Zero Shot-learning for Colour Recognition

Seminar in Artificial Intelligence - 188.962

Benedikt Haecker (11713128)

***Abstract*—A Zero-Shot learner can be seen as an extension of a machine learning algorithm, which did not see certain classes during the learning process, but is still able to classify instances into unseen classes. This is done by providing a semantic link between the unseen classes and the data which is available in the training process. We run experiments on a colour data set, where we create an artificial situations, where a linear classifier does not see classes. The semantic link between unlabeled instances and training data is established via textual description of colours.**

I. Introduction

The task of classifying colours is highly subjective. While two persons might have different images of the same colours in their head, a third one simply knows more colours. Still there is a common ground for the most significant and prominent colours. When teaching algorithms the art of colour recognition, one has to rely on such a common ground, at least when using machine learning approaches. We want to extend this task and ask ourselves: Is it possible for a machine learning algorithm to classify a colour, of which it has not seen an example during training phase?

The general frame work for this is so called *zero shot-learning*. In zero shot-learning the classifier has zero training instances of certain classes, but is still able to predict an instance, if it sees one. Since information does not appear out of nothing, there must by any other mechanism to map instances of unseen classes to the correct ones. For our purpose we use the zero shot learner ESAZSL, which learns a mapping between instance features and classes and a semantic embedding on seen classes, which is both transferred, to label instances of unseen classes. For the semantic embedding, we use textual description of individual colours. In this paper we provide:

- a short overview over the framework of zero shot-learning and its relation to unsupervised learning
- a mapping of RGB colour code to colour classes.
- a semantic description of colours

Furthermore we run experiments to test our hypothesis, which states, that there is a common ground when it comes to using words as a description of colours.

II. Related work

Even though zero shot learning is a relative new approach in machine learning, there are numerous algorithms and frameworks available [1]. While machine learning usually deals with supervised learning, it seems that zero shot-learning and unsupervised learning have a lot in common. In fact in the field of domain adaption one can formulate the task of domain adaption in both frameworks [2]: Let $D^s = \{(x_i^s, y_i^s)\}$ be a labelled data set, where x_i^s represent the feature vector and y_i^s the label of the i-th instance in a domain S. For the unsupervised learning task, we want to classify an unlabelled data set $D^t = \{x_i^t\}$ from the target domain T, where the target label space Y^t is equal to the source label space Y^s.

Under zero shot-learning conditions we have D^s as before and $D^{tl} = \{(x_i^{tl}, y_i^{tl})\}$ a labelled data set from the target domain T. The task is to map any given new instance x^t from the target domain to Y^t by learning an inference model $y = f(x^t) \in Y^t$. Notice that $Y^{tl} \subset Y^t = Y^s$, which means that there is only a subset labelled instances available during training.

In this paper we stay within the same domain. This has the advantage that we can make use of so called attributes, which are shared by all classes, regardless if they are seen or unseen during training. Attributes transfer knowledge from seen to unseen classes. But they are not the only source of information. Other approaches use semantic class taxonomies or text features [3]. Some of these approaches overlap or are extensions of each other. For example a semantic word space can be seen as an extension of attributes [4].

Another differentiation which can be made in zero shot learning is, if a zero shot-learner is able to classify an unseen instance into all classes [4] or just into the unseen ones [5]. While we focus on the later case, the zero shot learner we use makes equal use of the relation between instances and classes, learned by the available data, and the attributes, which connect unseen and seen classes. There is also the possibility to use the attributes directly to identify unseen classes, which would bring us back to unsupervised learning [6].

III. ESAZSL

We use the Zero Shot learner with the name "An embarrassingly simple approach to zero-shot learning" (ESAZSL) [5]. The algorithm works like this: At training stage there are z classes. Each class has a typical signature. Each signature consists of so called attributes. All classes share the same attributes, but with different values. The attributes take values between 0 and 1. These values make up the attribute matrix $S \in [0,1]^{a \times z}$, to which we most of the times refer to as *S-matrix*. Note that a is the number of attributes and z the number of classes.

Depending on the loss function L and the regularizer Ω we train a model to obtain a linear classifier W via

$$\underset{W \in \mathbb{R}^{d \times z}}{\text{minimize}}\ L(X^\top W, Y) + \Omega(W), \tag{1}$$

where Y is the ground truth and $X \in \mathbb{R}^{d \times m}$ is the data available at the training stage. By exchanging $W = V \cdot S$ we obtain

$$\underset{V \in \mathbb{R}^{d \times a}}{\text{minimize}}\ L(X^\top V S, Y) + \Omega(V), \tag{2}$$

where $S \in [0,1]^{a \times z}$ is the S-matrix. For unseen classes we provide the attribute matrix S' and solve the linear equation $W = V \cdot S$ for V. The prediction of a new instance x is done via:

$$\underset{i}{\operatorname{argmax}}\ x^\top V S'_i. \tag{3}$$

We note that the prediction of a new instance depends on the attributes and implicit on the linear classifier of the seen classes. We expect the linear classifier to perform better on the seen classes, when doing a normal classification task than the implicit classifier together with the attributes on unseen classes.

IV. Experiment setup

A. Data set

The RGB colour space is spanned by the colour intensity of red, green and blue, which can take discrete values from 0 to 255. Therefore there are $256 \cdot 156 \cdot 256 = 16777216$ colours in the colour space.
For our experiments we use the "Massive color dictionary" [7]. The data set consists of 18181 unique colour names and their corresponding hex code. For that approximately 0.11% of the colour space is named.

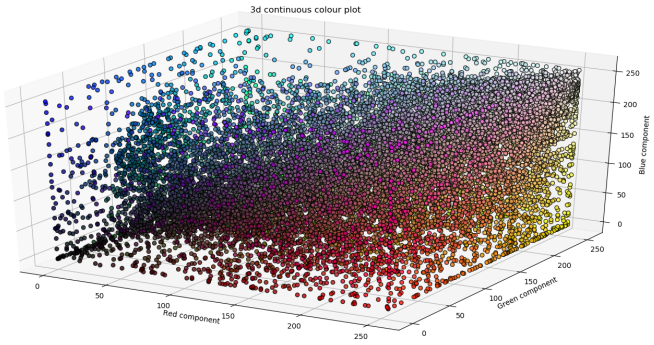

Figure 1: Colours with names in the RGB colour space

The names for colours range from more obvious ones "A Pair of Brown Eyes", "Aqua Deep", "Sahara Sun" over fancy sounding ones "Squeeze Toy Alien", "Snowball Effect, "Homeopathic Lavender" to exotic "Hassan II Mosque", "Goldvreneli 1882, "Giant Onion" and abstract ones "Feminism", "Quack Quack", "8Bit Eggplant".

B. Preprocessing

The hex code value is a six-digit number, which represent a triple of two numbers respectively. The two numbers are the colour intensity written in hex code. We are interested in the RGB values of the colours, so we translate the Hex code to RGB. We then drop the hex code values and obtain a data set with four features: *name, red, green* and *blue*. The last three are numerical values between 0 and 255.
Since we want to perform classification experiments we have to introduce classes in our data set. We want to classify each instance colour into one of the following classes: Red, green, blue, yellow, cyan, magenta, white, grey or black. To put it more formally we need a mapping

$$m : \{0,256\}^3 \to \{1,...,9\}. \tag{4}$$

In order to find such a mapping we explore the relations between the red, green and blue components. We introduce the following mapping, where a is a parameter and R,G,B are the values for red, green and blue respectively:

Colour	Condition
Black	R <a + 32, G <a + 32, B <a + 32
Green	G - R >a - 20, G - B >a - 20
Blue	B - R >a, B - G >a
White	R >255 - a, B >255 - a, G >255 - a
Red	R - G >a + 10, R - B >a + 10
Magenta	abs(R - B) <a - 10
Grey	abs(R - G) <a - 5, abs(R - B) <a - 5, abs(B - G) <a - 5
Yellow	abs(R - G) >a - 20
Cyan	abs(B - G) <a + 30

Table I: Mapping m of RGB values to colour classes

The mapping depends on the threshold a, which controls the quality of the mapping. We state the following requirements for a high quality mapping:

- The colours in a class have to meet our subjective idea of that colour class.
- The classes should be more or less balanced and if that is not possible,
- no class should be overpowered.
- The main colours red, green, blue (and yellow) should be more frequent than magenta and cyan.

Before we can determine a suitable parameter a we notice that the mapping does not map all instances to one of the nine classes. In fact approximately 27% of the instances are not classified after applying m.
To overcome this problem two approaches are made. The first one is to classify a not-yet classified instance by putting it into the classes red, green or blue depending on the highest value among these three. If they are even off, the instance gets classified as grey.
It turns out that these method produces highly overlapping classes as seen in figure 2.

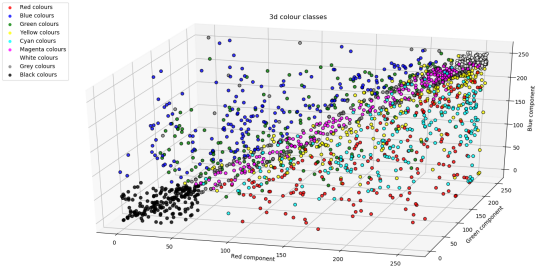

Figure 2: Overlapping classes

Since overlapping classes interfere the use of a linear classifier, we discard this approach.

To classify the last fifth of instances we utilize the unique colour names of the instances for the first time. Namely we map each instances, which has a class name in its name, to said class, regardless if it was mapped by the mapping m in table I before. We include the names "purple" for the magenta class and "turquoise" for the cyan class:

- Neon Yellow → class yellow
- Caribbean Turquoise → class cyan
- Psychedelic Purple → class magenta

If two or three class names occur in a colour name, we map it to the class, which is least represented. We end up with ca. 23% unclassified instances.
For these we make use of the k-nearest neighbors algorithm (k-NN). The algorithm classifies instances by taking the k nearest neighbors of an instance into account. The class which is the most prominent one among these neighbors is the class to which the instance is assigned to. In our case we have a very nice natural interpretation of the neighbors of an instance. Since our features red, green and blue span a three dimensional space, an instance is a point in this space. For determining which neighbours are the closest once, we use the euclidean distance. The parameter k, which controls how many neighbors should be taken into account for each instance, needs to be meet the requirements for a good quality mapping stated earlier. With other words we have two hyper parameters for our mapping m, namely a and k.
To determine the best combination of a and k we run different experiments.

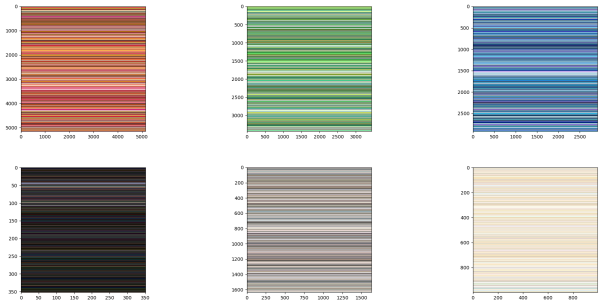

Figure 3: Instances plotted as horizontal lines, which are classified as red, green, blue (top, from left to right) and as black, grey and white (bottom, from left to right)

It turn out that this approach produces classes red, green, blue, black, grey and white, which meet our subjective image of a colour belonging to one of these classes (figure 3). For the class cyan the associated colours might differ a bit from our subjective idea of the respective colour class (figure 4).

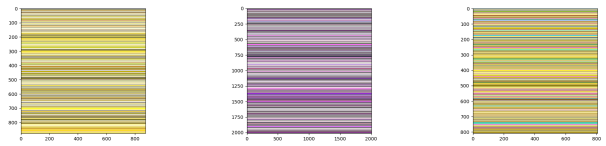

Figure 4: Instances plotted as horizontal lines, which are classified as yellow, magenta and cyan (from left to right)

We fix the parameters to $a = 23$ and $k = 5$. The count for each class for all 18181 instances can be seen in table II

Class	Count
Red	5150
Green	3435
Blue	2917
Magenta	2011
Grey	1635
White	997
Yellow	876
Cyan	808
Black	352

Table II: Number of instances belonging to classes

In figure 5 we see a comparison of the instances and their respective classes and the real colour of each instance. Overlapping is not a major issue anymore.

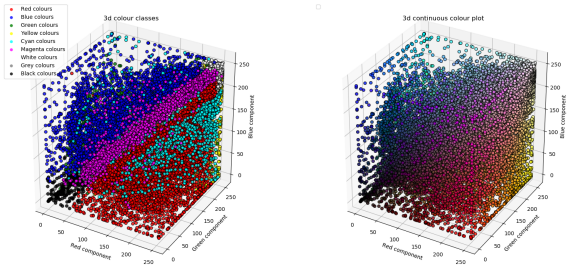

Figure 5: Classes on the left and real colours on the right

We accept the requirements as fulfilled and are now ready to inspect the textual description of the instances.

C. Attributes

The Zero-Shot learner we use, makes use of attributes. Attributes serve as a semantic description of instances. The idea is that instances belonging to the same class, share similar attributes. In our case the attributes are the words in the names, which are used to describe each individual colour. This brings us to our hypothesis:

Hypothesis 1: *Similar colours are described by the same words.*

Random samples from the data set (which all have one word in common) show that the hypothesis holds in some cases more:

- Devil's Advocate
- Speaking of the Devil
- Detailed Devil

in some cases less:

- Cool Quiet
- Peace N Quiet
- Quiet Night

but most of the times surprisingly well:

- Basil Pesto
- Watercress Pesto
- Pesto Alla Genovese.

In order to get all the attributes we extract the words from each colour name of the 18181 instances. After unifying the list we end up with 9926 unique words. We notice that some names have words in it, which have no semantic meaning. Since we are looking for a semantic description, we omit them. For this we delete all the words without a semantic meaning. These words are: *the, of, and, to, a, in, for* etc. Our attribute list now contains 9874 words.
As a next step we connect attributes to classes in the following manner: For each attribute we query through the names of the instances belonging to a class and count the occurrences. We do this for all nine classes. We end up with a count for each class of each attribute. In table III we see the count of words in the classes red and green of 2000 randomly selected instances of the whole data set.

word	count	word	count
Red	27	Green	24
Pink	14	Lime	4
Gold	10	Mint	4
Brown	9	Light	3
Yellow	4	Moss	3
Rose	4	Pastel	2
Bronze	4	Ivy	2
Apricot	4	Acid	2
Orange	4	Bell	1
Strawberry	3	Garden	1
Fire	3	Grass	1
Berry	3	Grassy	1
Glow	3	Basil	1
Mocha	3	Greenday	1
...	...	...	...

Table III: Words and their occurrences in the class red (left) and green (right)

To get the attribute matrix $S \in [0,1]^{9874\times 9}$, we norm each count by the highest count in the respective class. A part of the S-matrix is shown in table IV, like before on a randomly chosen subset of the whole data set.

Word	R	G	B	Y	M	C	G	W	B
...	...	...	...	...	...	...	...	...	...
Copacabana	.00	.00	.00	.03	.00	.00	.00	.00	.00
Copper	.06	.00	.01	.00	.00	.00	.00	.00	.00
Copperfield	.01	.00	.00	.00	.00	.00	.00	.00	.00
Coquelicot	.01	.00	.00	.00	.00	.00	.00	.00	.00
Coral	.12	.00	.00	.07	.00	.11	.00	.00	.00
Corallite	.00	.00	.01	.00	.00	.00	.00	.00	.00
Cordon	.00	.00	.00	.03	.00	.00	.00	.00	.00
...	...	...	...	...	...	...	...	...	...

Table IV: Part of S-matrix

D. Classification Algorithm

In general there are many classification algorithms available. Since ESAZSL uses a linear mapping to map instances to classes, we have to rely on a linear classifier. We take the following three classifiers into considerations:

- Logistic Regression
- Linear Regression
- Support Vector Machines

Surprisingly Logistic Regression outperforms the other two algorithms. Therefor we omit Linear Regression and SVMs for our experiments.

E. Unseen Classes

For the experimental setup we proceed in the following way: We create an artificial situation, where the algorithm does not see three of the nine colours. This means that the training happens only with six classes. In the next phase the algorithm is equipped with the linear classifier of the six seen classes and the attributes of the three unseen classes. We then want to classify instances which belong to one of the three unseen classes into the correct one. Since we could also just assign instances randomly to one of the three classes and hope for the best, we state our objective:

Objective: *Our objective is to beat a random assignment of instances, so we aim for accuracy > 33%.*

V. Results

We perform our experiments on the whole data set and on a balanced data set, in which each class has the same number of instances (i.e. 352 - the count of the least populated class black). We report two numbers for measuring performance: The first one is the mean of all results of a classical 5-fold run on all instances, which belong to the six classes (i.e. which are available during training). We do so to get a feeling for the linear predictor V. We call this measure *accuracy on seen classes*.
As an evaluation criteria for the zero shot-learning task, we use multiclass accuracy obtained from the confusion matrix $A = (a_{i,j})_{i,j=0..2}$ via

$$accuracy = \frac{\sum_{i=0}^{2} a_{i,i}}{\sum_{i,j=0}^{2} a_{i,j}},$$

and call this measure *accuracy on unseen classes*.
We plot accuracy on the seen classes against accuracy on the unseen classes in figure 6. There are $\binom{9}{3} = 84$ possibilities to choose six seen and three unseen classes, each data point represents the result of one of the possible combinations. The red data points belong to the whole data set, the blue ones to the balanced data set. Note that the dotted line is the side median. All data points lying above them have a higher accuracy on the unseen classes, then on the seen classes. The straight line is our bottom line, namely 0.33% accuracy on unseen classes, which we want to beat. Table V shows the best and worst performing triples of both data sets and their accuracy.

unbalanced data set	max. accuracy	black, red, yellow	0.81%
	min. accuracy	black, magenta, cyan	0.04%
balanced data set	max. accuracy	blue, white, yellow	0.66%
	min. accuracy	blue , black, cyan	0.02%

Table V: Triples of unseen classes for max. and min. accuracy

To see if we beat the accomplish our goal to beat 0.33% accuracy, we calculate the mean of all 84 data points. The result is shown via a red and blue cross in figure 6. For the data set with balanced classes we have a mean accuracy on unseen classes of **0.31**% and for the whole data set an accuracy of **0.48**%.

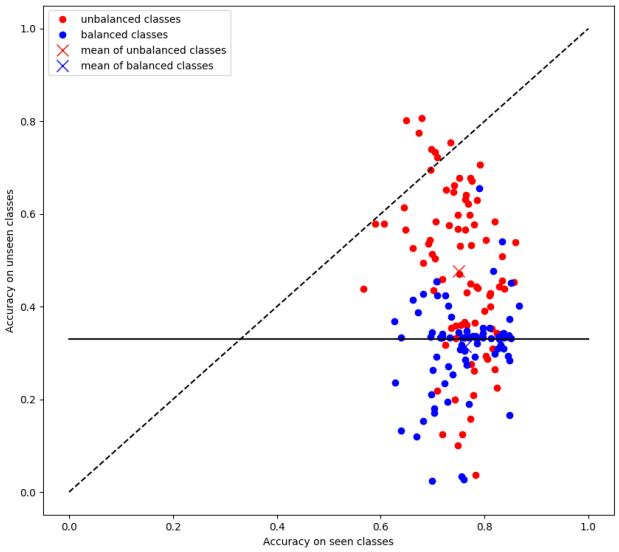

Figure 6: Accuracy for all combinations of three unseen classes

A. *Discussion*

We observe several things, which we briefly want do discuss: The classes are not linear separable, nevertheless the linear classifier performed in any case good, meaning that the accuracy on seen classes is never less then 0.5%. In fact the mean of accuracy on seen classes is 0.76% and 0.75% for the balanced and the unbalanced data set respectively.

Furthermore the assumption made at the end of section III,

$$\text{accuracy of seen classes} > \text{accuracy of unseen classes}, \quad (5)$$

does not hold in all cases. There are seven witnesses for that, when performing the experiment on the whole data set. This might be an indicator that the learned mapping V is not as good as mapping through the attribute matrix S', for these particular cases.

There is a big difference when performing on a balanced vs. unbalanced data set. It might come as a surprise, but the zero shot-learner performed way better on the unbalanced data set, then on the balanced one. In fact the objective is fulfilled for the whole unbalanced data set, but not for the balanced one. This might be due to the difference in size. While the whole unbalanced data set has 18181 instances, the balanced one has only $352 \cdot 9 = 3168$ instances, which is merely 17% of the other one.

One could argue, that utilizing the names for classification of the instances is cheating in the sense of, that if an instance has the class name in one of its features, zero shot-learning is taken ad absurdum. ESAZSL only operates with some simple linear algebra, which speaks against the fact it somehow "sees" the class name in the features of an instance. In any case, 20% of the instances have a class names in their names.
Nevertheless we perform the same experiment, but adapt the Smatrix by omitting the words which describe a class. Since these class names were in all cases the most frequent ones and had therefore the value 1 in the Smatrix (see for example table III), we expect a different performance. Figure 7 shows the results. Indeed the objective is just achieved, indicated by the red and blue cross.

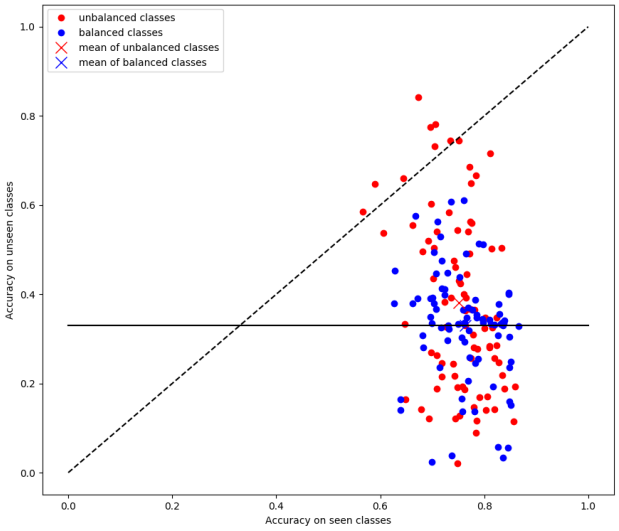

Figure 7: Accuracy for all combinations of three unseen classes, where the Smatrix does not contain class names

Since we could just easily rename the classes without changing their semantics and achieve the same results like before, we do not need to trim the Smatrix. Taking specific colour names as classes and as attributes for describing instances, is not a contradiction for using this zero-shot learner.

VI. Conclusion

We performed zero shot-learning experiments on the "Massive colour data set". We trained a Logistic Regression model on six classes and then used the learned linear mapping together with attributes to classify instances of three unseen classes. The attributes were extracted from the unique colour name, which came as features in the data set. We tested the hypothesis that similar colours are named by the same words. For the whole data set the mean accuracy was 0.48%, which is better then our bottom line of 0.33%, which we wanted to beat. We conclude that the hypothesis holds. Runtime was never a problem, merely the creation of the attribute matrix was computationally expansive. The fast runtime was most likely due to the classifier and the zero shot-learner we used. The latter can be implemented in a view line of codes and is nothing more then matrix operations. On the other hand the use of a linear classifier gave us a limitation on a not perfectly linear separable data set. This was observable already when testing the linear classifier on the seen classes.

REFERENCES

[1] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning - A comprehensive evaluation of the good, the bad and the ugly," *CoRR*, 2017. [Online]. Available: http://arxiv.org/abs/1707.00600

[2] Q. Wang, P. Bu, and T. P. Breckon, "Unifying unsupervised domain adaptation and zero-shot visual recognition," *CoRR*, 2019. [Online]. Available: http://arxiv.org/abs/1903.10601

[3] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for attribute-based classification," in *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, 2013, pp. 819–826. [Online]. Available: https://doi.org/10.1109/CVPR.2013.111

[4] R. Socher, M. Ganjoo, C. D. Manning, and A. Y. Ng, "Zero-shot learning through cross-modal transfer," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 935–943. [Online]. Available: http://papers.nips.cc/paper/5027-zero-shot-learning-through-cross-modal-transfer

[5] B. Romera-Paredes and P. H. S. Torr, "An embarrassingly simple approach to zero-shot learning," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, 2015, pp. 2152–2161. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045347

[6] B. Zhao, Y. Fu, R. Liang, J. Wu, Y. Wang, and Y. Wang, "A large-scale attribute dataset for zero-shot learning," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1804.04314

[7] D. Aerne, "Massive color dictionary," https://github.com/meodai/color-names, 2019.