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~~ discretization 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 an actual feature of the population ~~’s varying conception featuring widevariance, and a potentially multimodal~~ speaker varying conception of the color. The distributions feature wide, asymmetrical variance; and are potentially multimodal in nature. We demonstrate a GRU-based neural network learning the grounded compositional semantics of the terms used ~~with in a~~ within 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. Our results show that it is reliably able to do this, with only a small decrease in accuracy, compared to model trained directly with those combinations of terms. 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”

(??). It is a challenging domain, due to ambiguity, multiple roles taken by the same words, many modifiers, and shades of meaning. Understanding color is crucial to understanding language.

The mapping from color name to color could be considered a regression problem~~. Solving~~ , 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 monomodal [1] and symmetric with consistently small variance, then considering the problem as regression with noise would be adequate. ~~If the distribution were~~ However, the distributions in color space are generally non-systematic, wide variance, and are often 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~~ . Thus regression to a single value is inadequate.

Rather than attempting to predict a single point in color space, an estimate of the distribution in color space must be produced. To do this we divide the color space into a number of even sized regions, i.e. bins, and then discretize color observations from the training data into thethe probability distribution into these bins. This transforms the problem of estimating a distribution in color-space, into the very well understood problem of classification into categories – where each category is a region of color-space.

A classifier will output probabilities for each of the possible categories ~~an~~ its 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 distributionin color-space~~, thus fulfilling the goal of learning a probability distribution. Basic classification ~~models~~ model 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~~ In discretization the notion of continuousness ~~. A variety of approaches called ordinal regression or ordinal classification exist to handle thiscase, 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.~~ is lost. To counter this, we employ a novel blurring strategy, to train the model to learn that points near in color space, should be similar in likelihood.

~~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~~Understanding color distributions has a clear use as a subsystem many systems. For example, ~~as a subsystem in human interfacing image processing~~in a human-interfacing image-processing system, 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.

2 ~~Highly multimodal colors~~

~~One of the core motivating factors~~ The set of all possible color descriptions form a much large space of multiword descriptions than can reasonably be collected in detail. Possible

The core contribution of this work is ~~to be able to handle the~~ a method for estimating probability distributions in color-space given a color names which ~~have multiple distinct modes. That is to say there are distinct peaks of the most likely region in color space. ? identify “greenish” as a convex color , i.e. one with a multimodal distribution. We further identify several others “purplish grey”,~~

[1] It should be understood that in this paper, when say *monomodal* 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. ? call this convex, we prefer the term multimodal.

~~“purplish” and “blueish” amongst them to varying extents. Though this is not true for all “-ish” colors: “reddish”, “orangish”, “yellowish” are not. 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 in it can be seen that that a the hue distribution significantly favors blues and reds over greens and purples. Further, “light grey” increases the yellow peak (), and “dark grey” () 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 .~~

~~A core interest here is in colors with a multi-modal 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.~~ is able to handle color descriptions that are never seen during training. To handle distribution estimation we employ a novel discretization and blurring procedure. To allow for the capacity to predict distributions for colors never seen in training we define a GRU-based neural network which learns the compositional relationship expressed by the sequence terms making up the color description. We call this model the Color Distribution Estimation from Sequence of Terms (CDEST) model.

2 Related Work

2.1 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. **?** and **?** present a full description Bayesian approach, which outputs the probability of a whole description. **?** presents a per-word LSTM approach, which produces a conditional language model – sequentially outputting a probability of each word in the description. **?** 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.

2.2 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. **?** 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 (**?**). 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 distribution 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.

3 ~~Color Identification~~

2.1 Color Identification

? 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 **?**. 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.

3 ~~Method~~Color Distribution Estimation Framework

4 ~~Blurring and Discretization~~

~~To allow the network to readily learn the distributionof the colors the task is changed from learning a conditional~~

3.1 Discretization and Blurring

The core of this task is to estimate a conditional probability distribution. Estimating conditional discrete probability distributions is a very-well established neural network task. We thus transform the task of learning a continuous distribution to the well-established problem of learning a ~~conditional discrete distribution . We~~ discrete distribution by discretizing the output space. In evaluation we discretize to a resolutions of 64, and to 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 (?). The ~~discretization process is as follows.~~ estimation of a discretized distribution is very similar to outputting a histogram for the distribution. However, we do not discretize the training data sharply, but rather add a blur. The purpose of the blur is to encourage smoothness between adjacent output bins, thus capturing some of the notion of continuousness.

~~First, a blur~~ Blur is added to each observation ~~. This is done by defining a distribution with expected value equal to the observation, and with variance defined~~ in the training set by treating each point as the expected value of a distribution, with the variance given as a hyper-parameter of ~~our process~~training. 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 development 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

[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

hyper parameter. Further tuning of this parameter might enable better results – particularly using different blurring levels for the difference channels.

4 ~~Conditional Independence Assumption~~

3.1 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~~ it is assumed if the color name is known, then knowing the value of one ~~component would not inform us~~ channel would provide any additional information as to the value of the other ~~if we already know the name of the color.~~ channels. This assumption is ~~incorrect, but not as incorrect as might be suspected~~not made in related works such as ??. These works do not attempt to fully model the distribution, and instead make other assumptions – such as assuming symmetrical distributions, identical up to translation. When attempting to fully model the distributions, the assumption of conditional independence allows considerable saving in computational resources. Approximating the 3D joint distribution as the product of three 1D distributions decreases the space complexity from $O(n^3)$ to $O(n)$.

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 – which is implementation wise a trivial extension of the models, though practically the increased memory use renders it computationally infeasible on most hardware.

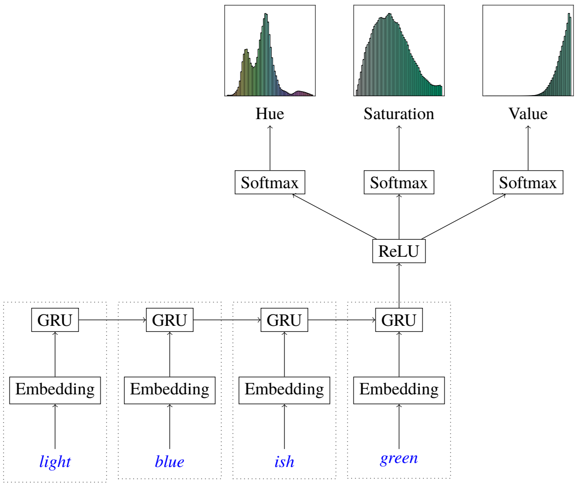

Figure 1: The ~~GRU Model~~ CDEST model for predicting the color-space probability distributions of color. The section in the dotted-boxes is repeated for each time step.

~~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_{res}^3 elements. This saves a large amount of memory during the training.~~

4 ~~The Models~~

3.1 ~~GRU~~ CDEST Model

We present a neural network based model, with sequential inputs which predicts the 3 separate output distributions shown in Figure 1. The general structure of this network is similar to ?, or indeed to most other word sequence learning models. ~~It is shown in .~~ Each word first is transformed to an embedding representation. This representation is randomly initialized and is trained with the rest of the network allowing a per word information to be efficiently learned. The embedding is used as the input for a Gated Recurrent Unit (GRU) (?).

The output of final time-step is feed to a Rectified Linear Unit (ReLU) (?). Finally, the this used as the input the three distinct softmax output layers – one for each of hue, saturation and value. The ~~network was trained to minimize to sum of the three cross-entropy losses of these~~ distinguishing features of this model compared to other word sequence learning models, is the use of GRU, rather than Long Short Term Memory (LSTM), and the split into three output layers. ~~The multiple output layers commonly occur joint learning 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 (?); including on the color naming problem (?). ~~The GRUforms the basis of the our sequence-of-terms to color-space-probability-distribution network~~ A component for processing per-term such as the GRU, is essential in allowing the model to learn the compositional function of each term, and thus to learn to handle color descriptions from outside the training set.

The three output layers are used to predict the distributions for the three channels – hue, saturation and value. Seperating them like this requires a conditional independence assumption (see Section 3.1). The network was trained to minimize to sum of the three cross-entropy losses of these output layers. The multiple output layers commonly occur joint l[illegible]ng and related transfer learning problems. The layers prior to the output are shared, allowing common patterns to be learned. This model is the primary contribution of this work.

3.2 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 Baseline model, the estimated distribution for each color description is produced by averaging all of that colors discretized observations per output bin. This features the same blurring, as in the ~~GRU Model~~CDEST 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~~ CDEST 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.

4 Experimental Setup

4.1 Data Preparation

We use the Monroe dataset (?), as prepared by McMahan and Stone (?). This dataset is partitioned into test, development, and training sets; and has had some cleaning from the original data colle[illegible] by Randell ?. It is also used by Will ? and ?. The color space is HSV, with all values between zero and one. Each is ~~pair with a short name for the color , and~~ paired with one of 829 color description, as 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 tokenizationdescribed above~~Using this tokenization, each description is split into between one and four ~~tokens~~terms. This results in a total of 311 unique tokens used by the CDEST model. The Baseline model does not function per term, so uses the original 829 descriptions directly.

4.2 Extrapolation Sub-Dataset

~~One of the key advantages of our proposed system~~ The key advantage of CDEST over the Baseline model is its ability to predict the distribution for never before ~~scene~~ seen 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 from the dataset, 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.

4.3 ~~GRU~~ CDEST Model Parameters

For the ~~GRU~~ CDEST 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 (**?**) with a probability of 0.5 was used during training, on all hidden layers, except the embedding layer.

4.4 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:

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

As the perplexity varies depending on the output resolution, we will also consider when comparing models of different resolution the standardized value of ~~$\frac{PP_c(\tau)}{n}$~~ $\frac{PP(\tau)}{n}$, 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 ~~$\frac{PP_c(\tau)}{n} = 1.0$~~$\frac{PP(\tau)}{n} = 1.0$. 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 mono-modal symmetric case, and allows ~~out~~ our model to be compared to regression models. To do this, the output bin with the ~~highest~~ maximum 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 is selected. The mean square error is found in the transitional way, averaging over the three channels. That is to say, with the definitions as before:

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

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

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

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

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 chosenin a greedy way~~greedily chosen.

4.5 Implementation

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

5 Results and Discussion

5.1 Qualitative Comparison of the Distribution

Shown in ~~, , , , and earlier in , ,~~ Figures 2 to 4 are side by-side comparisons of the output of the ~~64 bin GRU model, compared to the Baseline model~~CDEST and the Baseline models. Overall, it can be seen the Baseline model is a lot sharper, with more spikes, where as the ~~GRU~~ CDEST 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 , , , that small close peaks tend to be smoothed between. Where-as more separated peaks, as in~~ Smoothness is generally desirable, the ~~hue component in , , 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 ?. Also like in their work, this filling in results in~~ increased smoothness particularly as seen in the saturation

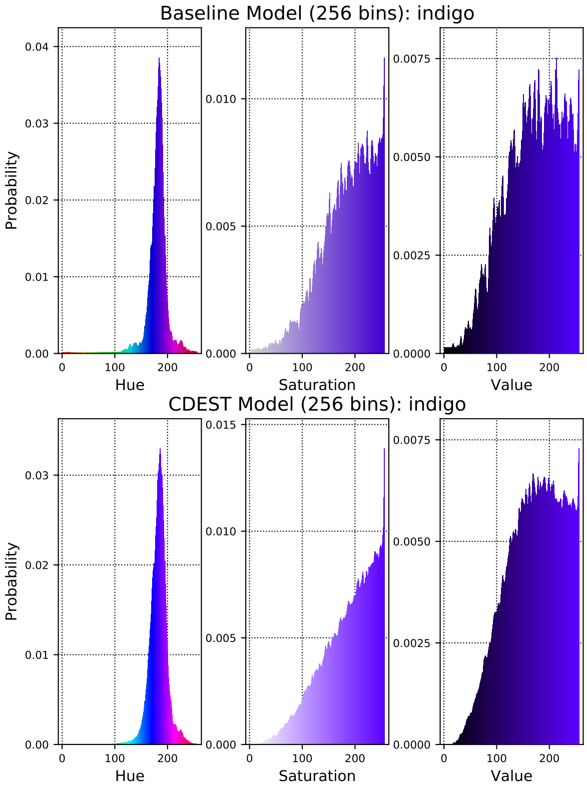

Figure 2: indigo

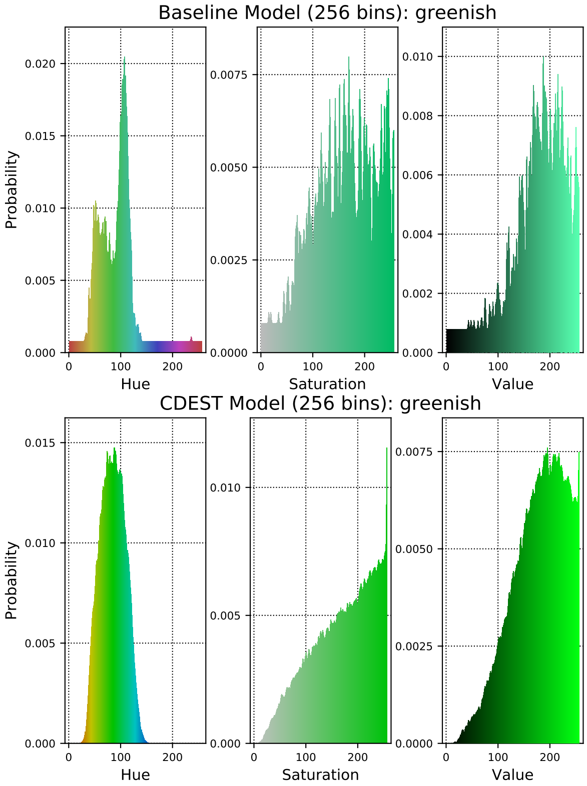

Figure 3: greenish

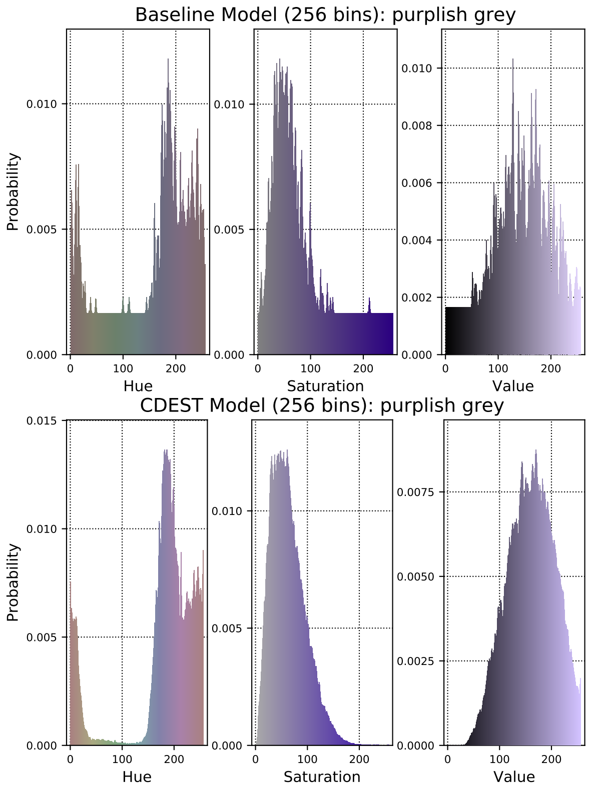

Figure 4: purplishgrey

and value channels, is intuitively correct. Though not always.

It can be seen that the CDEST model fails for some multimodal colors – such as the hue "greenish" ~~being filled in~~(Figure 3) where the concave section is filled in; but succeeds for others such as "purplish grey" (Figure 4). We suggest the reason for this may be difficulties caused by the use of greenish both as a modifier: "greenish blue" and as a standalone description covering the edges of the green band.

~~The distribution output by GRU model with resolution 256, and blurring $\sigma = \frac{1}{8n}$ 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 . However, this comes at the cost of overall worse performance, as~~ The horizontal bands in 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~~ Baseline models are the result of the add-one smoothing process, they are larger for colors with fewer examples – 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~~purplish grey". In the seminal work of **?** one of the motivations for employing neural networks in natural language processing was to better handle cases that do not occur in the training data, by sharing information between terms. While **?** was looking at language modeling, where the key technique for handling unseen cases was back-off, the case equally applies here for distribution estimation, where unseen cases are handled with add-one smoothing. The neural model of CDEST can, by knowledge sharing, better estimate the values for the unseen cases in color space.

5.2 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~~ confirms that the CDEST model is fitting correctly. The ~~GRU~~ CDEST 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 the 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~~ CDEST 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~~ CDEST model is its ability to predict a distribution for an unseen combination of colors, this is evaluated using the extrapolation task.

model	n	PP	MSE	PP_S	PP_H	PP_V	$\frac{PP}{n}$
CDEST	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
CDEST	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 CDEST	64	27.35	0.1774	41.4	15.84	31.19	0.4273
Non-extrapolating CDEST	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 CDEST	256	108.8	0.2072	165.3	62.31	125	0.425
Non-extrapolating CDEST	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~~extrapolation sub-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.

5.3 Extrapolation

A core motivation of using the ~~GRU mode~~CDEST model, over the Baseline, is its ability to learn to combine tokens in a description in ways not seen in training. The best the baseline model can do on extrapolation is a uniform distribution – $\frac{PP}{n} = 1.0$. To evaluate how well the model does at predicting these distributions, we compare a ~~GRU~~ CDEST 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~~ color distribution in the other cases.

The results for ~~estimating the distributions for the rare color descriptions in the extrapolation sub-dataset~~ this evaluation 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 CDEST results are better than the extrapolation model results. This is ~~to be expected, as they have the additional~~ as expected since the non-extrapolating models have training data for the rare ~~terms. It in interesting that most of the~~ color descriptions that occur in the extrapolation test set.

The non-extrapolating ~~results for the extrapolation sub-dataset are better than for the overall dataset in . it~~ CDEST also befitting from the same knowledge sharing that allows extrapolating CDEST model to function at all. This 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~~ from CDEST model out performing the Baseline model. The baseline model can not benefit from the knowledge sharing based on term use for estimating the curve of 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~~This is to the extend that in the high resolution case (256 bin), the sparsity of training data is such that the extrapolating CDEST model out performs the non-extrapolating Baseline.

6 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~~ CDEST model can estimate a distribution.

6.1 Future-work

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 handling of 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.

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 **?** to handle conditional distributions.