Chapter 1

Learning of Colors from Color Names: Distribution and Point Estimation

Abstract

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), to estimate colors from sequences of terms. We consider both point as well as distribution estimates of color. We argue that the latter has a particular value 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". Surprisingly, the sum of word embeddings generally performs the best on almost all evaluations.

1.1 Introduction

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 Australian bushland. Thus the *color intended* cannot be uniquely inferred from a color name. Without further context, it does nevertheless remain possible to estimate likelihoods of which colors are intended based on the population's use of the words.

Color understanding, that is, generating color from text, is an important subtask in natural language understanding, indispensable for *executable semantic parsing*. For example, in a natural language enabled human-machine interface, when asked to select the `dark bluish green` object, it would be much useful if we could rank each object based on how likely its color matches against a learned distribution of the color name `dark bluish green`. This way if the most-likely object is eliminated (via another factor), the second most likely one can be considered. A threshold can be set to terminate the search. This kind of likelihood based approach is not possible when we have only exact semantics based on point estimates.

Color understanding is a challenging domain, due to high levels of ambiguity, the multiple roles taken by the same words, the many modifiers, and the many 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 to demonstrate the capability of the-state-of-the-art image generation systems (**reed2016generative**; **2015arXiv151102793M**). The core focus of the work we present here is to address these linguistic phenomena around the short-phrase descriptions of a color, which can be represented in a color-space such as HSV (**smith1978color**). Issues of illumination and perceived color based on context are considered out of the scope.

1.1.1 Distribution vs. Point Estimation

As illustrated, proper understanding of color names 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 a high and asymmetrical variance, which further renders regression to a single point unsuitable. Having said that, we do produce point estimate results for completeness, though we argue the true usefulness of such estimates is limited.

A single point estimate, does not capture the diverse nature of the color names adequately. Moreover, it is impossible to find the single best point estimation method. For example: for a bimodal distribution, using the distribution mean as a point estimate will select a point in the valley between the peaks, which is less likely. Similarly for an asymmetrical distribution, where the mean will be off to one side of the peak. Conversely, using the distribution mode, will select the highest (most likely) peaks, but will on average be more incorrect (as measured by the mean squared error). The correct trade-off, if a point estimate is required, is dependent on the final use of the system. Another problem is that point estimates do not capture the sensitivity. In an asymmetrical distribution, having a point slightly off-centre in one direction may result in a 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 almost always multimodal or asymmetrical, and exhibit widely differing variances for different names (this can be seen in the histograms of the training data shown in Section 1.6.1). While by definition the mean color point minimizes the squared error, it may not actually be a meaningful point estimate. Given these issues, producing a point estimate has only limited value and estimating a distribution is more general task. However we do consider the point estimation task, as it allows contrast in assessing the input module (SOWE/CNN/RNN) of our proposed methods across the two different output modules (distribution/point estimation).

The generation of color from text has not received much attention in prior work. To the best of our knowledge, the only similar work is **DBLP:journals/corr/KawakamiDRS16**; which only considers point estimation, and uses a dataset containing far too few observations to allow for learning probability distributions from population usages of the color names. To our knowledge The generation of probability distributions in color-space based on sequences of terms making up the color name, has not been considered at all by any prior work. There has been several works on the reverse problem (**mcmahan2015bayesian**; **meomcmahanstone:color**; **2016arXiv160603821M**): the generation of a textual name for a color from a point in a color-space. From the set of work on the reverse problem there is a clear trend on data-driven approaches in recent years where more color names and observations are used.

1.1.2 Contributions

Problem statement: given a set of $\langle$color-name, $(h, s, v)\rangle$ pairs, we need to learn a mapping from any color-name, seen or unseen to a color-value or a distribution in HSV space.

We propose a neural network based architecture that can be broken down into an **input module**, which learns a vector representation of color-names, and a connected **output module**, which produces either a probability distribution or a point estimate. The **output module** uses a softmax output layer for probability distribution estimation, or a novel HSV output layer for point estimation. To carry out the representational learning of color-names, three different color-name embedding learning models are investigated for use in the **input module**: Sum Of Word Embeddings (SOWE), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). The capacity of these input models is of primary interest to this work.

To evaluate and compare the three learning models, we designed a series of experiments to assess their capability in capturing compositionality of language used in color names. These include: **(1)** evaluation on all color names (full task); **(2)** evaluation on color names when the order of the words matters (order task); **(3)** evaluation on color names which never occur in the training data in that exact form, but for which all terms occur in the training data (unseen combination task); **(4)** qualitative demonstration of outputs for color names with terms which do not occur in the training data at all, but for which we know their word embeddings (embedding only task).

An interesting challenge when considering this discretization is the smoothness of the estimate. The true space is continuous, even if we are discretizing it at a resolution as high as the original color displays which were used to collect the data. Being continuous means that a small change in the point location in the color-space should correspond to a small change in how likely that point is according to the probability distribution. Informally, this means the histograms should look smooth, and not spiky. We investigated using a Kernel Density Estimation (KDE) based method for smoothing the training data, and further we conclude that the neural networks learn

this smoothness. To qualify our estimate of the distribution, we discretize the HSV color-space to produce a histogram. This allows us to take advantage of the well-known softmax based methods for the estimation of a probability mass distribution using a neural network.

We conclude that the SOWE model is generally the best model for all tasks both for distribution and point estimation. It is followed closely by the CNN; with the RNN performing significantly worse (see Section 1.6). 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.

To the best of knowledge, this is the first of such investigation in term-wise mapping color names to values that can generate a visual representation, which bring the future of natural language controlled computer graphics generation and executable semantic parsing one step closer to reality.

1.2 Related Work

The understanding of color names has long been a concern of psycholinguistics and anthropologists (**berlin1969basic**; **heider1972universals**; **HEIDER1972337**; **mylonas2015use**). It is thus no surprise that there should be a corresponding field of research in natural language processing.

The earliest works revolve around explicit color dictionaries. This includes the ISCC-NBS color system (**kelly1955iscc**) of 26 words, 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 **berlin1969basic**. Works including **Berk:1982:HFS:358589.358606**; **conway1992experimental**; **ele1994computational**; **mojsilovic2005computational**; **menegaz2007discrete**; **van2009learning** which propose methods for the automatic mapping of colors to and from these small manually defined sets of colors. We note that **menegaz2007discrete**; **van2009learning** both propose systems that discretize the color-space, though to a much coarser level than we consider in this work.

More recent works, including the work presented in this article, function with a much larger number of colors, larger vocabularies, and larger pools of respondents. In particular making uses of the large Munroe dataset **Munroe2010XKCDdataset**, as we do here. This allows a data driven approach towards the modelling.

mcmahan2015bayesian and **meomcmahanstone:color** present color naming methods. Their work primarily focuses on mapping from colors to to their exact 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.

2016arXiv160603821M map a point in the color-space, to a sequence of probability estimates over color words. They extend 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 output – this is a conditional language model. We tackle the inverse problem to the creation of a conditional language model. Our distribution estimation models map from a sequence of terms, to a distribution in color-space. Similarly, our point estimation models map from a sequence of terms to a single point in color-space.

DBLP:journals/corr/KawakamiDRS16 proposes another compositional color naming model. They use a per-character RNN and a variational autoencoder approach. It is in principle very similar to **2016arXiv160603821M**, 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 a 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 to compare with their work.

DBLP:journals/corr/MonroeHGP17 present 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 color 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 **2016arXiv160603821M**,

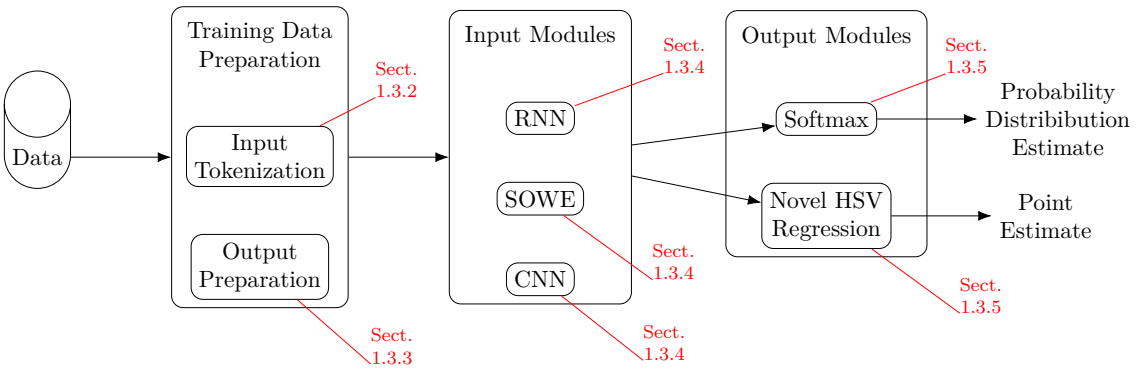

Figure 1.1: The overall architecture of our systems

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.

acl2018WinnLighter demonstrates a neural network for producing point estimates of how colors change based on the use of comparative terms such as `lighter`. The network's input is word embedding for a comparative adjective, and a point in RGB color-space; the model outputs a point in predicted RGB space that is the modified version of the input color point according to the given adjective. For example mapping from green to a darker green is: $((164, 227, 77), \text{darker}) \mapsto (141, 190, 61)$. The color adjectives may have up to two words, to allow for expressions such as `more neon`. This is allowed by taking as the fixed sized input two embeddings – when only one input is required, the other is replaced by zero vector. Their training and evaluation is based on data sourced from the Munroe dataset.

The work presented here closes the gap, that while we have language models conditional upon color, we do not have color models conditional upon language.

1.3 Method

1.3.1 System Architecture

Our overall system architecture for all models is shown in Figure 1.1. This shows how color names are transformed into distribution or point estimates over the HSV color-space.

1.3.2 Input Data Preparation

We desire a color prediction model which takes as input a sequence of words that make up the color's name; rather than simply mapping from the whole phrase (whole phrase mapping does not scale to new user input, given the combinatorial nature of language). Towards this end, color names are first tokenized into individual words. For the input into our neural network based models, these words are represented with pretrained word embeddings.

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`]. Other than spelling, the tokenization results in the splitting of affixes and combining tokens (such as hyphens). 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 1.3.2). The portmanteau `blurple` is not in common use in any training set used for creating word embeddings, so no pretrained embedding is available.[1] 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 reduces the sequential modelling task to the point of being uninteresting.

Embeddings

All our neural network based solutions incorporate an embedding layer. This embedding layer maps from tokenized words to vectors. We make use of 300 dimensional pretrained FastText embeddings (**bojanowski2016enriching**)[2].

The embeddings are not trained during the task, but are kept fixed. As per the universal approximation theorem (**leshno1993uat**; **SONODA2017uat**) the layers above the embedding layer allow for arbitrary continuous transformations. By fixing the embeddings, and learning this transformation, we can produce estimates of colors from embeddings alone, without any training data at all, as shown in Section 1.6.3.

1.3.3 Output Data Preparation for Training Distribution Estimation Models

To train the distribution estimation models we need to preprocess the training data into a distribution. The raw training data itself, is just a collection of $\langle$color-name, $(h, s, v)\rangle$ pairs – samples from the distributions for each named-color. This is suitable for training the point estimation models, but not for the distribution estimation models . We thus convert it into a tractable form, of one histogram per output channel – by assuming the output channels are conditionality independent of each other.

Conditional Independence Assumption

We make the assumption that given the name of the color, the distribution of the hue, saturation and value 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 **meomcmahanstone:color** and **mcmahan2015bayesian**. 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 for most colors (Q3 = 0.187). However, this measure underestimates correlation for values which have a circular relative value, such as hue. Of the 16 color-spaces evaluated, HSV had the lowest correlation by a large margin. Full details, including the table of correlations, are available in supplementary materials (Appendix .1.1). 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; as it does not impact on the correctness of results. A method that does not make this assumption may perform better when evaluated using the same metrics we use here.

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 1.3.3), we output one 256-bin histogram per channel. We note that 24-bit color (as was used in the survey that collected the dataset) can have all the

[1] Methods do exist to generate embeddings for out of vocabulary words (like `blurple`), particularly with FastText embeddings (**bojanowski2016enriching**). But we do not investigate those here.

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

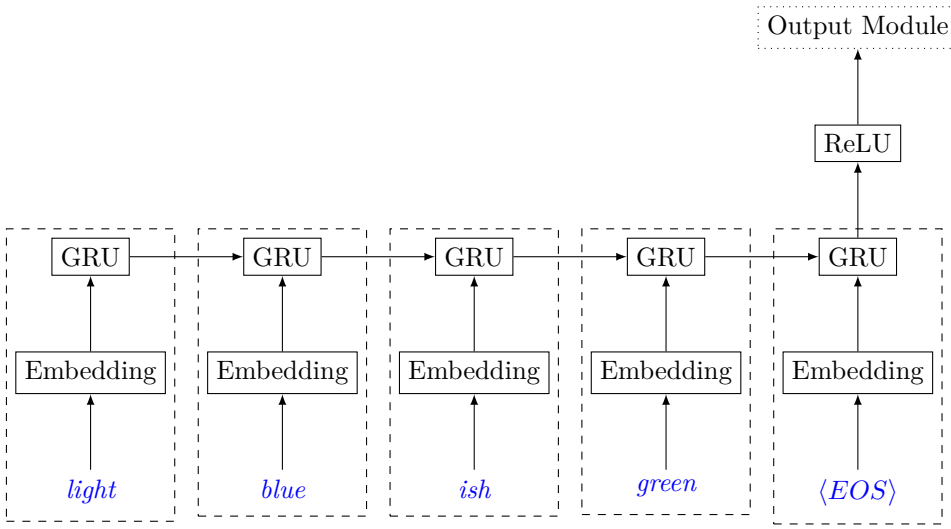

Figure 1.2: The RNN Input module for the example input `light greenish blue`. Each dashed box represents 1 time-step.

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 one-hot 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 investigate 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 a useful solution used in several other machine learning systems. **oord2016pixel**; **DBLP:journals/corr/OordDZSVGKSK16** apply a similar discretization step and found that their method to outperform the more complex truly continuous distribution outputting systems.

For training purposes we thus convert all the observations into histograms. One set of training histograms is produced per color description present in the dataset – that is to say a training histogram is create for `brownish green`, `greenish brown`, `green` etc. We perform uniform weight attribution of points to bins as described by **jones1984remark**. In-short, this method of tabulation is to define the bins by their midpoints, and to allocate probability mass to each bin based on how close an observe point is to the adjacent midpoints. A point precisely on a midpoint would have all its probability mass allocated to that bin; whereas a point halfway between midpoints would have 50% of its mass allocated to each. This is the form of tabulation commonly used during the first step of performing kernel density estimation, prior to the application of the kernel.

1.3.4 Color Name Representation Learning (Input Modules)

For each of the models we investigate we define an input module. This module handles the input of the color name, which is provided as a sequence of tokens. It produces a fixed sized dense representation of the color name, which is the input to the output module Section 1.3.5). In all models the input and output modules are trained concurrently as a single system.

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 1.2 is similar to **2016arXiv160603821M**, 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, 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 (**cho2014properties**), which we found to marginally out-perform the

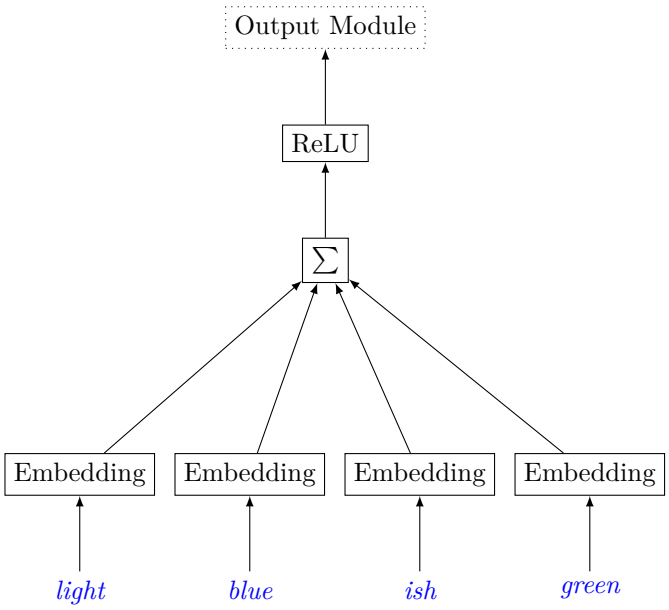

Figure 1.3: The SOWE input module for the example input `light bluish green`

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.

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 as an input to other classifiers such as support vector machines (e.g. as in **White2015SentVecMeaning**; **novelperspective**). Any number of word embeddings can be added to the sum, thus allowing it to handle sequences of any length. However, it has no representation of the order. The structure we used is shown in Figure 1.3.

Convolutional Neural Network(CNN)

A convolutional neural network (shown in Figure 1.4) can be applied to the task by applying 2D convolution over the stacked word embeddings. We use 64 filters of size between one and five. Five is number of tokens in the longest color-name, so this allows it to learn full length relations.

1.3.5 Distribution and Point Estimation (Output Modules)

On top of the input module, we define an output module to suit the neural network for the task of either distribution estimation or point estimation. The input module defines how the terms are composed into the network. The output module defines how the network takes its hidden representation and produces an output.

Distribution Estimation

The distributions are trained to produce the discretized representation as discussed in Section 1.3.3. Making use of the conditional independence assumption (see Section 1.3.3), we output the three discretized distributions. As shown in Figure 1.5, this is done using three softmax output layers – one per channel. They share a common input, but have separate weights and biases. The loss function is given by the sum of the cross-entropy for each of the three softmax outputs.

Point Estimation

Our point estimation output module is shown in Figure 1.6. The hidden-layer from the top of the input module is fed to four single output neurons.[3] Two of these use the sigmoid activation

[3]Equivalently these four single neurons can be expressed as a layer with four outputs and two different activation functions.

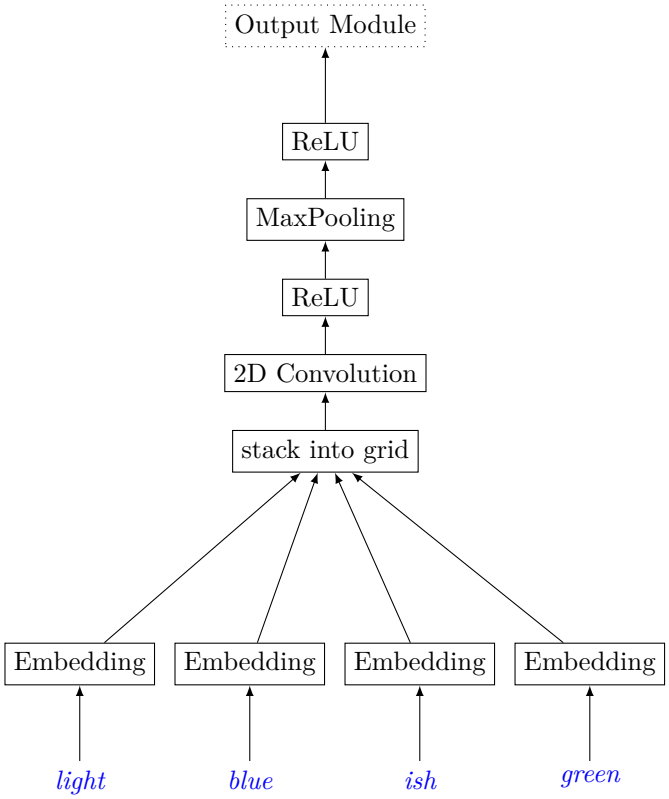

Figure 1.4: The CNN input module for the example input `light greenish blue`

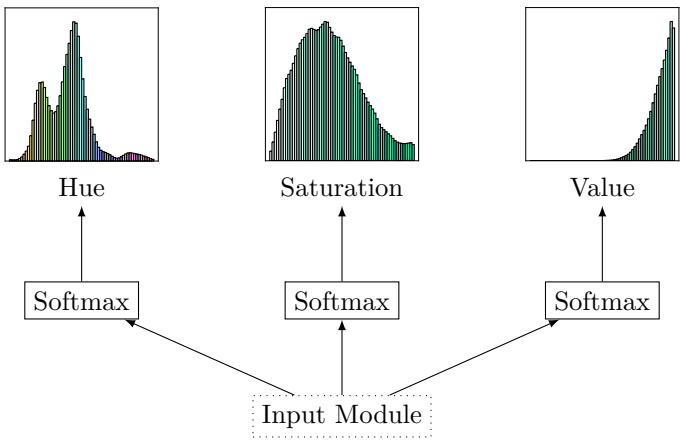

Figure 1.5: The Distribution Output Module

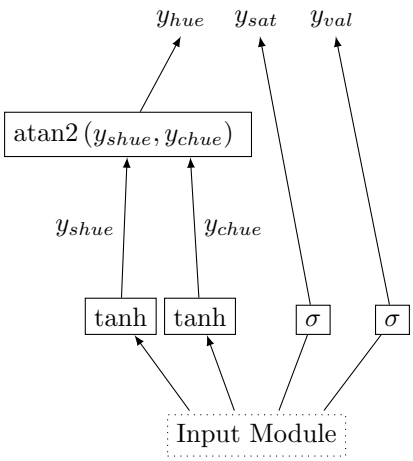

Figure 1.6: The Point Estimate Output Module. Here atan2 is the quadrant preserving arctangent, outputting the angle in turns.

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), and produce the intermediate output that we call y_{shue} and y_{chue} for the sine and cosine of the hue channel respectively. The hue can be found as $y_{hue} = \text{atan2}\,(y_{shue}, y_{chue})$. We use the intermediate values when calculating the loss function. During training we use the following loss function for each observation $y^\star$, and each corresponding prediction y.

$$\begin{aligned} loss = &\frac{1}{2}\left(\sin(y^\star_{hue}) - y_{shue}\right)^2 \\ &+ \frac{1}{2}\left(\cos(y^\star_{hue}) - y_{chue}\right)^2 \\ &+ \left(y^\star_{sat} - y_{sat}\right)^2 \\ &+ \left(y^\star_{val} - y_{val}\right)^2 \end{aligned} \tag{1.1}$$

This mean of this loss is taken over all observations in each mini-batch during training. This loss function is continuous and correctly handles the wrap-around nature of the hue channel (**WhiteRepresentingAnglesSE**).

1.4 Evaluation

1.4.1 Perplexity in Color-Space

Perplexity is a measure of how well the distribution, estimated by the model, matches the reality according to the observations in the test set. Perplexity is commonly used for evaluating language models. Here however, it is being used to evaluate the discretized distribution estimate. It can be loosely thought of as to how well the model's distribution does in terms of the size of an equivalent uniform distribution. Note that this metric 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 a color-space point $\tilde{x}$; and $p(\tilde{x} \mid t)$ is the output of the evaluated model. Perplexity is defined as:

$$PP(\tau) = \exp_2\left(\left(\frac{-1}{|\tau|}\sum_{\forall(t,(\tilde{x}))\in\tau}\log_2 p(\tilde{x} \mid t)\right)\right) \tag{1.2}$$

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 that the result is equal to what we would see 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 is

twice as good as if we were to simply use a uniform distribution: it is equivalent to saying that the correct bin is selected as often as it would be had a uniform distribution with half as many bins been used (i.e. larger bins with twice the area). 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.

1.4.2 Angularly Correct Calculations on HSV

We use the HSV color-space (**smith1978color**) through-out this work. In this format: hue, saturation and value all range between zero and one. Note that we measure hue in *turns*, rather than the more traditional degrees, or radians. Having hue measured between zero and one, like the other channels, makes the modelling task more consistent. Were the hue to range between 0 and 2π (radians) or between 0 and 360 (degrees) it would be over-weighted in the loss function and evaluation metrics 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 combinations of values from the different channels correspond to realizable colors.

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 that a hue of one and a hue of zero are equivalent (as we measure in turns, in radians this would be 0 and 2π).

The square error of two hue values is thus calculated as:

$$SE(h_1, h_2) = \min\left((h_1 - h_2)^2 , (h_1 - h_2 - 1)^2 \right) \tag{1.3}$$

This takes into account that the error can be calculated clockwise or counter-clockwise; and should be the smaller. Note that the -1 term is 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:

$$\bar{h} = \operatorname{atan2}\left(\frac{1}{N}\sum_{i=1}^{i=N} \sin(h_i),\ \frac{1}{N}\sum_{i=1}^{i=N} \cos(h_i) \right) \tag{1.4}$$

This gives the mean angle.

1.4.3 Operational Upper-bounds

To establish a rough upper-limit on the modelling results we evaluate a direct method which bypasses the language understanding component of the task. These direct methods do not process each term in the name:, they do not work with the language at all. They simply map from the exact input text (no tokenization) to the pre-calculated distribution or mean of the training data for the exact color name. This operational upper bound bypass the compositional language understanding part of the process. It is as if the input module (as discussed in Section 1.3.4) would perfectly resolve the sequence of terms into a single item.

They represent an approximate upper bound, as they fully exploit all information in the training data for each input. There is no attempt to determine how each term affects the result. We say approximate upper-bound, as it is not strictly impossible that the term-based methods may happen to model the test data better than can be directly determined by training data as processed per exact color name. This would require learning how the terms in the color name combine in a way that exceeds the information directly present in the training data per class. It is this capacity of learning how the terms combine that allow for the models to predict the outputs for combinations of terms that never occur in the training data (Section 1.4.4). Doing this in a way that generalizes to get better results than the direct exploitation of the training data, would require very well calibrated control of (over/under)fitting.

Operational Upper-bound for Distribution Estimation: KDE

To determine an operational upper-bound for the distribution estimation tasks, we use kernel-density estimation (KDE) in a formulation for non-parametric estimation (**silverman1986density**) . The KDE effectively produces a smoothed histogram from the training data as processed in Section 1.3.3. It causes adjacent bins to have most similar probabilities, thus matching to the mathematical notion of a continuous random variable. This is applied on-top of the histogram

used for the training data. We use the Fast Fourier Transform (FFT) based KDE method of the **silverman1982algorithm**. 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 effect near the boundaries, where the 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 discarding those bins and rescaling the probability to one. For the hue channel this wrap-around effect is exactly as desired.

In our evaluations using KDE rather than just the training histograms directly proved much more successful on all distribution estimation tasks. This is because it avoids empty bins, and effectually interpolates probabilities between observations. We found in preliminary investigations that using KDE-based method to be much better than add-one smoothing.

We also investigated the application of KDE to the training data, before training our term-based neural network based distribution models. Results for this can be found in Appendix .1.2. In brief, we found that smoothing the training data does not significantly affect the result of the neural network based models. As discussed in Section 1.6.2, this is because the neural networks are able to learn the smoothness relationship of adjacent bins.

Our KDE-based operational upper bound for distribution estimation bypasses the natural language understanding part of the task, and directly uses the standard non-parametric probability estimation method to focus solely on modelling the distributions. Matching its performance indicates that a model is effectively succeeding well at both the natural language understanding component and the distribution estimation component.

Operational Upper-bound for Point Estimation: Mean-point

In a similar approach, we also propose a method that directly 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 (as discussed in Section 1.4.2). Taking the mean of all the observations gives the theoretically optimal solution to minimize the 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 point estimation performance of the term based models. Though, as discussed in Section 1.1, the notion of mean and of minimizing the square error is not necessarily the correct way to characterize selecting the optimal point estimate for colors. It is however a consistent way to do so, and so we use it for our evaluations.

1.4.4 Evaluation Strategies

Full Task

We make use of the Munroe dataset as prepared by **mcmahan2015bayesian** from the results of the XKCD color survey. The XKCD color survey (**Munroe2010XKCDdataset**), 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.

Unseen combination Task

A primary interest in 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 system to make predictions about `bright salmon`, even though that description never occurs in the training data. The ability to make predictions, such as these, illustrates term-based natural language understanding. This cannot be done with the operational upper bound models, which by-passes the term processing step. To evaluate this generalisation capacity, we define new sub-datasets for both testing and training. We select 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 unseen combination testing set has only observations from the full test set that do use those rare descriptions. We define a corresponding restricted training set made up of the data from the full training set, excluding those corresponding to the rare descriptions. Similar restriction is done to create a restricted development set, so that no direct knowledge of the combined terms can leak during early-stopping.

By training on the restricted training set and testing on the unseen combination, we can assess the capacity of the models to make predictions for color descriptions not seen during training. A similar approach was used in **acl2018WinnLighter** and in **DBLP:journals/corr/AtzmonBKGC16**. We contrast this to the same models when trained on the full training set to see how much accuracy was lost.

Order Task

It is believed that the order of words in a color description matters, at least to some extent, for it's meaning. For example, `greenish brown` and `brownish green` are distinct, if similar, colors. To assess the models on their ability to make predictions when order matters we construct the order test set. 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 or more of the terms is a joining token such as `ish` or `-`. We only construct an order testing set, and not a corresponding training set, as this is an evaluation using the model trained on the full training data.

1.5 Experimental Setup

1.5.1 Implementation

The implementation of all the models was in the julia programming language (**Julia**). The full implementation can be downloaded from the GitHub repository.[4] The machine learning components make heavy use of the MLDataUtils.jl[5] and TensorFlow.jl,[6] packages. The latter of which we enhanced significantly to allow for this work to be carried out. The discretization and the KDE for the Operational Upper Bound is done using KernalDensityEstimation.jl.[7]

1.5.2 Common Network Features

Drop-out(**srivastava2014dropout**) is used on all ReLU layers and on the GRU in the RNN, with threshold of 0.5 during training. The network is optimized using Adam (**kingma2014adam**), and a learning rate of 0.001. Early stopping is checked every 10 epochs using the development dataset. Distribution estimation methods are trained using the full batch (where each observation is a distribution) for every epoch. Point Estimation methods are trained using randomized mini-batches of 2^{16} observations (which are each color-space triples). All hidden-layers, except as otherwise precluded (inside the convolution, and in the penultimate layer of the point estimation networks) have the same width 300, as does the embedding layer.

1.6 Results

1.6.1 Qualitative Results

To get an understanding of the problem and how the models are performing, we consider some of the outputs of the model for particular cases. To illustrate the models trained on the full dataset, Figure 1.7 shows examples of distribution estimates, and Figure 1.8 shows similar examples for point estimates. It can be seen that the models' outputs using term based estimation are generally similar to the non-term-based operational upper-bound, as is intended. This shows that the models are correctly fitting to estimate the colors. This aligns with the strong results found in the quantitative evaluations discussed in Section 1.6.2.

[4] Implementation source is at `https://github.com/oxinabox/ColoringNames.jl`
[5] MLDataUtils.jl is available from `https://github.com/JuliaML/MLDataUtils.jl`
[6] TensorFlow.jl is available from `https://github.com/malmaud/TensorFlow.jl`
[7] KernalDensityEstimation.jl is available from `https://github.com/JuliaStats/KernelDensity.jl`

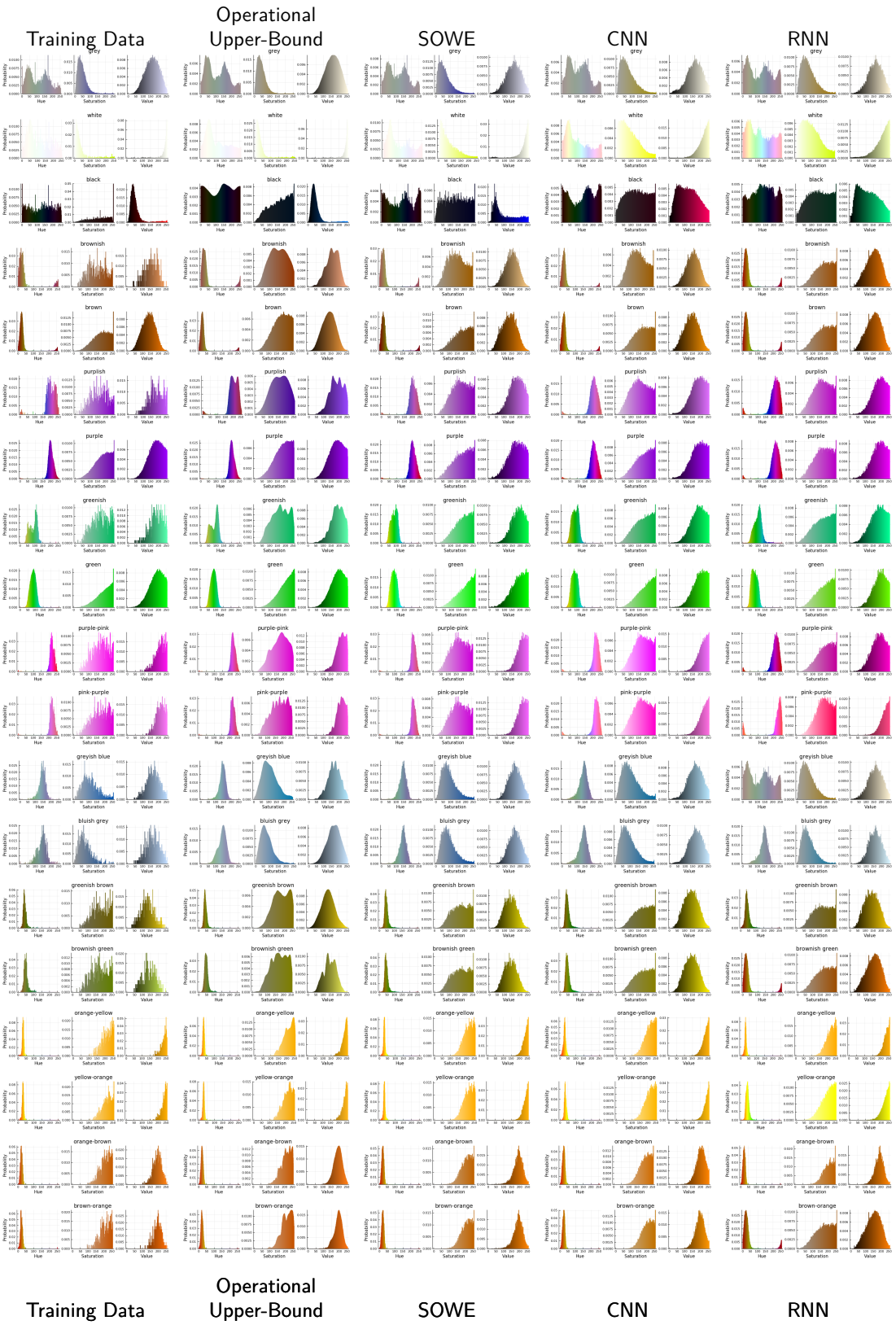

Figure 1.7: Some examples of the output distribution estimates from the models trained on the full dataset

Figure 1.8: Some examples of the output point estimates from the models trained on the full dataset

On the effects of word-order

The different input modules have a different capacity to leverage word-order. This is reflected in Figures 1.7 and 1.8, when considering the pairs of outputs that differ only in word order, such as `purple-pink` and `pink-purple`. The plots presented for the training data and for the Operational Upper-Bound show that such color name pairs are subtly different but similar. The SOWE model is unable to take into account word order at all, and so produces identical outputs for all orders. The CNN models produce very similar outputs but not strictly identical – spotting the difference requires a very close observation. This is in-line with the different filter sizes allowing the CNN to effectively use n-gram features, and finding that the unigram features are the most useful. The RNN produces the most strikingly different results. It seems that the first term dominates the final output: `purple-pink` is more purple, and `pink-purple` is more pink. We can see that the first term 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[8]. This shows that the GRU is functioning to remember the earlier inputs. This is happening too strongly however, as it is causing incorrect outputs.

On the smoothness of the distribution estimates

In Figure 1.7 it can be seen that the term-based distribution estimation models are much smoother than the corresponding histograms taken from the training data. They are not as smooth as the Operational Upper-Bound which explicitly uses KDE. However, they are much smoother than would be expected, had the output bins been treated independently. Thus it is clear that the models are learning that adjacent bins should have similar output values. This is a common feature of all the training data, no matter which color is being described. This learned effect is in line with the fact that color is continuous, and is only being represented here as discrete. We note in relation to this learned smoothness: 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 such as `white`, `grey`, `black`, `white`, are captured; but smaller dips such with as the hue of `greenish` being more likely to be on either side of the green spectrum are largely filled in. In general, it seems clear that additional smoothing of the training data is not required for the neural network based models. This aligns with the results presented in Appendix .1.2.

1.6.2 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 the results shown in Tables 1.1 to 1.6. The CNN and SOWE models perform almost as well as the operational upper-bound. With the SOWE having a marginal lead for distribution estimation, and the CNN and SOWE being nearly exactly equal for most point estimation tasks. We believe the reason for this is that the SOWE is an easier to learn model from a gradient descent perspective: it is a 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 (Table 1.2), on all other tasks for both point estimation and distribution estimation it is significantly worse. This may indicate that it is hard to capture the significant relationships between terms in the sequence. However, as shown Section 1.6.2 it did learn generally acceptable colors, but it is not as close a match to the population's expectation.

Ordered Task

The performance of SOWE on the order tasks (Tables 1.3 and 1.4) is surprising. For the distribution estimation it outperforms the CNN, and for point estimation it ties with the CNN. The CNN and RNN, can take into account word order, but the SOWE model cannot. The good results for SOWE suggest that the word-order is not very significant for color names. While word order matters, different colors with the same terms in different order are similar enough that it still performs very well. In theory the models that are capable of using word order have the capacity to ignore it, and thus could achieve a similar result. An RNN can learn to perform a sum of its inputs (the word embeddings), and the CNN can learn to weight all non-unigram filters to zero. In practice we see that for the RNN in particular this clearly did not occur. This can be attributed to the more complex networks being more challenging to train via gradient descent. It seems that color-naming is not a task where word order substantially matters, and thus the simpler SOWE model excels.

[8] So much so that we double checked our implementation to be sure that it wasn't processing the inputs backwards.

Method	$\frac{PP}{256^3}$
Operational Upper Bound	0.071
SOWE	**0.075**
CNN	0.078
RNN	0.089

Table 1.1: The results for the **full distribution estimation task**. Lower perplexity (PP) is better.

Unseen Combinations of Terms

The SOWE and CNN models are able to generalize well to making estimates for combinations of color terms that are not seen in training. Tables 1.5 and 1.6 show the results of the model on the test set made up of rare combinations of color names (as described in Section 1.4.4) for the restricted training set (which does not contain those terms). These results on this test set are compared with the same models when trained on the full training set. The Operation Upper Bound models are unable to produce estimates from the unseen combinations testing set as they do not process the color names term-wise. The RNN models continue to perform badly on the unseen combination of terms task for both point and distribution estimation.

On distribution estimation (Table 1.5) the SOWE results are only marginally worse for the restricted training set as they are for the full training set. The CNN results are worse again, but they are still better than the results on the full test-set. The distribution estimates are good on absolute terms, having low evaluated perplexity.

In the point estimation task (Table 1.6) the order is flipped with the CNN outperforming the SOWE model. In-fact the CNN actually performs better with the restricted training set. This may be due to the CNN not fitting to the training data as well as the SOWE, as on the full training set (for this test set) we see the SOWE outperforms the CNN. This suggests that the CNN point estimation model may be better at capturing the shared information about term usages, at the expense of fitting to the final point. Unlike for distribution estimates, the unseen color point estimates are worse than the overall results from the full task (Table 1.2), though the errors are still small on an absolute scale.

Extracting the mean from the distribution estimates

In the point estimation results discussed so far, have been from models trained specifically for point estimation (as described by Section 1.3.5). However, it is also possible to derive the mean from the distribution estimation models. Those results are also presented in Tables 1.2, 1.4 and 1.6. In general these results perform marginally worse (using the MSE metric) than their corresponding modules using the point estimation output module. We note that for the operational upper-bound, the distributions mean is almost identical to the true mean of points, as expected.

Beyond the output module there are a few key differences between the point estimation modules and the distribution estimate modules. When training distribution estimation models, all examples of a particular color name is grouped into a single high information training observation using the histogram as the output. Whereas when training for point estimation, each example is processed individually (using minibatches). This means that the distribution estimating models fit to all color names with equal priority. Whereas for point estimates, more frequently used color names have more examples, and so more frequent color names are fit with priority over rarer ones. Another consequence of using training per example using random minibatches, rather than aggregating and training with full batch, is increased resilience to to local minima. One of the upsides of the aggregated training used in distribution estimation is that it trains much faster as only a small number of high-information training examples are processed, rather than a much larger number of individual observations. It may be interesting in future work to consider training the distribution estimates per example using one-hot output representations; thus making the process similar to that used in the point estimate training. We suspect that such a method may have trouble learning the smoothness of the output space (as discussed in Section 1.6.1).

1.6.3 Completely Unseen Color Estimation From Embeddings

As an interesting demonstration of how the models function by learning the transformation from the embedding space to the output ,we briefly consider the outputs for color-names that do not occur in the training or testing data at all. This is even more extreme than the unseen combination

Method	MSE
Operational Upper Bound	0.066
SOWE	**0.067**
CNN	**0.067**
RNN	0.071
Distribution Mean Operational Upper Bound	0.066
Distribution Mean SOWE	0.068
Distribution Mean CNN	0.069
Distribution Mean RNN	0.077

Table 1.2: The results for the **full point estimation task**. Lower mean squared error (MSE) is better.

Method	$\frac{PP}{256^3}$
Operational Upper Bound	0.053
SOWE	**0.055**
CNN	0.057
RNN	0.124

Table 1.3: 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.

Method	MSE
Operational Upper Bound	0.065
SOWE	**0.066**
CNN	**0.066**
RNN	0.096
Distribution Mean Operational Upper Bound	0.065
Distribution Mean SOWE	**0.066**
Distribution Mean CNN	**0.066**
Distribution Mean RNN	0.095

Table 1.4: 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	Full Training Set $\frac{PP}{256^3}$	Restricted Training Set $\frac{PP}{256^3}$
Operational Upper Bound	0.050	–
SOWE	**0.050**	**0.055**
CNN	0.052	0.065
RNN	0.117	0.182

Table 1.5: The results for the **unseen combinations 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	Full Training Set MSE	Restricted Training Set MSE
Operational Upper Bound	0.062	–
SOWE	**0.065**	0.079
CNN	0.072	**0.070**
RNN	0.138	0.142
Distribution Mean Operational Upper Bound	0.062	–
Distribution Mean SOWE	0.073	0.076
Distribution Mean CNN	0.073	0.084
Distribution Mean RNN	0.105	0.152

Table 1.6: The results for the **unseen combinations 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.

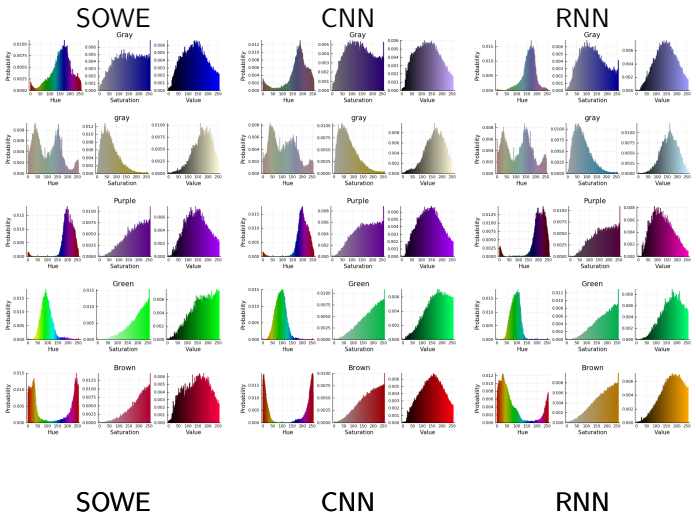

Figure 1.9: Some example distribution estimations for colors names which are completely outside the training data. The terms: `Brown`, `gray`, `Gray`, `Green`, and `Purple`, do not occur in any of the color data; however `brown`, `grey` `green`, and `purple` do occur.

task considered in Tables 1.5 and 1.6 where the terms appeared in training, but not the combination of terms. In the examples shown in Figures 1.9 and 1.10, where the terms never occurred in the training data at all, our models exploit the fact that they work by transforming the word-embedding space to predict the colors. 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 the word-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 the all the models make similar estimations for each color. This occurs both for point estimation and for distribution estimation. 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. This suggests that in general all models are learning a more generally similar transformation of the space. In general the overall quality of each model seems to be in line with that found in the results for the full tests.

1.7 Conclusion

We have presented three input modules (SOWE, CNN, RNN), and two output modules (distribution estimate, and point estimate) that are suitable for using machine learning to make estimates

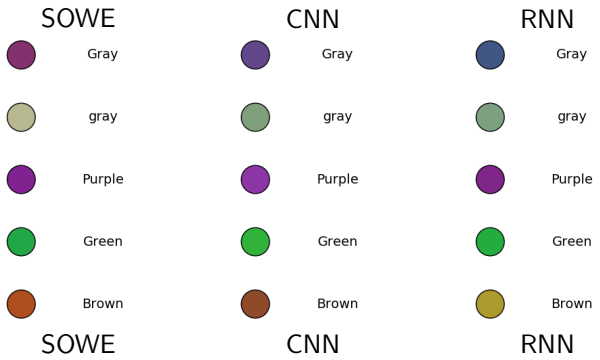

Figure 1.10: Some example point estimates for colors names which are completely outside the training data. The terms: `Brown`, `gray`, `Gray`, `Green`, and `Purple`, do not occur in any of the color data; however `brown`, `grey` `green`, and `purple` do occur.

about color based on the terms making up its name. We contrasted these to an operational upper bound model for each task which bypassed the term-wise natural language understanding component of the problem. We found the results for SOWE, and CNN were very strong, approaching this upper bound.

A key take away from our results is that using a SOWE should be preferred over an RNN for short phrase natural language understanding tasks when order is not a very significant factor. The RNN is the standard type of model for problems with sequential input, such as color names made up of multiple words as we considered here. However, we find its performance to be significantly exceeded by the SOWE and CNN. The SOWE is an unordered model roughly corresponding to a bag of words. The CNN similarly roughly correspondents to a bag of ngrams, in our case a 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 filters of length 1. 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.

In a broader context, we envisage the distribution learned for a color name can be used as a prior probability and when combining with additional context information, this can be used for better prediction in areas such as document classification and sentiment detection.

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 **Munroe2010XKCDdataset**, includes some demographic information which is not exploited by any known methods. It is expected that color-term usage may vary with subcultures.

.1 Appendix

.1.1 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 the 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 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.

In Monroe Color Dataset the 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 make the Markov assumption. The success of the main work indicates that the assumption does not cause substantial issues.

.1.2 KDE based smoothing of Training Data

It can be seen that smoothing has very little effect on the performance of any of the neural network based distribution estimation models. All three term based models (SOWE, CNN, RNN) all perform very similarly whether or not the training data is smoothed. This is seen consistently in all the distribution estimation tasks. Contrast Tables 8 to 10 to the tables for the unsmoothed results Tables 1.1, 1.3 and 1.5.

If however, smoothing is not applied to the operational upper bound, it works far worse. In Tables 8 to 10 the Direct result refers to using the training histograms almost directly, without any smoothing or term-based input processing. This is the same as the operational upper bound, minus the KDE. It works very poorly (by comparison). This is because the bins values are largely independent: a very high probability in one bin does not affect the probability of the adjacent bin – which by chance of sampling may be lower than would be given by the true distribution.

This is particularly notable in the case of the direct, full training set result on the unseen combinations task reported in Table 10. As these were some of the rarest terms in the training set, several did not coincide with any bins for observations in testing set. This is because without smoothing it results in estimating the probability based on bins unfilled by any observation. We do cap that empty bin probability at $\epsilon_{64} \approxeq 2 \times 10^{-16}$ to prevent undefined perplexity. We found capping the lower probability for bins like this to be far more effective than add-on smoothing.

Conversely, on this dataset the neural network models do quite well, with or without smoothing. 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 the adjacent ones. As such smoothing does not need to be applied to the training data.

Method	$\frac{PP}{256^3}$
Direct	0.164
Operational Upper Bound	0.071
SOWE-smoothed	0.075
CNN-smoothed	0.079
RNN-smoothed	0.088

Table 8: The results for the **full distribution estimation task** using smoothed training data. Lower perplexity (PP) is better. This corresponds to the main results in Table 1.1.

Method	$\frac{PP}{256^3}$
Direct	0.244
Operational Upper Bound	0.053
SOWE-smoothed	0.055
CNN-smoothed	0.058
RNN-smoothed	0.122

Table 9: The results for the **order distribution estimation task** using smoothed training data. Lower perplexity (PP) is better. This is a subset of the full test set containing only tests where the order of the words matters. This corresponds to the main results in Table 1.3.

Method	Full Training Set $\frac{PP}{256^3}$	Restricted Training Set $\frac{PP}{256^3}$
Direct	175.883	–
Operational Upper Bound	0.050	–
SOWE-smoothed	0.050	0.056
CNN-smoothed	0.053	0.063
RNN-smoothed	0.112	0.183

Table 10: The results for the **unseen combinations distribution estimation task** using smoothed training data. 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. This corresponds to the main results in Table 1.5.