Learning Distributions of Meant Color

Anonymous IJCNLP submission

Abstract

When a speaker says the name of a color, the color they picture is not necessarily the same color the hearer imagines. Color is a grounded semantic task, but that grounding is not a single value, but rather a range of possible values that could be intended. To handle this case, we propose a model that given a input color description such as "light greenish blue" produces an estimated probability distribution across HSV color space. This work presents a method for estimating probability distributions, based on samples using a description processes. Predicting distributions is useful beyond regressing to a single output for handling cases where the distribution is not simply a true value plus noise, but rather is actual feature of the population's varying conception featuring wide variance, and a potentially multimodal nature. We demonstrate a GRU-based neural network learning the grounded compositional semantics of the terms used with in a the color description. By learning per-term, rather than per whole description, the model is able to predict distributions that for combinations of terms that are not seen in the training data. The ability to predict distributions is useful as a component in human computer interaction systems.

1 Introduction

When a person says "tan" they may mean a number of colors: From the bronze of a tanned sunbather, to the brown of tanned leather. When they say "green" they may mean anything from "aquamarine" to "forest green"; and even "forest green" itself may mean the shades of a "rain-forest", or of a "fir-wood". Thus the color can not be deterministically known from the color name. However, based on knowledge of the population's use of the words, a probability distribution as to the color intended can be found. Here issues of illumination and perceived color based on context will be disregarded, to focus on the core problem of the color of a single patch.

Color understanding is a core subtask in natural language understanding. The color language subdomain displays many of the same features and difficulties as natural language as a whole. Every word has shades of meaning. No one understands colors exactly the same. Words occupy multiple roles: "pale" can be used to describe a all *pale* colors, or as a modifier: "pale blue". Basic colors themselves can act both as modifiers, and as targets: "blue green". Modifiers do not act on all colors constantly. New color descriptions are brought in from other sources, such as "salmon", and "coral" which come from the colors of the objects they describe. We note the noxious examples of "puke", "vomit" and "yuck" which not only are as consistently used as any color like "bright orange", but are also examples of nearly perfect synonyms describing the same area in color-space. Many of the problems of natural language are exemplified in their use in colors. Recent state of the art systems for image generation has demonstrated their capacity by generating from texts containing complex color descriptions such as "the flower has petals that are bright pinkish purple with white stigma" (Reed et al., 2016; Mansimov et al., 2015). Understanding color is crucial to understanding language.

The mapping from color name to color could be considered a regression problem. Solving to find a function that when input some text such as "forest green", outputs a numerical value in a color space such as HSV or RGB. However, regression discards information about the distribution. If the

distributions in color space were mono-modal [1] and symmetric with consistently small variance, then considering the problem as regression with noise would be adequate. If the distribution were multi-model (e.g. a mixture model), or non-symmetric (e.g. a truncated distribution) or with varying and wide variances, then regression is loosing valuable information and is unable to produce a model that aligns well with reality. At the other end from regression, is classification.

A classifier will output probabilities for each of the possible categories an input could belong to. By dividing color space into threshold bins, where each bin is its own category. Then by classifying a color, one gets getting probabilities of it laying in each bin. The output of the classifier defines an empirical distribution in color-space. Basic classification models consider each category as being distinct and unrelated. However, we know that if a color name has a high probability of corresponding to a particular point in color space, then it should have a similar probability for other points in that neighborhood – this is a notion of continuousness. A variety of approaches called ordinal regression or ordinal classification exist to handle this case, where there is a order to the categories. However, there is no natural total ordering of colors. So classical ordinal classification methods have limited utility on the problem.

We instead look to helping a normal classifier learn the continuous relationship between adjacent bins by enhancing the training data through a blurring process.

The core of this work is mapping from natural language space, to color space. This goes beyond direct one-to-one color generation. Given a color name, probability distributions in color space is generated. These distributions can be sampled, or the peaks selected, to generate colors. However, they have further use, as the whole distribution is known. For example, as a subsystem in human interfacing image processing, when asked to select the "dark bluish green" object, each object can be ranked based on how likely it's color is according to the distribution. This way if extra information eliminates the most-likely object, the second most likely object can immediately be determined. Further, as the probability of the color of the object being the color being described by the user input is known, a threshold can be given to report no object found, or to ask for additional confirmation.

[1] It should be understood that in this paper, when say *mono-modal* or *multimodal* it is meant in the sense of the number of peaks in the probability distribution; not in the sense of the number of modalities of the data – e.g. multimodal audio-visual data. McMahan and Stone (2015) call this convex, we prefer the term multimodal.

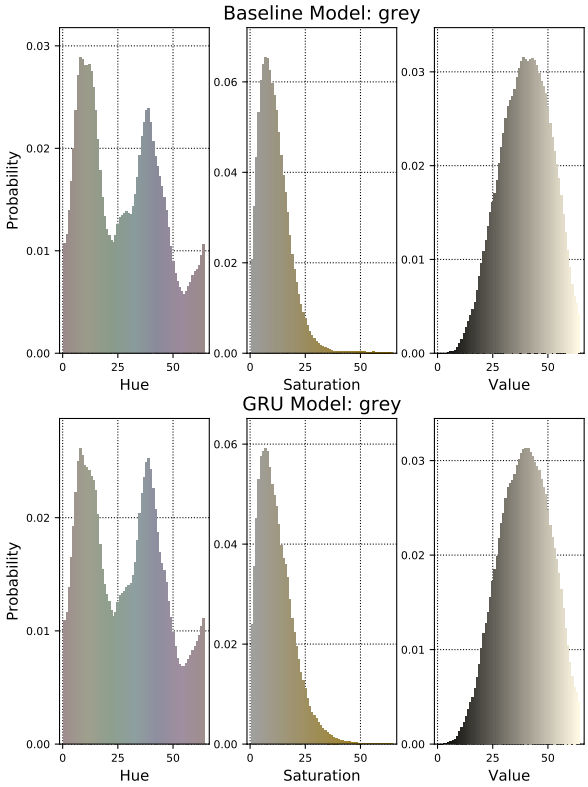

Figure 1: grey

2 Highly multimodal colors

One of the core motivating factors of this work is to be able to handle the color names which have multiple distinct modes. That is to say there are distinct peaks of the most likely region in color space. McMahan and Stone (2015) identify "greenish" as a convex color, i.e. one with a multimodal distribution. We further identify several others "purplish grey", "purplish" and "blueish" amongst them to varying extents. Though this is not true for all "-ish" colors: "reddish", "orangish", "yellowish" are . We also note this for shades of grey – where hue is traditionally considered not to matter.

One such color with a significant bi-modal distribution is "grey". Traditionally, "grey" has been considered to be achromatic – that is to say its hue component does not matter. However, by looking at the data from the Monroe dataset (Munroe, 2010) in Figure 1 it can be seen that that a the hue

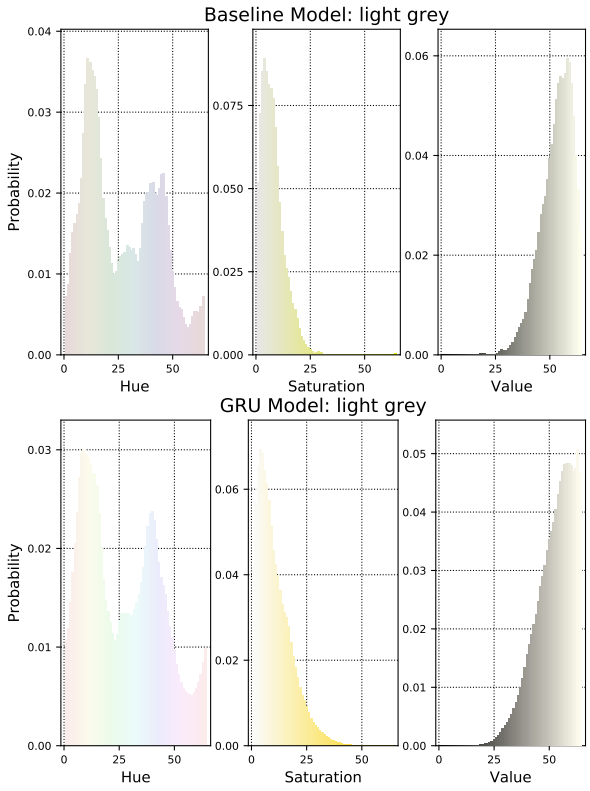

Figure 2: lightgrey

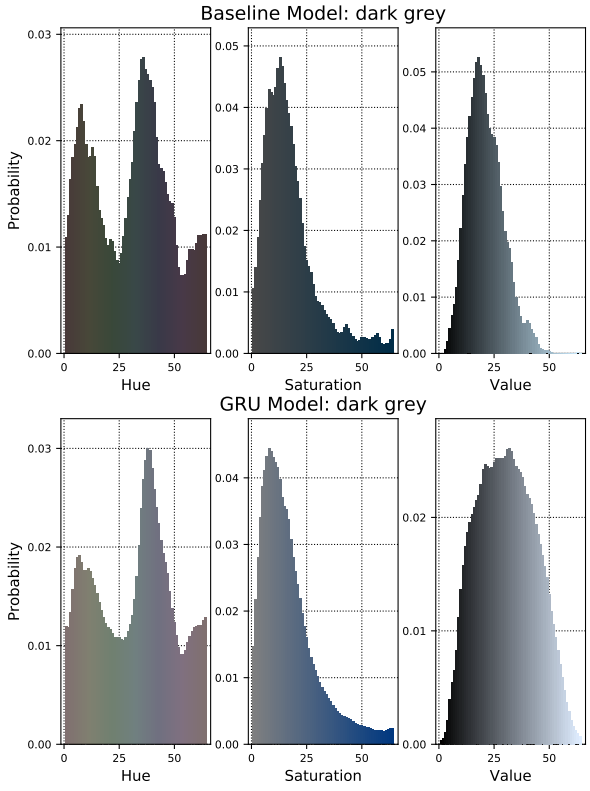

Figure 3: darkgrey

distribution significantly favors blues and reds over greens and purples. Further, "light grey" increases the yellow peak (Figure 2), and "dark grey" (Figure 3) increases the "blue" peak, while also changing the value dimension, as expected. Other multimodal colors include "greenish" which is less likely to be a pure green, than to be on the blue or yellow side of the color; and "purplish grey" which has a dip at "magenta". These colors are discussed further in Section 10.1.

A core interest here is in colors with a multimodal and asymmetric distributions; such colors can not be considered as targets for regression as they do not have a symmetric noise around their mode. As such they can not be estimated by using distance in color-space as a proxy for the probability of intent.

3 Related Work

3.1 Color Naming

Color naming is the reverse of the task investigated in this work. The color naming task takes a point in color-space as in input, and outputs a probability distribution over possible names for that color. There a several notable recent works on color-naming. Meo et al. (2014) and McMahan and Stone (2015) present a full description Bayesian approach, which outputs the probability of a whole description. Monroe et al. (2016) presents a per-word LSTM approach, which produces a conditional language model – sequentially outputting a probability of each word in the description. Kawakami et al. (2016) presents a per-character LSTM and Variational Autoencoder approach, which products a conditional character language model – sequentially outputting a probability of each character in the description. The work by Kawakami et al, also includes a method for generating colors.

3.2 Color Generation

Color generation closely related to the primary task considered here. The process of going from the name of a color, to an actual color – a single point in a color space. Kawakami et al. (2016) presents a method using RNN, and LSTM, as well as baselines using unigram and bigrams, over characters, to predict a point in *Lab* color space (Hunter, 1958). Color generation is the single output version of our task of color distribution estimation.

Color generation system systems outputting a single color can approximate a probability distri-

bution by using the distance in color space, from a observed color to the predicted color as a proxy for the probability of the observed color. However, this does not handle asymmetric, or multimodal distributions, nor does it take into account that the range of values reasonable for one color description, often significantly differs in width from that reasonable for another.

4 Color Identification

Monroe et al. (2017) presents a neural network solution to communication game, where a is presented with three color patches and ask to describe one of them, such that when the listener is presented with the same color patches in randomized order they can select the one the speaker was describing. In this game, the color descriptions have context, for example the speaker can say "the darker blue one, not the cyan". Both speaker and listener models are trained, using and LSTM based decoder and encoders respectively. They present several variants of their models, including pragmatic models, and models that combine knowledge. The base listener model is of particular relevance to the distribution modeling task. The final time-step of the base listener LSTM produces a 100 dimensional representation of description provided. From this, a Gaussian distributed score function, over the Fourier-transform based color space of Monroe et al. (2016). By normalizing the scores of the three colors the listener is to choose from, the conditional probability of each can be found. It should be noted that while this method does output a probability distribution, as a by-product of the task it is Gaussian distributed for all inputs – both symmetric and mono-modal, albeit in a high-dimensional non-linear color space. This is arguably reasonable distribution for use in this color-game, where the speaker is expressly trying to avoid ambiguity, and where color descriptions can feature reference to other colors in the context. Without this contextual information in the color-naming, distributions over all colors are required, and these distributions are not expected to be symmetric in any consistent color space.

5 Method

6 Blurring and Discretization

To allow the network to readily learn the distribution of the colors the task is changed from learning a conditional continuous distribution to the well-established problem of learning a conditional discrete distribution. We discretize to a resolutions of 64, and 256 bins per channel. For the case of 256 bins per channel, there is effectively no information lost in a discretized representation as the original data was collected using a web interface displaying 24 bit color (Munroe, 2010). The discretization process is as follows.

First, a blur is added to each observation. This is done by defining a distribution with expected value equal to the observation, and with variance defined as a hyper-parameter of our process. Saturation and Value are given truncated Gaussian distributions. Hue is given a wrap-around Gaussian [2]. To discretize the distribution, the support is partitioned into a number of equally sized bins. The cumulative distribution is evaluated between each bin boundary, resulting in a vector of values between zero and one – summing to one. Functionally, this is very similar to converting to a one-hot representation, based on bin boundaries, but with some blurring to shift part of the mass into adjacent indices.

The blurring during the discretization encoded the prior knowledge of the smooth relationship between adjacent output bins. The outputs from softmax output layer are intrinsically unordered and purely categorical, this relationship must be trained into the network. By blurring all training cases, it ensures that knowledge learned. This effectively penalizes the network for outputting very different values for adjacent output bins. A nearly equivalent formulation could be defined for the loss function, were one-hot output encoding used. The blurring level is effectively a hyper-parameter in training.

There is a trade-off, in blurring level. With very small blurring level the model is not informed of the smoothness – there is no penalty for sharp differences. With high blurring there is less overall penalty for incorrect predictions, and the model is unable to fit sharper shaped curves.

To determine the blurring level we conducted a coarse hyper-parameter sweep using the develop-

[2] In implementation, the wrapped normal distribution was initially approximated with a von Mises distribution with support between 0 and 1; however the calculating cumulative distribution for this was computationally expensive; so it was switched to a truncated Gaussian with the support between -1 and 2, which allowed for very fast implementation by aliasing the memory locations outside the true value's support of 0 to 1. The difference in value is negligible until variance becomes much large than is considered here, as the values become zero far before reaching the ends of the extended supports

ment dataset. Best results were found for were to set the standard deviation of the distributions used in the discretization process to be $\sigma = \frac{1}{2n}$, where n is the output resolution. The blurring of $\sigma = \frac{1}{2n}$, redistributes the probability mass assigned in the discretization, to the surrounding bins. For a training point that would be at the center of a bin, this roughly corresponds to 68.3% of the probably mass assigned to the central bin, 15.7% assigned to the bin on each side, and the remaining 0.3% distributed to the remaining bins. However, in general points are not aligned to the center of bins, so they generate asymmetric training cases. All results presented here are for this value of the blurring hyper parameter. Further tuning of this parameter might enable better results – particularly using different blurring levels for the difference channels.

7 Conditional Independence Assumption

For HSV colors we make the assumption that given the name of the color, then the distribution of the H, S and V components are independent. That is to say, we assume that if knowing the value of one component would not inform us as to the value of the other if we already know the name of the color. This assumption is incorrect, but not as incorrect as might be suspected.

Superficial checks were carried out the the accuracy of this assumption. The Spearman's correlation on the training data suggests that for over three quarters of all color names, there is only weak to zero maximum pairwise absolute correlation between the H,S, and V components (Q3 = 0.187). However, this measure underestimates correlation for values that have circular relative value, such as hue. This correlation measure was the lowest by a large margin when compared amongst 16 color spaces; RGB, HSV, HSI, HSL, xyY, XYZ, CIELab, Luv, LCHab, LCHuv, DIN99, DIN99d, DIN99o, LMS, YIQ, and YCbCr; by a full 100%. The table is available in the supplementary materials. Given the limitations of the evaluation methodology, these results are suggestive, rather than solidly indicative of the degree of correctness of the conditional independence assumption. For the investigation here, we consider the conditional independence assumption sufficient.

Note that the evaluation metrics chosen do not assume conditional independence. Though the models, including the baseline model, do. Better results may be obtained by outputting a 3D joint distribution.

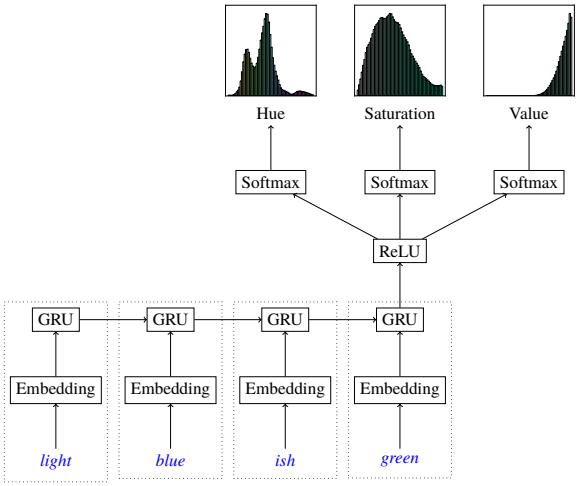

Figure 4: The GRU Model for predicting the color-space probability distributions of color. The section in the dotted-boxes is repeated for each time step.

The conditional independence assumption is not intrinsic to our method. The discretization and blurring function identically in 3D joint distribution space, as in three, 1D independent distribution spaces. However, the independence assumption significantly decreases the computational requirements. Assuming independence allows the model to be defined with 3 output layers each containing n elements, where n is the output resolution (number of bins). Rather than one output layer containing n^3_{res} elements. This saves a large amount of memory during the training.

8 The Models

8.1 GRU Model

We present a neural network based model, with sequential inputs which predicts the 3 separate output distributions. The general structure of this network is similar to Monroe et al. (2016), or indeed to most other word sequence learning models. It is shown in Figure 4. Each word first is transformed to an embedding representation. This representation is randomly initialized and is trained with the rest of the network. The embedding is used as the input for a Gated Recurrent Unit (GRU) (Cho et al., 2014; Chung et al., 2014). The output of final time-step is feed to a Rectified Linear Unit (ReLU) (Dahl et al., 2013). Finally, the this used as the input the three distinct softmax output layers – one for each of hue, saturation and value. The network was trai-

ned to minimize to sum of the three cross-entropy losses of these output layers. The multiple output layers commonly occur joi[illegible]arning and related transfer learning problems.

We choose GRU as the basis of our reused structure in the recurrent network. GRU has fewer parameters to learn than the more established LSTM. It has generally been found to preform similarly well to LSTM (Chung et al., 2014); including on the color naming problem (Monroe et al., 2016). The GRU forms the basis of the our sequence-of-terms to color-space-probability- distribution network.

8.2 Baseline Model

For comparison we define an additional model based more directly on the training data. This is a simpler model with no machine learning component. For this model, the estimated distribution for each color description is produced by averaging all of that colors discretized observations. This features the same blurring, as in the GRU Model. During our investigations we found that a model based only taking on the mean would return a predicted probability of zero for some of observations in the development dataset. This causes the perplexity to be undefined (or ∞ when evaluated using IEEE floating point math.) To handle this add-one smoothing is applied to each output distribution. Effectively this is adding an number number of additional training observations for each color name, corresponding to a one-hot vectors for each output bin. The result of this is that when the mean over all observations is taken, there a no output bins with a probability mass of zero. The Baseline model can be used to predict distributions for all color descriptions in the training set. This is inferior in generalisability to the GRU model, which can handle any combination of tokens from the training set. Without the requirement to learn the how the compositional structure of the terms in the color name function, it is a much simpler modeling problem, as such we suggest it is a strong baseline for evaluational.

9 Experimental Setup

9.1 Data Preparation

We use the Monroe dataset (Munroe, 2010), as prepared by McMahan and Stone (McMahan and Stone, 2015). This dataset is partitioned into test, development, and training sets; and has had some cleaning from the original data collected by Randell Munroe (2010). It is also used by Will[illegible]nroe et al. (2016) and Kawakami et al. (2016). The color space is HSV, with all values between zero and one. Each is pair with a short name for the color, and provided by participants in the Color Survey.

The text descriptions are loosely tokenized into separate words and affixes. Beyond simply breaking up a description "greenish blue" into words: "greenish", "blue", the suffixes "-ish" and "-y' are also separated at their own tokens: "green", "ish", "blue". This tokenization is achieved through a short list of word replacement rules. Hyphens are also treated as their own tokens: "blue-green" becomes "blue", "-", "green". The beginning and end of the color description is not demarcated with any form of marker token.

The Monroe dataset has 829 unique color descriptions. Each description has a varying number of observations, where each observation is a pairing of description and a point in HSV space. Using the tokenization described above, each description is split into between one and four tokens. This results in a total of 311 unique tokens.

9.2 Extrapolation Sub-Dataset

One of the key advantages of our proposed system is its ability to predict the distribution for never before scene descriptions of colors. For example, based on the learned understanding of "bright", from examples like "bright green" and "bright red", and of "salmon", our system can suggest the distribution in color space of "bright salmon", even though that color never occurs in the training data. To evaluate this, a new dataset is derived from Monroe dataset, which we will call the extrapolation sub-dataset. This is defined by selecting the rarest 100 color descriptions, with the restriction that every token in a selected description must still have at least 8 uses in other descriptions. The selected examples include multi-token descriptions such as: ""bright yellow green" and also some single tokens that occur more commonly as modifiers than as stand-alone descriptions: "pale". The test and development datasets are restricted to contain only observations of these selected color descriptions. Conversely, the extrapolation training set has no observations of these color descriptions. This produces a dataset suitable for evaluating the capacity of our model to estimate the distributions for color descriptions not seen in training.

9.3 GRU Model Parameters

For the GRU model, regardless of output resolution the same network parameters are used. All hidden layers have width 128, except the embedding layer with width 16. These values were found on a coarse search of the hyper parameters using the development portion of the data set with the output resolution being 64 bins. These parameters were also used for the 256 bin output resolution, to simplify comparison, though we suggest increasing the hidden layer size would give additional benefit for the higher output resolution case. During the hyper-parameter search, it was noted that the accuracy continued to improve as hidden layer width was increased, however significantly diminishing returns in terms of training time vs accuracy lead us to limit the hidden layer sizes. Dropout (Srivastava et al., 2014) with a probability of 0.5 was used during training, on all hidden layers, except the embedding layer.

9.4 Evaluation Metrics

We propose two key measures of evaluation: Perplexity, and Mean Squared Error. The Perplexity allows us to evaluate how well our estimated distribution matches the distribution of the observations in the test set. Perplexity is commonly used for evaluating language models, however here it is being used to evaluate the discretized distribution. It can loosely be through of as to how well the model's distribution does compared to a uniform distribution – which has a perplexity equal to the number of bins.

We define perplexity per channel of the color-space, and also report the geometric mean of the perplexities, to give the perplexity of the whole space. For τ the test-set made up of pairs consisting of a textual color name t, and color-space observation (v_H, v_S, v_V). We define $p_c(v_c \mid t)$ giving the predicted probability of the observation v_c being the color described by the color name t, in the given channel c. This is found by determining which of the discretized bins from the model's output the v_c would lay in, and giving the probability mass of that bin. From this perplexity is given by

$$PP_c(\tau) = 2^{-\left(\frac{1}{|\tau|}\sum_{\forall(t,(v_H,v_S,v_V))\in\tau}\log_2 p_c(v_c \mid t)\right)}$$

We also define the over-all perplexity by the geometric mean:

$$PP(\tau) = \sqrt[3]{PP_H(\tau) + PP_S(\tau) + PP_V(\tau)}$$

As the perplexity varies depending on the output resolution, we will also consider when comparing models of different resolution the standardized value of $\frac{PP_c(\tau)}{n}$, where n is the output resolution of the model. Using this standardized perplexity, if the model always output a uniform distribution then no matter the output resolution n it would always be true that $\frac{PP_c(\tau)}{n} = 1.0$. Perplexity is a measure of how well the distribution estimated by the model, matches reality according to the observations in the test set.

As a second measure, we use the mean squared error to peak (MSE). This is useful in the monomodal symmetric case, and allows out model to be compared to regression models. To do this, the output bin with the highest probability according to $p_c(v_c \mid t)$ is found and it's index is used to find the into the continuous space value at the center of the bin's range. The mean square error is found in the transitional way, averaging over the three channels. That is to say, with the definitions as before:

$$binpeak_c(t) = \arg\max_{1\le i\le n} p_c\left(\tfrac{i}{n} \mid t\right)$$

$$peak_c(t) = \frac{binpeak(t)}{n} - \frac{1}{2n}$$

$$SE(t,(v_H,v_S,v_V)) = \frac{1}{3}\sum_{\forall d\in H,S,V}(peak_c(t) - v_d)^2$$

$$MSE(\tau) = \frac{1}{|\tau|}\sum_{\forall(t,(v_H,v_S,v_V))\in\tau} SE(t,(v_H,v_S,v_V))$$

The space that the error is measured in is the 0-1 scaled HSV space that is used in the source dataset. This measurement of the error to peak is the error that would be obtained if a single color output was required, and that color was chosen in a greedy way.

9.5 Implementation

The implementation of the models and all evaluations was made in the julia programming language (Bezanson et al., 2014), using the bindings for TensorFlow (Abadi et al., 2015). The full source code is included in the supplementary materials.

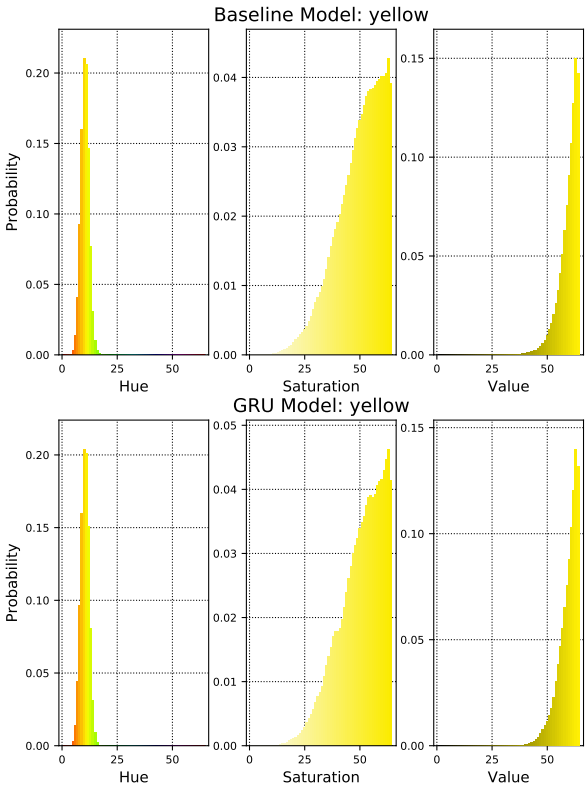

Figure 5: yellow

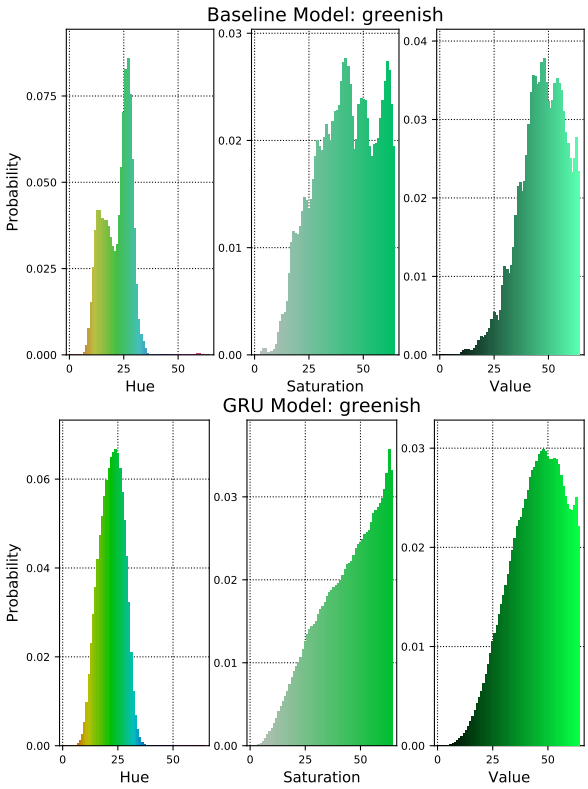

Figure 7: greenish

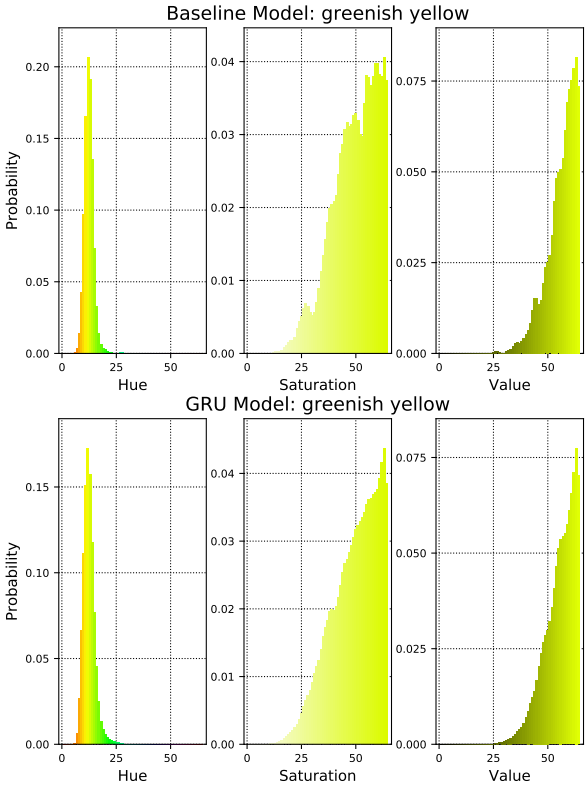

Figure 6: greenishyellow

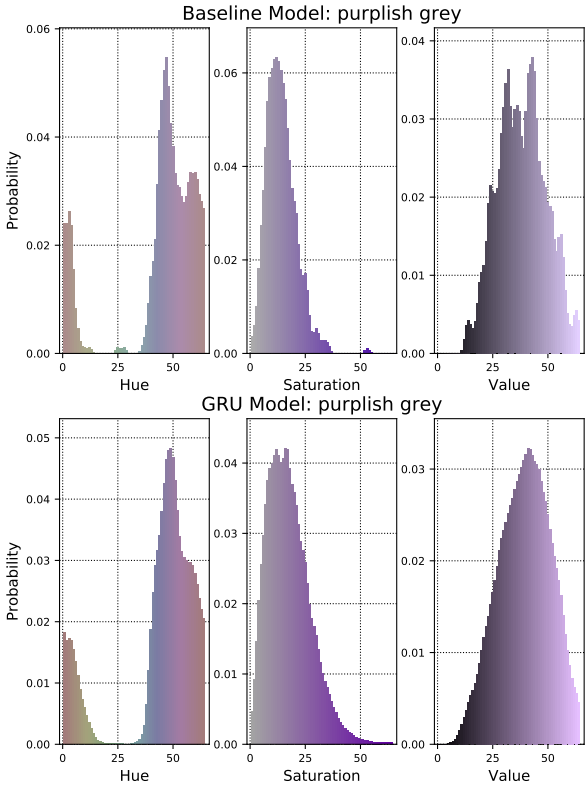

Figure 8: purplishgrey

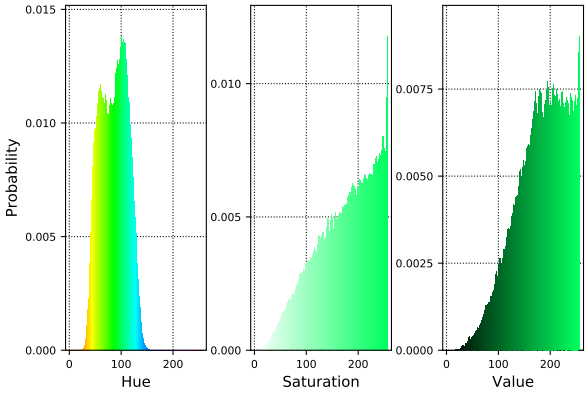

Figure 9: The distribution output by GRU model with resolution 256, and blurring $\sigma = \frac{1}{8n}$

10 Results and Discussion

10.1 Qualitative Comparison of the Distribution

Shown in Figure 5, Figure 6, Figure 7, Figure 8, and earlier in Figure 1, Figure 2, Figure 3 are side by-side comparisons of the output of the 64 bin GRU model, compared to the Baseline model. Overall, it can be seen the Baseline model is a lot sharper, with more spikes, where as the GRU model tends to be much smoother, even though both use the same blurring during discretization. Close together peaks, seem to be leveled out, with the valley between being filled. It can be noted in Figure 5, Figure 6, Figure 7, that small close peaks tend to be smoothed between. Where-as more separated peaks, as in the hue component in Figure 1, Figure 2, Figure 3 more distinct peaks remain separate. We note that smoothing, by partially filling valleys is functionally similar to the fuzzy rectangles used in the input processing of McMahan and Stone (2015). Also like in their work, this filling in results in "greenish" being filled in.

We did find that when the blurring level was made very small, it was possible to see the two peaks in "greenish". This can be seen in Figure 9. However, this comes at the cost of overall worse performance, as the model does not learn about the smoothness of the distribution.

To enhance the capacity to model the different ways a word can be used in a color description we suggest a parsing step could be added prior to any modeling to tag each token with a linguistic role. We suggest that this would improve the capacity of this, and other related models to handle cases such as "greenish". The two uses of "greenish", as a color and as a modifier (e.g. "greenish blue") is currently both supported by the same embedding layer representations. Leaving the output layers to determine the shape of the curve when it is used on its own. By adding additional role labels by a parsing step these cases could more easily be distinguished and given separate representations.

10.2 Distribution Estimation

The primary task here is the estimation the distribution in color-space for a given color description. The results are shown in Table 1. It can be seen that all models perform similarly. This shows that the GRU model is fitting correctly. The GRU model basing on its input sequence of color tokens, reflects real use of the terms in the test set; equally well as the non-compositional Baseline, that counts exact uses of whole descriptions. Across all models, the perplexity for the hue channel is much smaller than for the saturation or value channels. This suggests that in the data there is more consistency in the hue, associated with a color name, than with the This aligns with the notion that people describe color primarily with reference to the hue, rather than the shade. It also aligns with the notion that how *dark* for example "dark blue" is, is not a precise quantity. The GRU model performs similarly to similarly to the baseline, when trained on a full set of color terms with all combinations of terms present in the training data. The key advantage of the GRU model is its ability to predict a distribution for an unseen combination of colors, this is evaluated using the extrapolation task.

10.3 Extrapolation

A core motivation of using the GRU mode, over the Baseline, is its ability to learn to combine tokens in a description in ways not seen in training. To evaluate how well the model does at predicting these distributions, we compare a GRU model trained on e the extrapolation sub-dataset, to the models trained on the full dataset. Both the non-extrapolating, and extrapolating models are evaluated on the same set of rare color descriptions, but the non-extrapolating models are also shown these rare descriptions during training. The extrapolating model has never been trained on these combinations of color terms, and instead must use the knowledge of how those color terms influence the colors in other cases.

The results for estimating the distributions for the rare color descriptions in the extrapolation sub-

model	n	PP	MSE	PP_S	PP_H	PP_V	$\frac{PP}{n}$
GRU	64	27.24	0.1426	41.83	15.35	31.49	0.4257
Baseline	64	27.19	0.1364	41.72	15.32	31.43	0.4248
GRU	256	106.7	0.1559	164.5	60.09	122.7	**0.4167**
Baseline	256	110.1	0.1489	167.5	62.93	126.4	0.4299

Table 1: The results of evaluation on the full Monroe color dataset. Here n is the output resolution of the model, PP is the perplexity, and MSE is the mean squared error to the peak of the output distribution.

model	n	PP	MSE	PP_S	PP_H	PP_V	$\frac{PP}{n}$
Extrapolating GRU	64	27.35	0.1774	41.4	15.84	31.19	0.4273
Non-extrapolating GRU	64	24.78	0.1644	40.06	12.68	29.95	0.3872
Non-extrapolating Baseline	64	26.24	0.1355	40.71	13.91	31.88	0.41
Extrapolating GRU	256	108.8	0.2072	165.3	62.31	125	0.425
Non-extrapolating GRU	256	94.77	0.1668	152.8	48.31	115.3	0.3702
Non-extrapolating Baseline	256	128.9	0.1391	186.1	74.89	153.6	0.5035

Table 2: The results of evaluation on the full Monroe color dataset. Here n is the output resolution of the model, PP is the perplexity, and MSE is the mean squared error to the peak of the output distribution.

dataset are shown in Table 2. It can be seen that the extrapolation is successful, the results on the extrapolation sub-dataset are similar to the overall results for the whole dataset in Table 1. The non-extrapolating results are better than the extrapolation model results. This is to be expected, as they have the additional training data for the rare terms. It in interesting that most of the non-extrapolating results for the extrapolation sub-dataset are better than for the overall dataset in Table 1. it can be seen in particular that the non-extrapolating GRU models perform better than the non-extrapolating Baseline models. This suggests that they are successfully able to transfer the knowledge of the tokens used in other contexts, to the rare uses in the extrapolation sub-dataset, and also while benefiting from the small number of examples of use in the fine-tuning using the small number training case for the rare descriptions. The baseline model is unable to do this, relying only on the small number of training cases for that exact color description. This in-particular shows for the Baseline with output resolution of 256. Given the high number of output bins, and the small number of training case, many bins would be empty. The GRU model can use its knowledge of the terms in other uses to predict the distribution for bins not seen in training.

11 Conclusion

We have presented a method for estimating the probably distribution of colors that may be ascribed an input name. This methods uses a discretization process based on treating each training point as the center of a Gaussian, or wrap-around Gaussian distribution, and finding the probability distribution for discrete regions of the color-space. The blurring in the discretized training points helps the model's softmax output to learn a reasonable continuous probability distribution, as approximated using a discrete distribution. Working with probability distributions, rather than regression to a single color-space point on color, allows for better handling of colors with observed distributions that are asymmetric, wide variance or multimodal in the color-space – most colors.

The model learns the compositional structure of a color name, which it is able to use to predict distributions for colors not seen given during training. The input terms learn separate representations, which are together used to estimate the distribution. For example: the color "dirty brown" does not occur in the training data, but there are many used of "dirt", the suffix "y" and "brown" in other combinations. So the GRU model can estimate a distribution.

11.1 Future-work

The discretization process representing a continuous probability distribution as a discrete distribution is pragmatically effective, but unsatisfying. We suggest there are avenues for advancement here by the extension of Magdon-Ismail and Atiya (1998) to handle conditional distributions.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. 2014. Julia: A fresh approach to numerical computing .

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

George E Dahl, Tara N Sainath, and Geoffrey E Hinton. 2013. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pages 8609–8613.

Richard S Hunter. 1958. Photoelectric color difference meter. *Josa* 48(12):985–995.

Kazuya Kawakami, Chris Dyer, Bryan R. Routledge, and Noah A. Smith. 2016. Character sequence models for colorfulwords. *CoRR* abs/1609.08777.

Malik Magdon-Ismail and Amir Atiya. 1998. Neural networks for density estimation. In *NIPS*. pages 522–528.

E. Mansimov, E. Parisotto, J. Lei Ba, and R. Salakhutdinov. 2015. Generating Images from Captions with Attention. *ArXiv e-prints* .

Brian McMahan and Matthew Stone. 2015. A bayesian model of grounded color semantics. *Transactions of the Association for Computational Linguistics* 3:103–115.

T. Meo, B. McMahan, and M. Stone. 2014. Generating and resolving vague color reference. *Proc. 18th Workshop Semantics and Pragmatics of Dialogue (SemDial)* .

W. Monroe, N. D. Goodman, and C. Potts. 2016. Learning to Generate Compositional Color Descriptions. *ArXiv e-prints* .

Will Monroe, Robert X. D. Hawkins, Noah D. Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *CoRR* abs/1703.10186.

Randall Munroe. 2010. Xkcd: Color survey results. Website.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*. volume 3.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.