSE, the higher the R^2 value, in general. This was not a perfect correlation, but it was strong enough to give us confidence in both methods of evaluation.

Class	**MSE**	**R^2 (coefficient of determination)**
Pizza (train)	0.0115	0.3441
Pizza (test)	0.011016242613550276	0.16050229966640472
Farm (train)	0.0124	0.3556
Farm (test)	0.018782748840749264	0.62442014614741
Forest road (train)	0.0075	0.3533
Forest road (test)	0.0041881662845197655	0.49351459079318577
Butte (train)	0.0145	0.2224
Butte (test)	0.016533436076272102	0.37838205837068106
Celeb (train)	0.0075	0.2753
Celeb (test)	0.008440232176023224	0.12973994612693787
Average (train)	0.01068	0.248202
Average (test)	0.01179216519	0.35731180822

Fig. 17. Our Fusion Encoder Decoder model's MSE and coefficient of determination on all classes.

While all of the mean squared errors were small, the best performing model with respect to MSE was again forest road and the second best was, again, the Celeb class. The consistency of strong performance in colorization among these classes could be explained by the relative homogeneity of both forest road and celeb when compared to other classes in our dataset - forest road is generally a road with trees on the sides of the image and celeb is always a celebrity headshot. Again, the worst performer was the farm class, but the low mean squared error highlights that our model's colorizations perform fairly strongly when evaluated as a regression problem where distance to the true color is all that's considered and not the true probability distribution. As was the case in our implementation based on Zhang et al, our implementation based on the Fusion Encoder Decoder produced results that generally had less vibrant colors, as is illustrated in figure 18.

Fig. 18. Comparison of first 6 Fusion Encoder Decoder results vs. ground truth in the farm class.

Again, the model performed reasonably well even in the worst case with respect to MSE (farm class) but produced colorizations that were less vibrant than the ground truth. In this case, the evaluation function on the regression problem was MSE, so the losses are much lower and more accurately reflect that colorization was successful, but tended to produce images that were closer to dark shades, which are more frequent than lighter shades.

As was the case in our Zhang based model, the Fusion Encoder Decoder based model also had bias in colorizing red on someone's head. In general, there was bias in almost all features, but the most notable was red hair or headwear being completely discolored. Figure 19 highlights this bias.

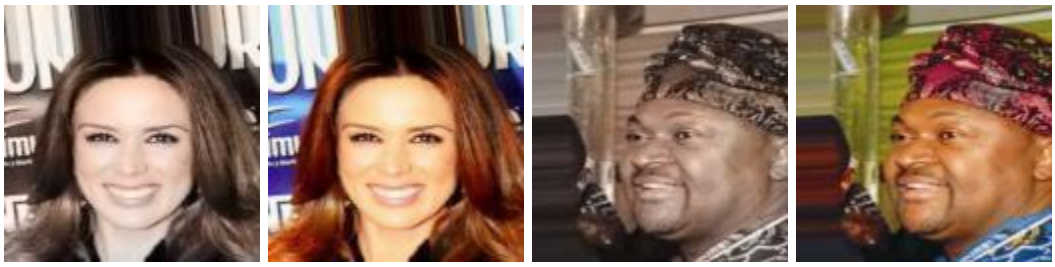

Fig. 19. Comparison of our Fusion Encoder Decoder model's results vs. ground truth to illustrate bias.

One potential solution to this problem would be to balance the training set to have an even split of hair colors, but that distinction is difficult given that hair has shades. Similarly, miscoloring hair as brown is much more likely to produce an image that a human would classify as probable or realistic. On the contrary, colorizing hair incorrectly as vibrant colors like red, green, or blue would likely cause less human classification as realistic. Thus, while that proposed training set configuration would likely produce a more holistic model that allowed for red hair, it would likely confuse our model and colorize images in ways that are less probable, in general.

Of particular interest to our team was that all models produced strong results in colorizing distinct objects as one object. For example, the cat image in figure 7 shows a cat that is colorized as several distinct objects - the figure eight in blue is certainly not part of the cat or it's L channel's distinct objects. However, our models colorized distinct aspects of the image well, which is very hard to capture in an evaluation function but is simple to see in image. Figure 19 highlights that distinct aspects of images in the butte class were colorized as distinct aspects. Specifically, the strips of white across the rockface are appropriately colored differently than the rest of the rock face in the first comparison.

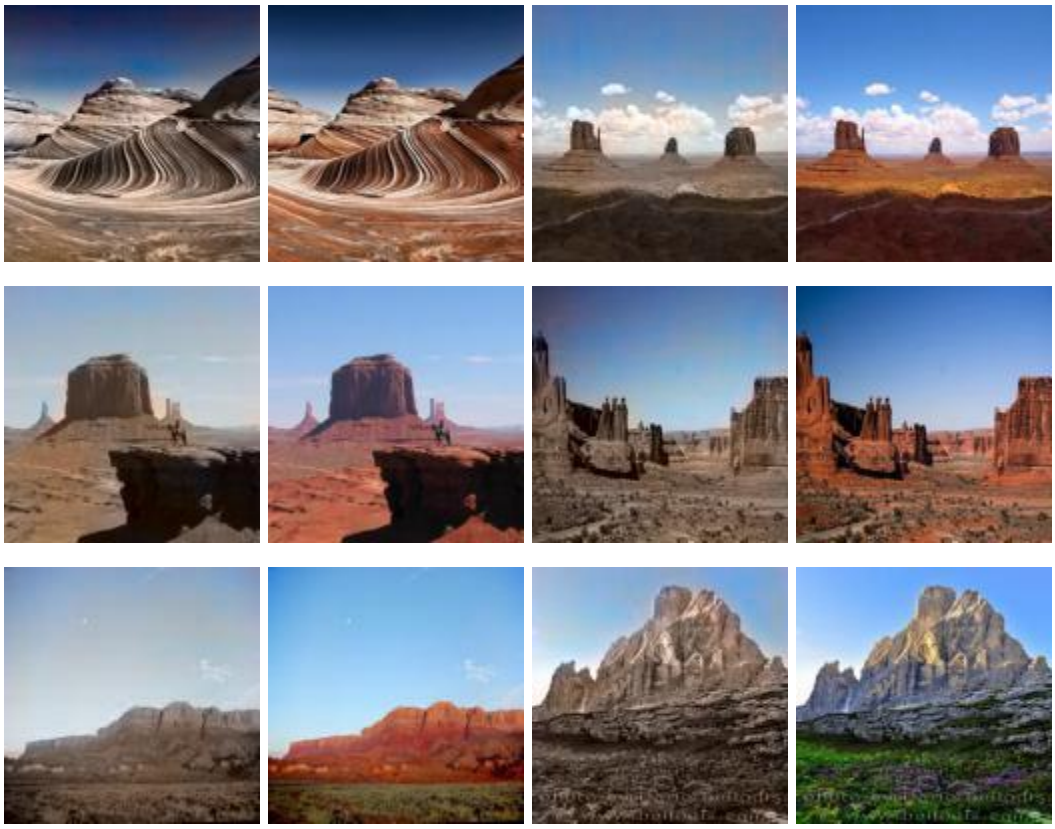

Fig. 20. Fusion Encoder Decoder results vs. ground truth in the butte class.

While the comparisons in figure 19 are not always perfect - often the red is discolored - the different aspects of the image are still clear. For example, in the last comparison, the mountains are still distinct from the grass and sky even though the colorizations are weak. We consider this a strength of both of our models since an image could become very difficult to interpret if the colorization was completely off for distinct images since depth would be very hard to understand in that case.

Comparisons

The evaluation of results subsection largely focused on evaluating interesting colorization patterns we found in the results or highlighting areas of strength or weakness of our model. Below, we show comparisons between our models and the ground truth for one image from each class to illustrate the similarities and differences in the colorizations done by various models. The images are listed in rows by class in this order: Butte, Celeb, Farm, Forest road, and Pizza.

Grayscale	**Zhang based model**	**Fusion Encoder-Decoder based model**	**Ground truth**
			
			
			

Fig. 21. Comparison of results between models and ground truth

In most cases we can see that our models output similar image colorizations even when they do not match the ground truth, which is the case in the pizza class example. Additionally, we tried to include images that were good representations of most of the test class results. Thus, we chose images that highlight a pattern we saw, which is that the Zhang based model generally outputs colorizations that are closer to the ground truth than the Fusion Encoder Decoder. Similarly, we noticed the Zhang based results output images with greater vibrance in the colorizations. This is particularly noticeable in the Celebs example in figure 21 where we can see that the face is considerably paler in the Fusion Encoder Decoder output.

Conclusion

We believe our models performed very well in the vast majority of cases with some instances of model bias and undercolorizations for vibrant colors. Similarly, we are happy with the results after we reduced the problem space to only training models on specific classes. A natural next step of our project would be to attempt to train the model on all classes and test on images from various classes, as well. We believe that is an achievable goal using something like the discovery cluster and the full Places 365 dataset, but the time per iteration on each change to any given model may be several days given that results would need that long to be created. Thus, tweaking hyperparameters is likely a difficult short term task in that case.

Our implementation worked fairly well and we attribute that mostly to our neural network architecture and that we reduced the problem space to fit within the bounds of the semester. Our main

hurdle was finding a meaningful evaluation technique that correctly captured what we believed to be a "good" colorization versus a "bad" colorization. While we do believe the metrics we found all provide insight when considered relative to each other (e.g. determining if the celeb class is colorized better than Butte) we do not believe that we solved the evaluation problem perfectly. Our initial hope was to evaluate with some method similar to the Zhang et al. evaluation method where we ask individuals to choose the real image between the model output and the ground truth, but that would require monetary investment on our behalf, so we opted not to do so. Similarly, we considered and tested mean pixel error of the entire data set and of individual images, but again, that failed to capture the nuances of coloring images like those highlighted in figure 12. We also attempted to use a clustering algorithm to analyze the efficacy of colorization on specific colors in an image, but we were not able to successfully implement that algorithm due to time constraints. We believe that the most natural extension of this project would be to develop an evaluation technique that can be used on both the Zhang et al based implementation and the Fusion Encoder Decoder based implementation. One idea our team had was to ask a human to rank an image colorization from 0-10 without seeing the ground truth based on if they thought the image appropriately showed colors, then training a model on what colorizations were "good" as rated by the humans and "bad" as rated by the humans. This could be one method to move towards eliminating bias in the training set.

Finally, our team also believes that a natural extension would be to create an approximate classification accuracy. That is, we would classify ab colors into bins of similar colors or similar (r,g,b) colors depending on the problem formulation. Then, when we calculate approximate classification accuracy, a specific classification is classified perfectly if it falls in the same bin as the true classification. Through this process, one would be able to find if images were colorized close to optimally, which we believe is a better evaluation of appropriate colorization.

Individual tasks

Since everyone in our team is technically sound enough we thought to have different implementations of the paper and pick the best one in the end. We all explored the whole web and found that there is no legit open source PyTorch implementation for this project. Hence, I decided to implement it in PyTorch, while Nolan and Naveen explored Tensorflow implementation individually. Nolan and Naveen have explored most of the Fusion Encoder Decoder architecture. I worked with producing colorizations for Zhang architecture training against CIFAR-10 dataset. Everyones of us was successful with implementations and produced good colorizations. Later, Nolan started work on exploring various evaluation strategies for the model. Naveen had an out-of-the-box idea to reduce the training time by

modifying the architecture of CNN. He was successful in his attempt and finally we had a very simple model that produced almost similar colorization with a few minutes (40 min) of training per class. Everyone of us was happy with this outcome and so we finally decided to use this model. Then I suggested trying basic multiclass Cross-Entropy loss function instead of rebalanced multiclass Cross-Entropy loss function and surprisingly we got similar colorizations with even less training time (15-20 min per class). Overall, the collaboration in our team was pretty smooth. We used to regularly connect over Microsoft Teams. Although, most of us worked independently in the beginning, we finally collated all the results and wrote the report together. Nolan's help to produce the literature with a fine quality is invaluable. I drew the graphics for the neural networks using the PlotNeuralNet tool available on GitHub. Working with my teammates was real fun and I would love to collaborate with them again in the future.

References

[1] Richard Zhang, Phillip Isola, Alexei A. Efros. Colorful Image Colorization (https://arxiv.org/pdf/1603.08511.pdf).

[2] Federico Baldassarre, Diego Gonzalez Morin, Lucas Rodes-Guirao. Deep Koalarization: Image Colorization using CNNs and Inception-Resnet-v2. (https://arxiv.org/pdf/1712.03400.pdf).

[3] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. CIFAR-10 Dataset (https://www.cs.toronto.edu/~kriz/cifar.html).

[4] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Antonio Torralba, Aude Oliva. Places Dataset (http://places2.csail.mit.edu/index.html).

[5] Stanford Vision Lab, Stanford University, Princeton University. ImageNet Dataset (http://image-net.org/index).

[6] PyTorch torchvision package. (https://pytorch.org/vision/0.8/datasets.html#imagenet).

[7] Lecture slides from Richard Zhang presenting Colorful Image Colorization. (http://videolectures.net/site/normal_dl/tag=1078750/eccv2016_zhang_image_colorization_01.pdf)

[8] Nolan Bock, Sai Nikhil Thirandas, Naveen Verma. Our group's implementation of all models. (https://github.com/naveen-verma19/CS6140-Image-Colorization)

[9] Ziwei Liu, Ping Luo, Xiaogang Wang, Xiaoou Tang. Large-scale CelebFaces Attributes (CelebA) Dataset. (http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html)

[10] Nolan Bock, Sai Nikhil Thirandas, Naveen Verma. Our group’s full train and test image dataset. (https://drive.google.com/file/d/1fjim4AO1qUFtONpb8RGC92kyquY3HfHc/view?usp=sharing)

[11] Nolan Bock, Sai Nikhil Thirandas, Naveen Verma. Embeddings for the Fusion Encoder Decoder. (https://drive.google.com/file/d/1zsYf9xgBWV728qBzzQcd5S9TFzw3Z2PH/view?usp=sharing)

[12] Keras Documentation. (https://keras.io/api/)

[13] Data Augmentation. (https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html)

[14] PlotNeuralNet. DOI: 10.5281/zenodo.2526396 (https://github.com/HarisIqbal88/PlotNeuralNet)