Estimating the Intended Color from the Terms that make up it's Name

Lyndon White, Roberto Togneri, Wei Liu,
Mohammed Bennamoun
lyndon.white@research.uwa.edu.au, roberto.togneri@uwa.edu.au,
wei.liu@uwa.edu.au,

mohammed.bennamoun@uwa.edu.au
The University of Western Australia. 35 Stirling Highway, Crawley, Western Australia

2018-06-22

Abstract

When a speaker says the name of a color, the color that they picture is not necessarily the same as the listener imagines.

Further we argue that color understanding is a grounded semantic task, but that grounding is not a trivial mapping of a single word or phrase to a single point in color-space as there are many different interpretations of what color is mean by a particular color name. Proper understanding of color language requires the capacity to map a sequence of words to a probability distribution in color-space. As such we focus our investigation on distribution estimation from color name. A distribution is required as there is no clear agreement between people as to what a particular color describes – different people have a different idea of what it means to be "very dark orange". To access if our conclusions about our methods hold more generally, we also assess them on the task of point estimation of color names. We find the results are consistent, though we argue this task is less useful.

Color names are often made up of multiple words, as a task in natural language understanding we investigate in depth the capacity of neural networks based on sums of word embeddings (SOWE), recurrence (RNN) and convolution (CNN), on their ability to make estimates of colors from sequences of terms. We contrast their performance to direct non-term based methods which form a rough upper-bound by directly using the training data. Surprisingly, the sum of word embeddings generally performs the best, coming close to this upper bound on almost all evaluations. It is trailed slightly by the CNN, and most substantially by the RNN.

1 Introduction

Color understanding is an important subtask in natural language understanding. It is a challenging domain, due to ambiguity, multiple roles taken by the same words, the many modifiers, and shades of meaning. In many ways it is a grounded microcosm of natural language understanding. Due to its difficulty, texts containing color descriptions such as `the flower has petals that are bright pinkish purple with white stigma` are used as demonstrations for state of the art image generation systems (**??**). The core focus of the work we present here is addressing these linguistic phenomena around the short-phrase descriptions of the color, in a single patch, as represented in a color-space such as HSV (**?**). Issues of illumination and perceived color based on context are considered out of the scope. We evaluated a variety of neural-network based systems, for their capacity to be used for this natural language understanding task. We believe our results results are suggestive to the capacity of these systems for other similar tasks involving short token sequences.

Consider that the word `tan` may mean one of many colors for different people in different circumstances: ranging from the bronze of a tanned sunbather, to the brown of tanned leather; `green` may mean anything from `aquamarine` to `forest green`; and even `forest green` may mean the rich shades of a rain-forest, or the near grey of the Australian bush. Thus the color intended cannot be uniquely inferred from the color name. Without further context, it does nevertheless remain possible to estimate likelihoods of which colors are be intended based on the population's use of the words. The primary aim of this work is to map a sequence of color description words to a probability distribution over a color-space. This is required for a proper understanding of color language. We also consider the more basic point estimation of colors, though we question its value.

Proper understanding requires considering *the color intended* as a random variable. In other words, a color name should map to a distribution, not just a single point or region. For a given color name, any number of points in the color-space could be intended, with some being more or less likely than others. Or equivalently, up to interpretation, it may intend a region but the likelihood of what points are covered is variable and uncertain. This distribution is often multimodal and has high and asymmetrical variance, which further renders regression to a single point unsuitable. We do produce results point estimate results for interest in **??**, however for any form of precise work the use of such systems is limited A single point estimate, does not capture the nature of the problem adequately. The mean of a multimodal distribution (one with two peaks) will lie in the valley between the – a less likely color. Similarly it will be off to the side of the mode, in an asymmetrical distribution. The other problem is that is that a point estimate does not capture the sensitivity. In an asymmetrical distribution, having point

slight off-center in one direction may result in very different probability, this more generally holds for a narrow variance distribution. Conversely for a very wide variance distribution (for example one approaching the uniform distribution) the point estimate value may matter very little with all points providing similar probabilities. Color distributions are always one or more of multimodal or asymmetrical, and feature widely different variances for different names. As such, we feel producing a point estimate has limited value. However we do consider the task, for comparison, and it remains an interesting challenge for assessing the systems under evaluation. While by definition the mean color point minimizes the squared error, it may not meaningful actually be a reasonable point estimate. Better is to estimate the distribution and allow the consumer of the estimate to determine how to reduce that to a point estimate.

When we estimate a probability distribution over the color-space. To qualify our estimate of the distribution we discretize the space to produce into a histogram. This allows us to take advantage of the well-known softmax based methods for estimating a probability mass distribution using a neural network.

An interesting consideration when considering this discretization is the smoothness estimate. The true space is continuous, even if we are discretizing it at resolution as high as the original color displays used to collect the data. Being continuous means that a small change in point the color-space should correspond to a small change in how likely that point is. Or more informally: histograms should look smooth, and not spiky. We investigate using a KDE based method for smoothing the training data, and further we conclude that the neural network based models are learning this smoothness with or without the smoothing of the training data.

Estimating color probabilities has a clear use as a subsystem in many systems. For example, in a human-interfacing system, when asked to select the `dark bluish green` object, each object can be ranked based on how likely its 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, if the probability of the color of the object being described by the user input is known, a threshold can be set to report that no object is found, or to ask for additional information. More generally, the distribution based on the color name alone can be used as a prior probability and combined with additional context information to yield better predictions.

1.1 Contributions

Our contributions in this work are (in order of significance)

- Investigation of term-based neural network color estimation models based on:

3

- processing inputs per term using Sum of Word Embeddings (SOWE), Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

- producing outputs as distributions models using softmax based histograms, and as point estimates using a novel HSV neural network output layer.

- we conclude that while all these models learn useful outputs, the SOWE and the CNN far out perform the RNN.

- Investigation into the effect of smoothing the distribution training histograms, using a kernel density based (KDE) method; featuring wrapped around effects on the hue channel. We find the interesting conclusion that while smoothing is essential for direct methods of estimation, it has little effect on the machine learning based methods that we investigate.

The investigations include investigating the estimation capacity of the models:

- generally on all color names (full task)

- for color names when the order of the words matters (order task)

- for color names which never occur in the training data exactly, but for which all terms occur in the training data (extrapolation task

- for color names with terms that do not occur in the training data at all, but for which we know word embeddings for (embedding only task).

The last investigation is not carried out in depth, but is presented merely as a point of interest. We believe that due to the nature of color understanding as a microcosm of natural language understanding, the results of our investigations have some implications for the capacity of the models for their general use in short phrase understanding.

2 Related Work

The understanding of color names has long been a concern of psycholinguistics and anthropology (**????**). It is thus no surprise that there should be a corresponds field of research in natural language processing.

The earliest works revolve around explicit color dictionaries. This includes the ISCC-NBS color system (**?**) of 26 words, including modifiers, that are composed according to a context free grammar such that phrases are mapped to single points in the color-space; and the simpler, non-compositional, 11 basic colors of **?**. Works including **??????** which propose methods for

4

the automatic mapping of colors to and from these small manually defined sets of colors. We note that **??** both propose systems that discretize the color-space, though to a much courser level than we consider in this work.

More recent works, including the work presented here, function with much larger number of colors, larger vocabularies, and larger pools of respondents. In particular making uses of the large Munroe dataset **?**, as we do here. This allows a data driven approach towards the modelling.

? and **?** present color naming methods, mapping from colors to to their names, the reverse of our task. These works are based on defining fuzzy rectangular distributions in the color-space to cover the distribution estimated from the data, which are used in a Bayesian system to non-compositionally determine the color name. **?** maps a point in the color-space, to a sequence of probability estimates over color terms. They extends beyond, all prior color naming systems to produce a compositional color namer based on the Munroe dataset. Their method uses a recurrent neural network (RNN), which takes as input a color-space point, and the previous output word, and gives a probability of the next word to be output – this is a conditional language model. In this work we tackle the inverse problem to the creation of a conditional language model. Our distribution estimation models map from a sequence of terms, to distribution in color space. Similarly, our point estimation models map from sequence of terms to single point in color-space.

? propose another compositional color naming model. They use a per-character RNN and a variational autoencoder approach. It is in principle very similar to **?**, but functioning on a character, rather than a word level. The work by Kawakami et al. also includes a method for generating colors. However they only consider the generation of point estimated, rather than distributions. The primary focus of our work is on generating distributions. The datasets used by Kawakami et al. contain only very small numbers of observations for each color name (often just one). These datasets are thus not suitable for modelling the distribution in color space as interpreted by the population. Further, given the very small number of examples they are not well suited for use with word-based modelling: the character based modelling employed by Kawakami et al. is much more suitable. As such we do not attempt comparison to their work.

? presents a neural network solution to a communication game, where a speaker is presented with three colors and asked to describe one of them, and the listener is to work out which is being described. Speaker and listener models are trained, using LSTM-based decoders and encoders respectively. The final time-step of their model produces a 100 dimensional representation of the description provided. From this, a Gaussian distributed score function is calculated, over a high dimensional color-space from **?**, which is then used to score each of the three options. While this method does work with a probability distribution, as a step in its goal, this distribution is always both symmetric and unimodal – albeit in a high-dimensional color-space.

The generation of color from text has not received a signficant ammount of attention in prior work. In particular the generation of probability distributions in color space, to our knowledge has not been considered at all. Conversely, there has been several works on the reverse problem: the generation of a textual name for a color from color space point. The work presented here closed that gap.

3 Method

3.1 HSV color-space

We use the HSV color-space (**?**). through-out this work. We use the format which the data is originally provided in. In this format hue, saturation and value all range between zero and one. Note that in this dataset hue is measured in *turns*, rather than the more traditional degrees, or radians. Having hue be measured between zero and one (like the other channels) makes the modelling task more consistent as were the hue to range between 0 and 2π (radians) or between 0 and 360 (degrees) it would be over-weighted compared to the other channels. This regular space means that errors on all channels can be considered equally. Unlike many other colors spaces (CIELab, Luv etc.) the gamut is square and all values of all channels corresponds to realizable colors.

When working with the hue, all measures need to take into account the wrap-around effect. When ever we refer to mean squared error, mean or mode on the HSV space in this paper, we are referring to the angularly corrected forms given in Section 3.2.

Unlike the RGB color space, the HSV channels do correspond to how humans perceive colors. However, it is not designed to be a perceptionally uniform color space across hue (unlike CIELab), which does suggest using mean squared error for the point estimates is not optimal. However, given the other issues outlined above with point estimates we do not judge this a major concern. There are no such issues for our distribution estimations.

One of it's important advantages over other color spaces is that it best meets the assumption that for a given color name each channel is statistically independent.

3.2 Angularly Correct Calculations on HSV

When performing calculations with the HSV color space it is important to take into account that hue is an angle. As we are working with the color space regularized to range between zero and one for all channels, this means a hue of one and a hue of zero are equivalent (as we measure in turns, in radian's this would be 0 and 2π).

The square error of two hue values is thus calculated as

$$SE(h_1, h_2) = \min \left((h_1 - h_2)^2, \ (h_1 - h_2 - 1)^2 \right) \quad (1)$$

to take into account the error could be calculated clockwise or counter clockwise. (Note that the -1 term related to using units of turns, were we using radians it would be -2π)

The mean of a set of hues ($\{h_1, \ldots, h_N\}$) is calculated as

$$\bar{h} = \mathrm{atan2}\left(\frac{1}{N} \sum_{i=1}^{i=N} \sin(h_i), \ \frac{1}{N} \sum_{i=1}^{i=N} \cos(h_i) \right) \quad (2)$$

(Note again: as we measure angle in turns we use the turn trigonometric functions in implementation, though this mean is the same expression or any units).

4 Distribution Estimation

4.1 Conditional Independence Assumption

We make the assumption that given the name of the color, the distribution of the H, S and V channels are independent. That is to say, it is assumed if the color name is known, then knowing the value of one channel would not provide any additional information as to the value of the other two channels. The same assumption is made, though not remarked upon, in **?** and **?**. This 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 $O(n)$ in the discretized step that follows.

Superficial checks were carried out on the accuracy of this assumption. Spearman's correlation on the training data suggests that for over three quarters of all color names, there is only weak correlation between the channels (Q3 = 0.187). However, this measure underestimates correlation for values that have circular relative value, such as hue. HSV had the lowest correlation by a large margin of the 16 color-spaces evaluated. Full details, including the table of correlations, are available in supplementary materials. These results are suggestive, rather than solidly indicative, on the degree of correctness of the conditional independence assumption. We consider the assumption sufficient for this investigation.

4.2 Discretization

For distribution estimation, our models are trained to output histograms. This is a discretized representation of the continuous distribution. Following standard practice interpolation-based methods can be used to handle it as

a continuous distribution. By making use of the conditional independence assumption (see Section 4.1), we output one 256-bin histogram per channel. We note as 24-bit color (as was used in the survey that collected the dataset) can have all information captured by a 256 bin discretization per channel. 24 bit color allows for a total of 2^{24} colors to be represented, and even onehot encoding for each of the 256 bin discretization channels allows for the same. As such there is no meaningful loss of information when using histograms over a truely continuous estimation method, such as a Gaussian mixture model. Although such models may have other advantages (such as the a priori information added by specifying the distribution), we do not investigating them here, instead considering the simplest non-parametric estimation model (the histogram), which has the simple implementation in a neural network using a softmax output layer.

Discretizing the data in this way is is a useful solution used in several other machine learning systems. **??** apply a similar discretization step and found their method to outperforming the more complex continuous distribution outputs.

For training purposes we thus need to convert all the observations into histograms. One set of histograms is produced per color description present in the dataset. We perform an uniform weight attribution of points to bins as described by **?**. This method of tabulation is in-short to define the bins by there midpoints, and for a point observed between the centre of two bins then probability mass is allocated to each in proportion to how close the point is to the centre of each.

4.3 Kernel-Density Estimation Based Smoothing

As the distribution is not truly discrete, we investigate adding a Kernel-Density Estimation (KDE) based smoothing step. KDE is a common approach in non-parametric estimation.

This is expected as a blurred discrete distribution captures some of the notions of continuity that a truly continuous output distribution would intrinsically feature.

We use the Fast Fourier Transform (FFT) based KDE method of the **?**. We use a Gaussian kernel, and select the bandwidth per color description based on leave-one-out cross validation on the training data. A known issue with the FFT based KDE method is that it has a wrap-around near the boundaries where mass that would be assigned outside the boundaries is instead assigned to the bin on the other side. For the value and saturation channels we follow the standard solution of initially defining additional bins outside the true boundaries, then discard those bins and rescale the probability to one. For the hue channel this wrap-around effect is exactly as desired.

4.4 Perplexity in Color-Space (Evaluation Metric)

Perplexity is a measure of how well the distribution, estimated by the model, matches reality according to 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 estimate. It can loosely be thought of as to how well the model's distribution does in terms of the size of an equivalent uniform distribution. Note that this metrics does not assume conditional independence of the color channels.

Here τ is the test-set made up of pairs consisting of a color name t, and color-space point $\tilde{x}$; and $p(\tilde{x} \mid t)$ the output of the evaluated model. Perplexity is defined:

$$PP(\tau) = \exp_2 \left(\frac{-1}{|\tau|} \sum_{\forall (t,(\tilde{x})) \in \tau} \log_2 p(\tilde{x} \mid t) \right) \tag{3}$$

As the perplexity for a high-resolution discretized model will inherently be very large and difficult to read, we define the standardized perplexity: $\frac{PP(\tau)}{n_{res}}$, where n_{res} is the total number of bins in the discretization scheme. For all the results we present here $n_{res} = 256^3$. This standardized perplexity gives the easily interpretable values *usually* between zero and one. It is equivalent to comparing the relative performance of the model to that of a uniform distribution of the same total resolution. $\frac{PP(\tau)}{n_{res}} = 1$ means the result is equal what we would have to if we had distributed the probability mass uniformly into all bins in a 3D histogram. $\frac{PP(\tau)}{n_{res}} = 0.5$ means the result twice as good as if we were to simply use an uniform distribution: it is equivalent to saying that the correct bin is selected as often as it would be had an uniform distribution with half as many bins (ie larger bins with twice the area) were used. The standardised perplexity is also invariant under different output resolutions. Though for brevity we only present results with 256 bins per channel, our preliminary results for using other resolutions are similar under standardized perplexity.

5 Experimental Setup

5.1 Implementation

The implementation of the all models was in the Julia programming language (**?**). The full implementation is included in the supplementary materials. can be downloaded from the GitHub repository.[1] The machine learning components make heavy use of the MLDataUtils.jl[2] and TensorFlow.jl,[3]

---

[1]Implementation source is at `https://github.com/oxinabox/ColoringNames.jl`

[2]MLDataUtils.jl is available from `https://github.com/JuliaML/MLDataUtils.jl`

[3]TensorFlow.jl is available from `https://github.com/malmaud/TensorFlow.jl`

packages. The latter of which we enhanced significantly to allow for this work to be carried out. The discretization and blurring process is done using KernalDensityEstimation.jl.[4]

5.2 Datasets

5.2.1 Full Training and Testing set

We make use of the Munroe dataset as prepared by **?** from the results of the XKCD color survey. The XKCD color survey (**?**), collected over 3.4 million observations from over 222,500 respondents. McMahan and Stone take a subset from Munroe's full survey, by restricting it to the responses from native English speakers, and removing very rare color names with less than 100 uses. This gives a total of 2,176,417 observations and 829 color names. They also define a standard test, development and train split.

5.2.2 Extrapolation Training and Testing Set

The primary goal in constructing using the term based models is to be able to make predictions for never before seen descriptions of colors. For example, based on the learned understanding of `salmon` and of `bright`, from examples like `bright green` and `bright red`, we wish for the systems to make predictions about `bright salmon`, even though that description never occurs in the training data. To evaluate this generalisation capacity, we define an extrapolation sub-dataset for both testing and training. This is defined by selecting the rarest 100 color descriptions from the full dataset, with the restriction that every token in a selected description must still have at least 8 uses in other descriptions in the training set. The selected examples include multi-token descriptions such as: `bright yellow green` and also single tokens that occur more commonly as modifiers than as stand-alone descriptions such as `pale`.

The extrapolation training set is made up of the data from the full training set, excluding those corresponding to the rare descriptions. Similar is done for the development set, so as no direct knowledge of the combined terms can leak during early-stopping. Conversely, the extrapolation test set is made up of only the observations from the full test set that do use those rare descriptions.

By training on the extrapolation training set and testing on the extrapolation test set, we can assess the capacity of the models to make predictions for color descriptions not seen in training (extrapolating result). A similar approach was used in **?**. We contrast this to the same models when trained on the full training set, but tested on the extrapolation test set, to see how much accuracy was lost (non-extrapolating result).

---

[4]KernalDensityEstimation.jl is available from `https://github.com/JuliaStats/KernelDensity.jl`

5.2.3   Order Testing set

It is known that the order of words in a color description to some extent matters. `greenish brown` and `brownish green` are distinct, if similar, colors. To assess the models on there ability to make predictions when order matters we construct the order testset. This is a subset of the full test set containing only descriptions with terms that occur in multiple different orders. There are 76 such descriptions in the full dataset. Each of which has exactly one alternate ordering. This is unsurprising as while color descriptions may have more than 2 terms, normally one of the terms is a joining token such as `ish` or `-`. We only constructed a order testing set: not a corresponding training set, as this is an evaluation using the model trained on the full training data.

5.3   Common Network Features

Dropout(**?**) is used on all layers, other than the embedding layer, with threshold of 0.5 during training. The network is optimized using Adam **?**, using a learning rate of 0.001. Early stopping is checked every 10 epochs using the development dataset. Distribution estimation methods are trained using full batch (where each observation is a distribution) for every epoch. Point Estimation trains using randomized mini-batches of size 2^16 observations (which are each color-space triples). All hidden-layers, except as otherwise precluded (in side the convolution, and in the penultimate layer of the point estimation networks) have the same width 300, as does the embedding layer.

5.3.1   Embeddings

All our neural network based solutions incorporate an embedding layer. This embedding layer maps from tokenized words to vectors. We make use of 300d pretrained FastText embeddings **?**[5].

The embeddings are not trained during the task, but are kept fixed. As per the universal approximation theorem (**??**) the layers above allow for arbitrary non-linear continuous transformation. By fixing the embeddings, and learning this transformation, we can produce estimates of colors from their embedding alone – without any training data at all. This is shown in Section 6.2.

5.3.2   Tokenization

For all the term based methods, we perform tokenization. During tokenization a color name is split into terms, with consistent spelling. For example, `bluish kahki` would become the sequence of 3 tokens: [`blue`, `ish`, `khaki`].

---

[5] Available from `https://fasttext.cc/docs/en/english-vectors.html`

Other than spelling, the tokenization results in the splitting of affixes and combine tokens. Combining tokens and related affixes affect how multiple colors can be combined. The full list of tokenization rules can be found in the accompanying source code. Some further examples showing how combining tokens and affixes are used and tokenized:

- `blue purple` $\mapsto$ [`blue`, `purple`].

- `blue-purple` $\mapsto$ [`blue`, `-`, `purple`].

- `bluish purple` $\mapsto$ [`blue`, `ish`, `purple`]

- `bluy purple` $\mapsto$ [`blue`, `y`, `purple`]

- `blurple` $\mapsto$ [`blue-purple`]

The final example of `blurple` is a special-case. It is the only portmanteau in the dataset, and we do not have a clear way to tokenize it into a series of terms which occur in our pretrained embedding's vocabulary (see Section 5.3.1). The portmanteau `blurple` is not in common use in any training set used for creating word embeddings, so no pretrained embedding is available. As such we handle it by treating it as the single token `blue-purple` for purposes of finding an embedding. There are many similar hyphenated tokens in the pretrained embeddings vocabulary, however (with that exception) we do not use them as it reduced the sequential modelling task to the point of being uninteresting.

5.4   Input Modules

??

5.4.1   Recurrent Neural Network(RNN)

A Recurrent Neural Network is a common choice for this kind of task, due to the variable length of the input. The general structure of this network, shown in Figure 1 is similar to **?**, or indeed to most other word sequence learning models. Each word is first transformed to an embedding representation. This representation is trained with the rest of the network allowing per word information to be efficiently learned. The embedding is used as the input for a Gated Recurrent Unit (GRU) The stream was terminated with an End of Stream (`<EOS>`) pseudo-token, as represented using a zero-vector. The output of the last time-step is fed to a Rectified Linear Unit (ReLU).

We make use of a GRU (**?**), which we found to marginally out-perform the basic RNN in preliminary testing. The small improvement is unsurprising, as the color names have at most 5 terms, so longer short term memory is not required.
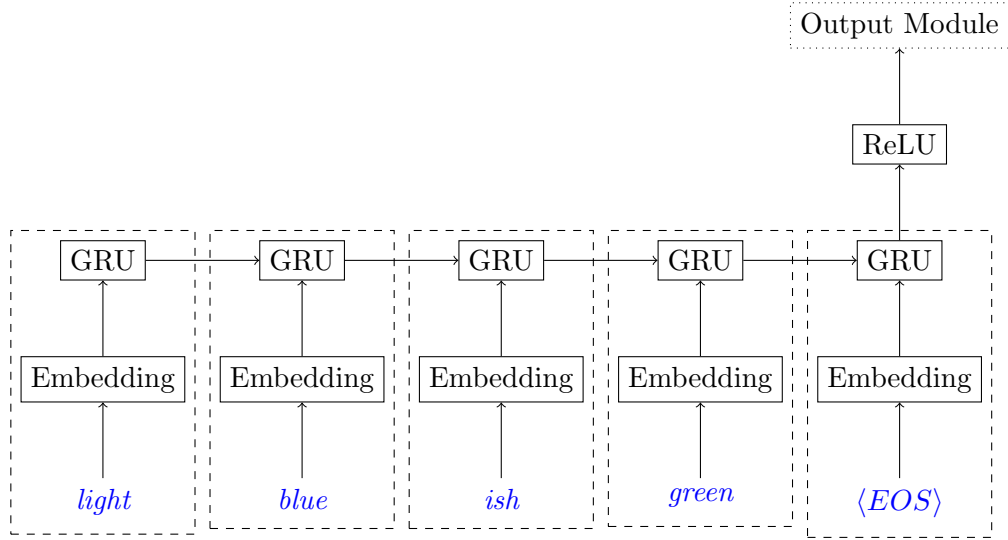
Figure 1: The RNN Input module for the example input `light greenish blue`. Each dashed box represents 1 time-step.

5.4.2 Sum of Word Embeddings (SOWE)

Using a simple sum of word embeddings as a layer in a neural network is less typical than an RNN structure. Though it is well established as a useful representation, and has been used an input to other classifiers such as support vector machines. Any number of word embeddings can be added to the sum. However, it has no representation of the order. The structure we used is shown in Figure 2.

5.4.3 Convolutional Neural Network(CNN)

We apply a convolutional neural network to the task by applying 2D convolution over the stacked word embeddings. Figure 3 We use 64 filters of size between 1 and the length of the longest padded embedding (5).

5.5 Non-term based Direct Methods

To establish an rough upper-limit on the modeling results we evaluate simple direct methods. These direct methods do not process term-by-term, they do not work with the language at all. They simply map from the exact input text (no tokenization) to the precalculated histogram or mean of the training data for the exact color name. It is direct in the sense that it bypassing the compositional language understanding part of the process. It is as if the input module (**??**) perfectly resolved the sequence of terms into a single item.
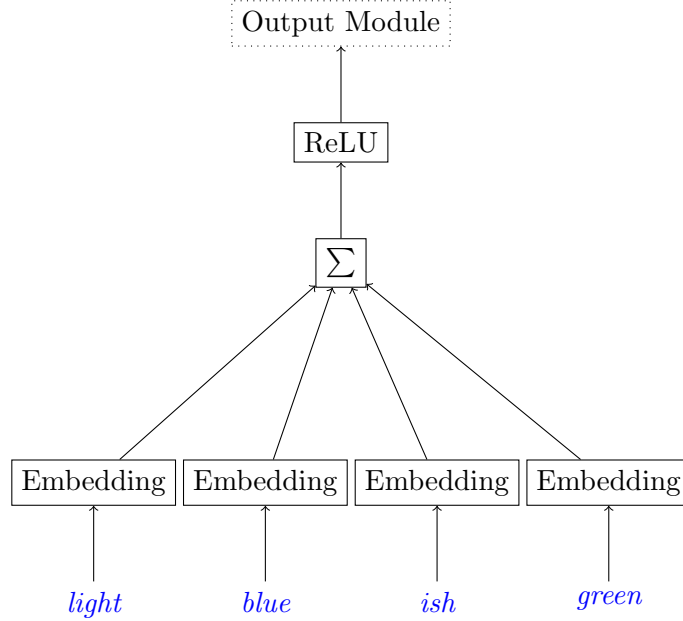
Figure 2: The SOWE input module for the example input `light greenish blue`

They represent an approximate upper bound, as they fully exploit all information in the training data for each input. There is no attempting to determine how each term affects the result. We say approximate upperbound, as it is not strictly impossible that the ML methods may happen to model the test data better than can be directly determined by training data as processed per exact color name. A term based model could out perform them, however this would require learning how the terms in the color name combine in a way that exceeds the information present in the training data. It is this capacity of learning how the terms combine that allow for the models to extrapolate the outputs for combinations of terms that never occur in the training data (Section 5.2.2).

5.5.1 Direct Histogram (Distribution Estimation, only)

The Direct Histogram method simply returns the histogram as found for training using the method given by Section 4.2. With smoothing, it returns the smoothed histogram as per Section 4.3. However, with two variation over what is used for the creation of the training data. Firstly, they histograms are calculated using 64 bit IEEE Floating point rather than 32 bit (as the remainder of the models use), this means that when smoothing is applied the estimated nonzero area is wider. Secondly, any empty bins in final histogram are replaced with ones containing an arbitrary small value. We
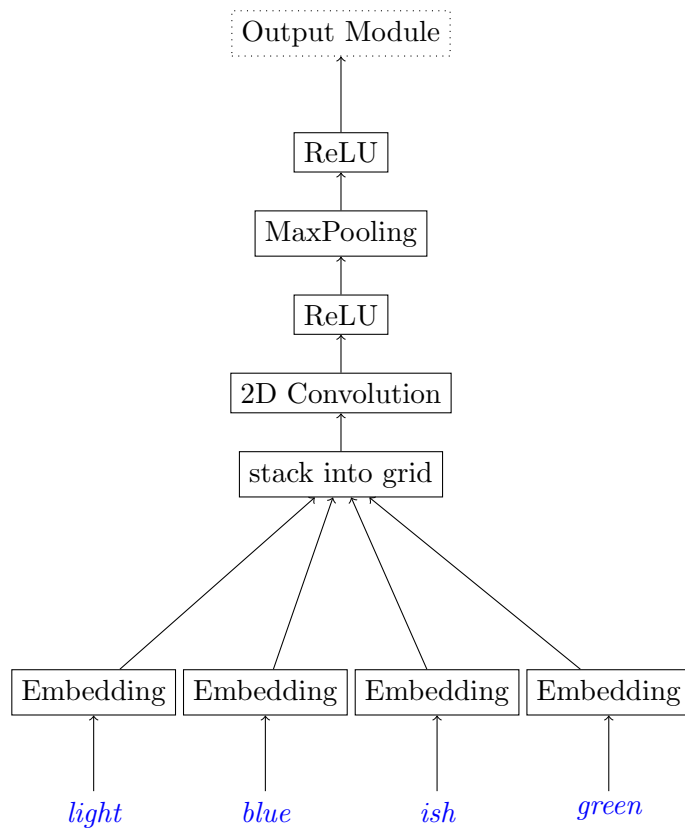
Output Module

ReLU

MaxPooling

ReLU

2D Convolution

stack into grid

Embedding Embedding Embedding Embedding

light   *blue*   *ish*   *green*

Figure 3: The CNN input module for the example input `light greenish blue`

(arbitrarily) use the 64 bit machine epsilon$\approx 2^{-16}$ as this minimum value. While this technically this change would cause the bins to not sum to a total of one, the effect is negligible and is well below the error from using 32 bit math in the other models (which also cause those models not to sum to one). The motivation for both these changes was to ensure the direct models would not return a probability estimate of 0 for any input. Without this the perplexity of any model that incorrectly reported a probability of zero would be undefined (IEEE `+Inf`). We found that capping the minimum value to the machine epsilon the gives much better results than add-one smoothing, (also known as Laplace or Lidstone smoothing) on the development dataset. While this minimum cap penalizes results that would be zero more than add-one smoothing would, it does not reassign probability mass away from results that would be nonzero. We consider giving the direct method this advantage is in-line with determining a rough upper bound on the result.

5.5.2 Direct Mean Point (Point Estimation, only)

In a similar method approach, we also a method that directy produces a point estimate from a color name. We define this by taking the mean of all the training observations for a given exact color name (The mean is taken in the angularly correct way Section 3.2). Taking the mean of all the observations gives optimal solution to minimizing the mean squared error on the training data set. As with our direct distribution estimation method, this bypasses the term based language understanding, and directly exploits the training data. It thus represents an approximate upper bound on the performance of the term based models.

5.6 Output Modules

5.6.1 Distribution Estimation

For all the distribution estimation systems we investigate here, we consider training both on the binned-data, and on the smoothed data (as described in Section 4.3). Making use of the conditional independence assumption (see Section 4.1), we output the three discretized distributions. This is done using 3 softmax output layers.

The output module for distribution estimation

Contrasting to estimating continuous conditional distributions, estimating a discrete conditional distributions is a significantly more studied application of neural networks – this is the basic function of any softmax classifier. To simplify the problem, we therefore transform it to be a discrete distribution estimation task, by discretizing the color-space. Discretization to a resolution of 64 and 256 bins per channel is considered.

For the case of the machine learning models, the output is produced using softmax layers.
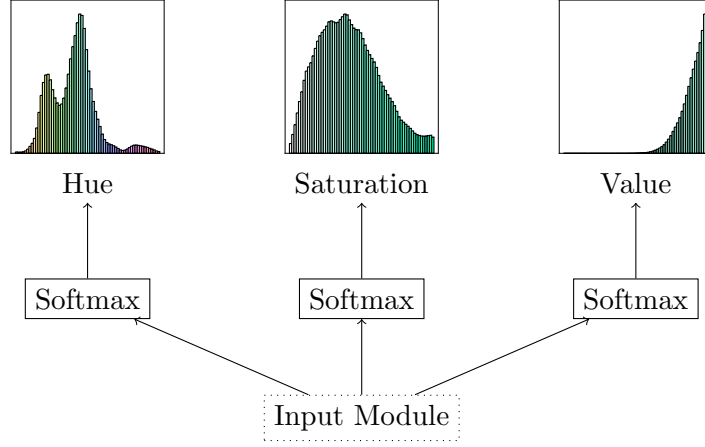
Figure 4: The Distribution Output Module

5.6.2 Point Estimation

Our point estimation output model for the neural network is shown in Figure 5. The hidden-layer from the top of the input module is feed to an 4 single output neurons.[6] Two of these are used the sigmoid activation function (range 0:1) to produce the outputs for the saturation and value channels. The other two use the tanh activation function (range -1:1), they produce the intermediate output that we call y_{shue} and y_{chue} for the sine and cosine of the hue channel respectively. We use these intermediate values when calculated this loss function as it results in a loss function that is continuous and correctly handles the wrap-around nature of the hue channel.

During training we use the following loss function for each observation $y^{\star}$, and each corresponding prediction y.

$$
\begin{aligned}
loss = {} & \frac{1}{2} \left(\sin(y_{hue}^{\star}) - y_{shue} \right)^2 \\
& + \frac{1}{2} \left(\cos(y_{hue}^{\star}) - y_{chue} \right)^2 \\
& + \left(y_{sat}^{\star} - y_{sat} \right)^2 \\
& + \left(y_{val}^{\star} - y_{val} \right)^2
\end{aligned} \tag{4}
$$

This mean of this loss is taken over all observations in each mini-batch during training.

6 Results

---

[6]Equivalently these 4 single neurons can be expressed as a layer with 4 outputs and 2 different activation functions.
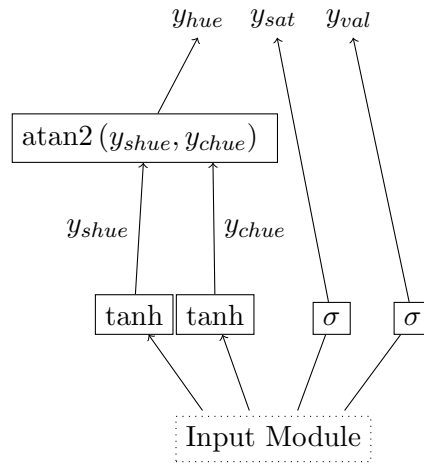
Figure 5: The Point Estimate Output Module. Here atan2 is the quadrant preserving arctangent, outputting as a regularized angle (as per in all evaluations).

;

6.1 Qualitative Results

Figure 6 and Figure 7 show some examples of distribution estimates for the models trained on the full data sets. Figure 8 shows similar examples for point estimates. It can be seen the the model's outputs using term based estimations are generally similar to the direct outputs, as is intended. This aligns with the strong results found in that quantitative evaluations in Section 6.3. The models are correctly fitting to estimate the colors.

When considering the pairs of outputs that differ only in word order, such as `purple-pink` and `pink-purple` the models differ in behaviour. The Direct results show that the ground truth is that such color name pairs are subtly different but very similar. The SOWE models is unable to take into account word order at all, and so produces identical output for both. The CNN models produce very similar outputs but not strictly identical – spotting the different requires very close observation. This is in-line with the different filter sizes allowing it to to use n-gram features, and it finding the unigram features most useful. The RNN produces the most strikingly different results. It seems the first term dominates the final output: `purple-pink` is more purple, and `pink-purple` is more pink. We can see that the first time is not solely responsible for the final output however, as `purple-pink`, `purple` and `purplish` (tokenized as `purple`, `ish`) are all different. It is surprising that the RNN is dominated by the first term and not the latter terms[7] this shows the GRU is functioning to remember the earlier inputs. This is happening too strongly however, as it causes the colors to differ to much.

The effect of the models on the smoothness when estimating distributions is interesting. As expected, applying the KDE based smoothing to the training data produces a smoother output. Further: the all the outputs of the neural network models are smoother than the correspond direct models. This is true both when trained on smoothed and on unsmoothed training data. The models are learning that adjacent bins should have similar output values, as this is a common feature of all the training data no matter the color. An effect of this is that while the models capture the highly asymmetrical shapes of most distributions well, they do not do well at capturing small dips. Larger multi-modes as seen in the achromatic colors like `white`, `grey`, `black`, `white`, are captured; but smaller dips such as the hue of `greenish` being mostlikely to be either side of the green spectrum are largely filled in. In general, it seems clear that the smoothing of the training data is not required for the neural network based models.

---

[7]So much so that we double checked out implementation to be sure it wasn't processing the inputs backwards.
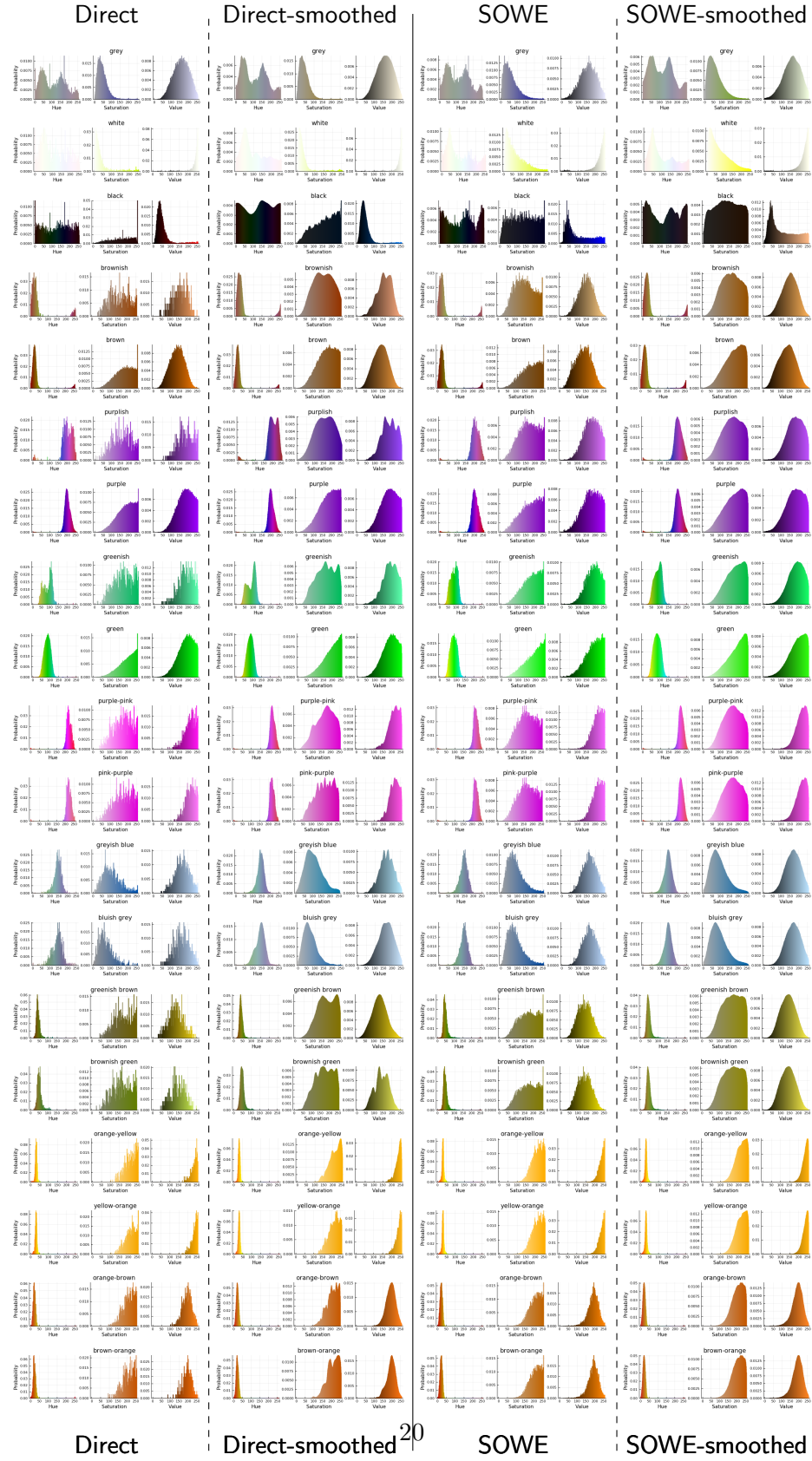
Figure 6: Some examples of the output distribution estimations from the models trained on the full dataset
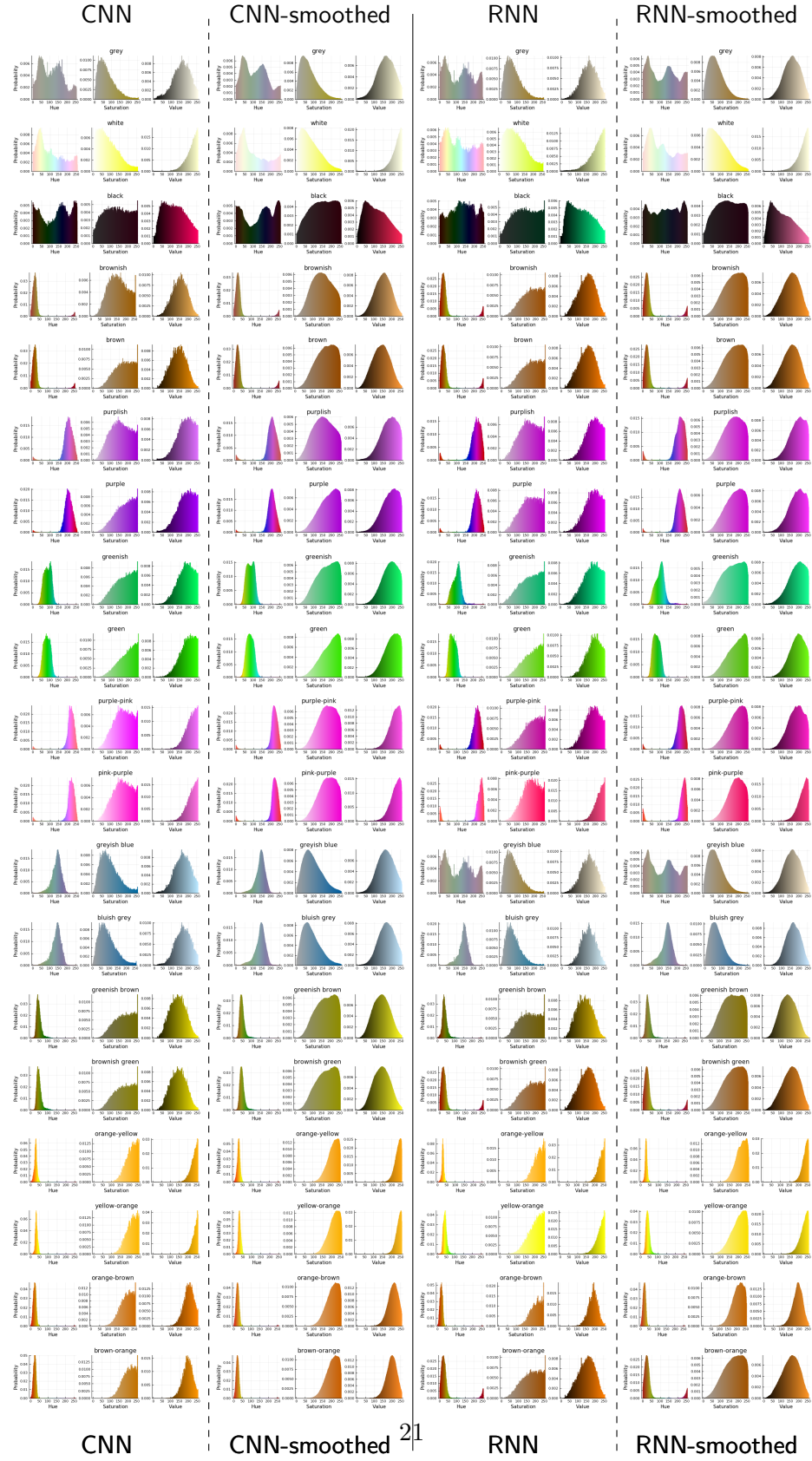
Figure 7: Some examples of the output distribution estimations from the models trained on the full dataset

Figure 8: Some examples of the output distribution estimations from the models trained on the full dataset

6.2 Completely Unseen Color Estimation From Embedding Only

Figure 9 and Figure 10 show quantitative examples of estimated colors for color-names that do not occur in the training data (or testing) at all. This is even more extreme than the extrapolation tasks considered in Table 3 and Table 6 where the terms appeared in training but not the combination of terms. In these examples the terms are never occurred in the training data, but out models exploit the fact that they work by transforming the embedding space to use the word embeddings to predict the colors for them anyway. There is no equivalent for this in the direct models. While `Grey` and `gray` never occur in the training data; `grey` does, and it is near by in embedding space; similar is true for the other colors that vary by capitalization. We only present a few examples of single term colors here, and no quantitative investigation, as this is merely a matter of interest.

It is particularly interesting to note that that all the models both for point estimation and distribution estimation make similar estimations for each color. They do well on the same colors and make similar mistakes on the colors they do poorly at. The saturation of `Gray` is estimated too high, making it appear too blue/purple, this is also true of `grey` though to a much lesser extent. `Purple` and `Green` produce generally reasonable estimates. The hue for `Brown` is estimated as having too much variance, allowing the color to swing into the red or yellowish-green parts of the spectrum. In general the overall quality of each model seems to be in line with that found in the quantitative results for the full tests.
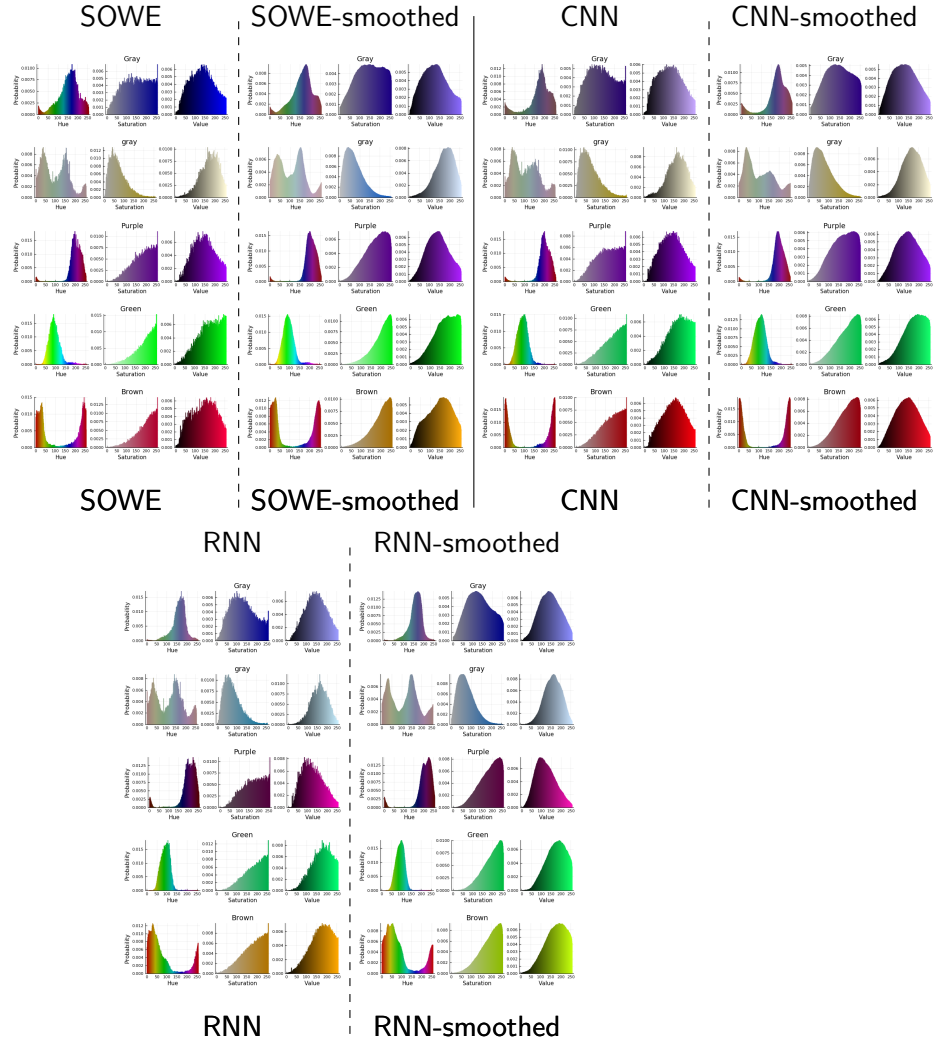
Figure 9: Some example distribution estimations for colors names which are completely outside the training data. The terms: Brown, gray, Gray, Green, and Purple, do not occur in any of the color data; however brown, grey green, and purple do occur.

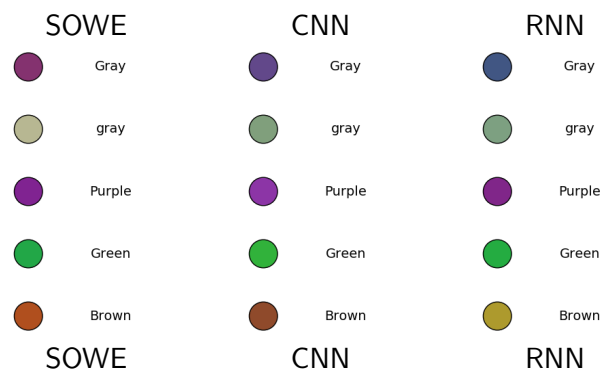Figure 10: Some example point estimates for colors names which are completely outside the training data. The terms: `Brown`, `gray`, `Gray`, `Green`, and `Purple`, do not occur in any of the color data; however `brown`, `grey` `green`, and `purple` do occur.

6.3 Quantitative Results

Overall, we see that our models are able to learn to estimate colors based on sequences of terms. From the consideration of all results (Table 6, Table 3, Table 4, Table 1, Table 2, Table 5) The CNN and SOWE models perform almost as well as the direct methods. With the SOWE taking the lead on most tasks CNN for point estimation. We believe the reason for this is the SOWE is an easier to learn model from a gradient descent perspective: is very shallow model with only one true hidden layer. The RNN did not perform as well at these tasks. While it is only marginally behind the SOWE and CNN on the full point estimation task (**??**), on all other tasks for both point estimation and distribution estimation is is significantly worse. This may indicate that it is hard to capture the significant relationship between terms in the sequence.

However, as will be shown in the examples in the next section it did learn generally acceptable colors.

The performance of SOWE on the order tasks (Table 2 and Table 5) is surprising. SOWE can not take into account word order, its good results suggest that the word-order is not very significant for color names. While word order matters, colors with the same terms in there name but in different order are similar enough that it still performs well.

It can be seen that smoothing has very little effect on the performance of any of the neural network based distribution estimation models. All 3 term based models (SOWE, CNN, RNN) all perform very similarly whether or not the training data is smoothed. The direct method which bypasses learning the functioning of terms is very substantially effected by the smoothing. This is because without smoothing it results in estimating the probability based on bins unfilled by any observation (Which is zero due to the cap on the minimum value. See Section 5.5.1). This is particularly notable in the case of the direct, (unsmoothed) nonextrapolating result reported in Table 3. As these were some of the rarest terms in the training set, they were thus more likely to not coincide with any terms from the testing set. Conversely, on this dataset the term models do quiet well, with or without smoothing. The SOWE model out out performs the the Direct method. As the network can effectively learn the smoothness, not just from the observations of one color but from all of the observations. It learns that increasing the value of one bin should increase adjacent es. As such it does not need the smoothing applied to the training data.

We see both for point estimation (Table 6) and for distribution estimation (Table 3), when the network is forced to extrapolate to new combinations of color names, the SOWE and CNN can do so with a reasonable degree of correctness. The RNN results continue to perform badly when extrapolating, doing much worse than the already poor results on the non-extrapolating task. However, the CNN and SOWE do well; while as expected

Method	$\frac{PP}{256^3}$
Direct	0.164
Direct-smoothed	0.071
SOWE	**0.075**
SOWE-smoothed	**0.075**
CNN	0.078
CNN-smoothed	0.079
RNN	0.089
RNN-smoothed	0.088

Table 1: The results for the **full distribution estimation task**. Lower perplexity (PP) is better.

Method	$\frac{PP}{256^3}$
Direct	0.244
Direct-smoothed	0.053
SOWE	**0.055**
SOWE-smoothed	**0.055**
CNN	0.057
CNN-smoothed	0.058
RNN	0.124
RNN-smoothed	0.122

Table 2: The results for the **order distribution estimation task**. Lower perplexity (PP) is better. This is a subset of the full test set containing only tests where the order of the words matters.

the results are worse than for non-extrapolating they are not much worse. For the Distribution extrapolation task, they are actually still below the overall full datasets best result for perplexity.

In the point estimation results we find that using the networks trained specifically for point estimation (As described by Section 5.6.2) performs consistently better than taking the weighted mean of the distributions output by the distribution estimation methods. We attribute this primarily to the effective additional training data. The distribution estimation method combines all training observation points in to one training datum per color description. Where as the point estimation uses the points directly. We note that for the direct method, the distributions mean is almost identical to the true mean of points, as is expected.

Method	Nonextrapolating $\frac{PP}{256^3}$	Extrapolating $\frac{PP}{256^3}$
Direct	175.883	–
Direct-smoothed	0.050	–
SOWE	**0.050**	**0.055**
SOWE-smoothed	**0.050**	0.056
CNN	0.052	0.065
CNN-smoothed	0.053	0.063
RNN	0.117	0.182
RNN-smoothed	0.112	0.183

Table 3: The results for the **extrapolation distribution estimation task**. Lower perplexity (PP) is better. This uses the extrapolation subset of the full test set. In the extrapolating results certain rare word combinations were removed from the training and development sets. In the non-extrapolating results the full training and development stet was used.

Method	MSE
Direct	0.066
SOWE	**0.067**
CNN	**0.067**
RNN	0.071
Distribution Mean Direct	0.066
Distribution Mean Direct-smoothed	0.066
Distribution Mean SOWE	0.068
Distribution Mean SOWE-smoothed	0.068
Distribution Mean CNN	0.069
Distribution Mean CNN-smoothed	0.069
Distribution Mean RNN	0.077
Distribution Mean RNN-smoothed	0.077

Table 4: The results for the **full point estimation task**. Lower mean squared error (MSE) is better.

Method	MSE
Direct	0.065
SOWE	**0.066**
CNN	**0.066**
RNN	0.096
Distribution Mean Direct	0.065
Distribution Mean Direct-smoothed	0.065
Distribution Mean SOWE	**0.066**
Distribution Mean SOWE-smoothed	**0.066**
Distribution Mean CNN	**0.066**
Distribution Mean CNN-smoothed	**0.066**
Distribution Mean RNN	0.095
Distribution Mean RNN-smoothed	0.097

Table 5: The results for the **order point estimation task**. Lower mean squared error (MSE) is better. This is a subset of the full test set containing only tests where the order of the words matters.

Method	Nonextrapolating MSE	Extrapolating MSE
Direct	0.062	–
SOWE	**0.065**	0.079
CNN	0.072	**0.070**
RNN	0.138	0.142
Distribution Mean Direct	0.062	–
Distribution Mean Direct-smoothed	0.062	–
Distribution Mean SOWE	0.073	0.076
Distribution Mean SOWE-smoothed	0.073	0.076
Distribution Mean CNN	0.073	0.084
Distribution Mean CNN-smoothed	0.073	0.094
Distribution Mean RNN	0.105	0.152
Distribution Mean RNN-smoothed	0.105	0.151

Table 6: The results for the **extrapolation point estimation task**. Lower mean squared error (MSE) is better. This uses the extrapolation subset of the full test set. In the extrapolating results certain rare word combinations were removed from the training and development sets. In the non-extrapolating results the full training and development stet was used.

7 Conclusion

The RNN is the standard type of model for this task. However, we find its performance to be significantly exceeded by the SOWE and CNN. The SOWE is an unordered model correspondent to a bag of words. The CNN over word embeddings is correspondent to a bag of ngrams, in our case as bag of all 1,2,3,4 and 5-grams. This means the CNN can readily take advantage of both fully ordered information, using the filters of length 5, down to unordered information using the filters of length one. The RNN however must fully process the ordered nature of its inputs, as its output comes only from the final node. It would be interesting to further contrast a bidirectional RNN.

We believe that to improve results further for the distribution estimation models, it would benefit from not merging the training data into a single histogram per color. But rather converting each observation into a one-hot representation of its bin. We initially avoiding this solution as it precludes (directly) using KDE-based smoothing of the histograms. However our results in this work indicate that the machine learning distribution estimation models is not substantially effected by the use of KDE-based smoothing or not. While the single observations represented as one-hot would be less informative, there would be several orders of magnitude more of them, and the natural of training with random minibatches would result in a model that is less likely to get stuck in local minima, as so may achieve better results.

A further interesting avenue for investigation would condition the model not only on the words used but also on the speaker. The original source of the data **?**, includes some demographic information which is not exploited by any known methods. It is expected that color-term usage may vary with subcultures.

A   On the Conditional Independence of Color Channels given a Color Name

As discussed in the main text, we conducted a superficial investigation into the truth of our assumption that given a color name, the distributions of the hue, value and saturation are statistically independent.

We note that this investigation is, by no means, conclusive though it is suggestive. The investigation focusses around the use of Spearman's rank correlation. This correlation measures the monotonicity of the relationship between the random variables. A key limitation is that the relationship may exist but be non-monotonic. This is almost certainly true for any relationship involving channels, such as hue, which wrap around. In the case of such relationships Spearman's correlation will underestimate the true strength of the relationship. Thus, this test is of limited use in proving the conditional independence. However, it is a quick test to perform and does suggest that the conditional independence assumption may not be so incorrect as one might assume.

For the Monroe Color Dataset training data given by $V \subset \mathbb{R}^3 \times T$, where $\mathbb{R}^3$ is the value in the color-space under consideration, and T is the natural language space. The subset of the training data for the description $t \in T$ is given by $V_{|t} = \{(\tilde{v}_i, t_i) \in V \mid t_i = t\}$. Further let $T_V = \{t_i \mid (\tilde{v}, t_i) \in V$ be the set of color names used in the training set. Let $V_{\alpha|t}$ be the α channel component of $V_{|t}$, i.e. $V_{\alpha|t} = \{v_\alpha \mid ((v_1, v_2, v_3), t) \in V_{|t}\}$.

The set of absolute Spearman's rank correlations between channels a and b for each color name is given by $S_{ab} = \{|\rho(V_{a|t}, V_{b|t})| \mid t \in T_V\}$.

Color-Space	$Q3(S_{12})$	$Q3(S_{13})$	$Q3(S_{23})$	max
HSV	0.1861	0.1867	0.1628	0.1867
HSL	0.1655	0.2147	0.3113	0.3113
YCbCr	0.4005	0.4393	0.3377	0.4393
YIQ	0.4088	0.4975	0.4064	0.4975
LCHab	0.5258	0.411	0.3688	0.5258
DIN99d	0.5442	0.4426	0.4803	0.5442
DIN99	0.5449	0.4931	0.5235	0.5449
DIN99o	0.5608	0.4082	0.5211	0.5608
RGB	0.603	0.4472	0.5656	0.603
Luv	0.5598	0.6112	0.4379	0.6112
LCHuv	0.6124	0.4072	0.3416	0.6124
HSI	0.2446	0.2391	0.6302	0.6302
CIELab	0.573	0.4597	0.639	0.639
xyY	0.723	0.5024	0.4165	0.723
LMS	0.968	0.7458	0.779	0.968
XYZ	0.9726	0.8167	0.7844	0.9726

Table 7: The third quartile for the pairwise Spearman's correlation of the color channels given the color name.

We consider the third quartile of that correlation as the indicative statistic in Table 7. That is to say for 75% of all color names, for the given color-space, the correlation is less than this value.

Of the 16 color-spaces considered, it can be seen that the HSV exhibits the strongest signs of conditional independence – under this (mildly flawed) metric. More properly put, it exhibits the weakest signs of non-independence. This includes being significantly less correlated than other spaces featuring circular channels such as HSL and HSI.

Our overall work makes the conditional independence assumption, much like n-gram language models making Markov assumption. The success of the main work indicates that the assumption does not cause substantial issues.