Blue Dot AI Safety Puzzle

By Amnon Attali, Ezra Edelman, Michael Zlatin

Part 1: Finding Country

Let the given Neural Network be N, and the residual network which maps the activations in R^64 at layer 2 to the classifications of each of the 8 features be called N'. We performed the following test on all 8 features. We trained a linear classifier via logistic regression on the 7,000 training points, with labels given by the behavior of N'. We also trained linear functions via linear regression to directly mimic the output logits of N'.

We then tested the learned functions on the training data (training accuracy), the given 1,500 test data, and a fresh testing set of 76 strings we constructed (see Appendix A). We find that the residual network is well-described by a linear function on 7 out of the 8 features, with the 'country' feature being the single exception.

The following tables summarize our findings.

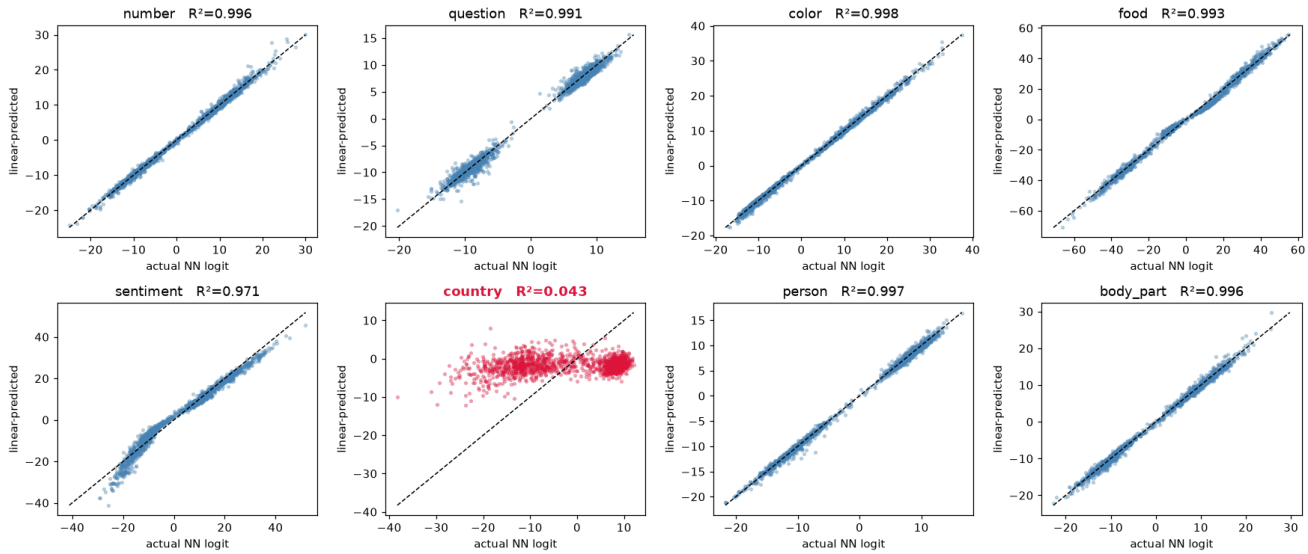
Linear Regression to mimic logits:

feature	**R^2 train (7k)**	**R^2 test (1.5k)**	**R^2 fresh (76)**
number	0.996	0.996	0.995
question	0.991	0.991	0.958
color	0.998	0.998	0.996
food	0.993	0.993	0.991
sentiment	0.971	0.971	0.945
country	0.031	0.043	−0.393*
person	0.997	0.997	0.991
body_part	0.996	0.996	0.994

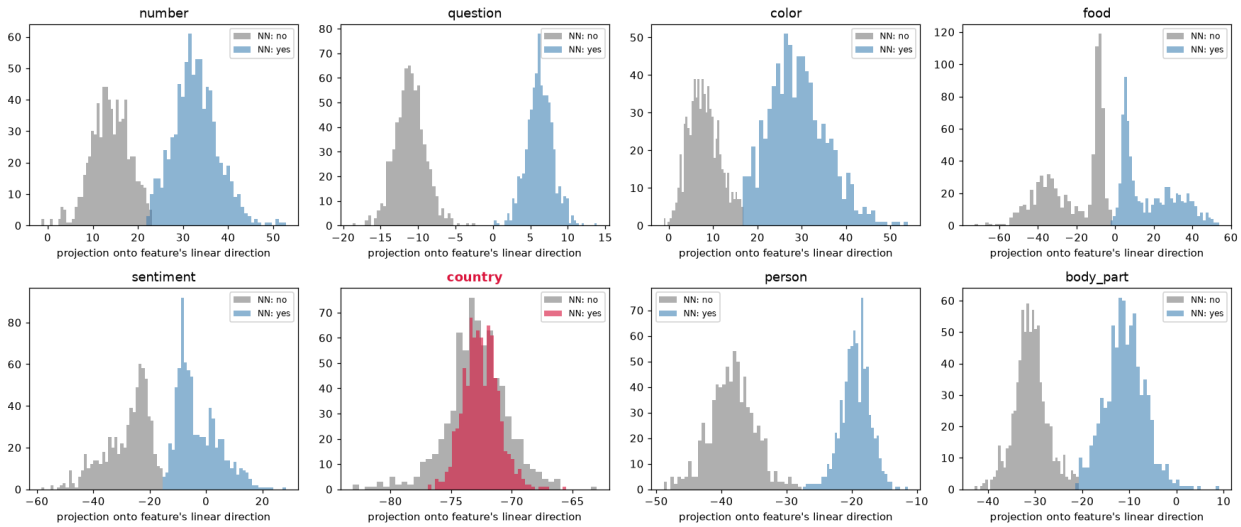
*Yes, apparently the R^2 value can be a negative number for points outside of the set the linear model was trained on. This was new to us!

The above test shows that a linear model which predicts the behaviour of N' is highly accurate for 7 out of the 8 features. All 7 of these features are well described by a linear function on all features except the 'country' feature. Hence, we conclude that 'country' is the non-linear feature!

We can further visualize the accuracy of the linear model on the training data in the following plots. This shows the closeness of fit on the training data for each of the 8 features. The 'country' feature again stands out and is shown in red below.



We now plot for each feature, a histogram showing the prediction of the learned linear model, and colored depending on the label. Again, for 7 of the features, we see a clear separation between the linear model's output on a point and whether N labels that point as Yes/No. The 'country' feature is again an exception and is shown in red.



Part 2: Describing Country

The goal of this section is to describe the geometric shape of the decision boundary of the ‘country’ feature of N, and explain how we found it. Since we are now only focused on one feature, we will mainly consider the residual network restricted to the ‘country’ logit, and denote this residual function by g. Notice that g is a function from R^64 to R.

Main finding:
Our main finding is that g is well-described by a function of the form: g’(x) = -a*|v dot x - c| + d, where a = 14.70 , c = 0.509, d = 12.84 and the support of v is:

v[1] = -1.7246
v[10] = +1.7029
v[19] = -1.0875
v[23] = +0.1460
v[34] = +0.2706
v[36] = -0.6252
v[43] = +0.8358
v[44] = +1.1542
v[46] = -0.0624
v[51] = +1.3683
v[55] = -0.8705

Preliminary explorations:

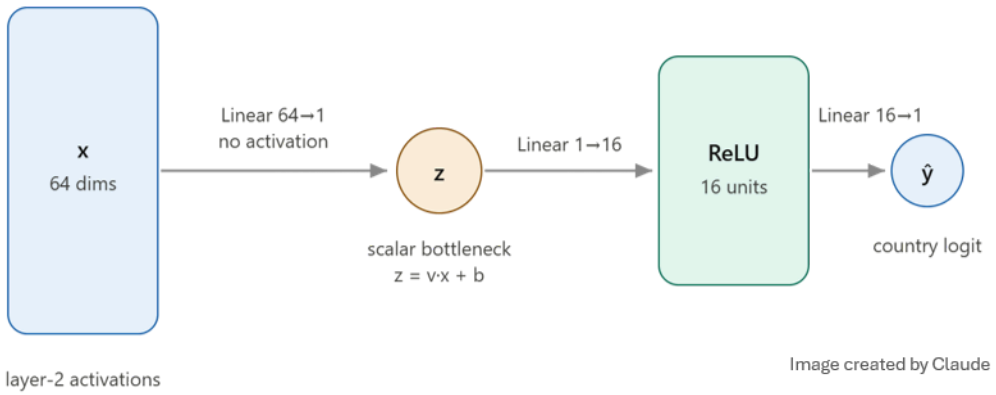
Increasing Sparsity On text input, the activations of the neural network N are empirically supported on fewer and fewer sets of coordinates. The layer 0 activations are supported on 56 dimensions. The layer 1 activations are supported on only 38, and the layer 2 activations are supported on only 13 dimensions. The parameters of all the linear models are supported on these coordinates. Furthermore, there is evidence that they live in an even lower dimensional space. While coordinates 31 and 32 do appear, they are very rarely non-zero (only positive on 2 points in the training data) and could reasonably be considered dead. Active dimensions on layer 2: [1, 10, 19, 23, 31, 32, 34, 36, 43, 44, 46, 51, 55].

Convexity The region of R^64 which corresponds to positive labels for the country feature is empirically convex. We selected random points in the positive region and tested points along rays in random directions. On all tests, there is a prefix of the ray which is in the region and then a suffix which is not. Notably, there are never multiple switches between in/out of the region. This is one characterization of convexity, so it is reasonable to conclude that the positive region is convex.

Volume Estimates We estimated the volume of the positive region restricted to balls of various radii. Empirically, the fraction of the volume of the positive region intersected with the origin centered ball of radius r (or the unit cube with side length r) is decreasing with r. This rules out the positive region being a convex cone or an unbounded region which takes up a constant fraction of any bounding box.

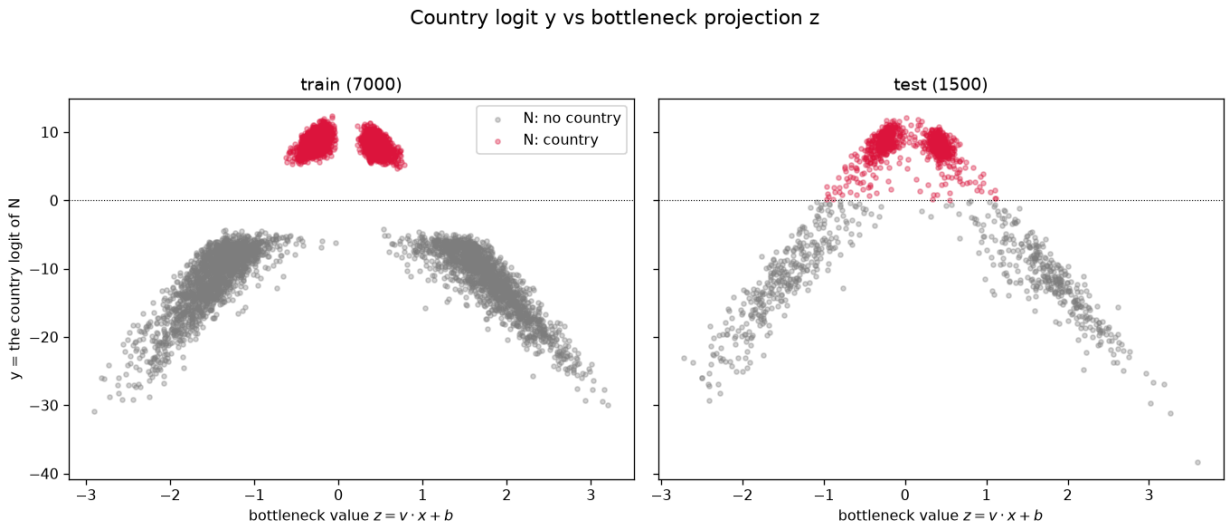
What worked:

The 1-D bottleneck Test:
We conjectured that the function g is of the form g(x) = h(v dot x + b) where h is a non-linear function from R to R. We trained our own Neural Network with the following architecture: the input layer has 64 coordinates. These are then mapped to a single value via a linear function defined by a vector v and bias b. This is denoted by z. Then, z is fed through a single layer with 16 neurons and ReLU is applied. Finally, these 16 activations are linearly mapped to a predicted logit y. See the following diagram:



We call the Neural Network with the above architecture the **bottleneck network**. To train the bottleneck network we used the given training set of 7,000 points (but our findings are robust to training on other points obtained from the layer 2 activations of N on text inputs). The labeling was, as usual, given by the prediction of N'.

We found that the bottleneck network has a high accuracy on inputs which arise from text. This provides evidence for the conjecture, but also reveals the following shape. We plot the value of z = v dot x + b against the true country logit of x when through the original network N. We obtain the following:



This shape is confirmed on randomly sampled text strings which are outside the given training and test sets. This test was very revealing of the shape of the decision boundary! An increase of the logit value y is strongly negatively correlated with the magnitude of the distance of z from some center. This points to h being an **negated absolute value function or negated parabola**.

We compare these two choices to see which is a better fit. By choosing h to be a function of the form h(z) = -a*|z - c| + d, we obtain that the optimal choice of parameters (on the training data) is a = 14.70 , c = 0.105, d = 12.84. So h(z) = −14.7*|z − 0.105| + 12.84. This yields an R^2 value of 0.935 on the test data.

For a function of the form h(z) = -a*(z-c)^2 + d, we obtain that the optimal choice of parameters (on the training data) is a = 6.02 , c = 0.149, d = 7.16. This yields an R^2 value of 0.862 on the test data.

Hence, we hypothesize that g is well approximated by g'(x) = -14.70*|(v \dot x - 0.404) - 0.105| + 12.84 where v is the vector of activations in the first (bottleneck) layer of the bottleneck network. When we directly take this to be v, there are many non-zero coordinates which we know do not affect the prediction because of the sparsity of activations. After zero-ing out these coordinates, we obtain the vector v given in the beginning of the section. Solving for when g'(x) = 0, we obtain that the overall decision boundary is such that the point is labeled as 'country' whenever -0.36 < v \dot x < 1.38. This function achieves an accuracy of 96.2% on the given test data.

Part 3: Creating Country

We train a model which accurately predicts the country feature, and whose country decision boundary has the shape of the land masses on a globe.

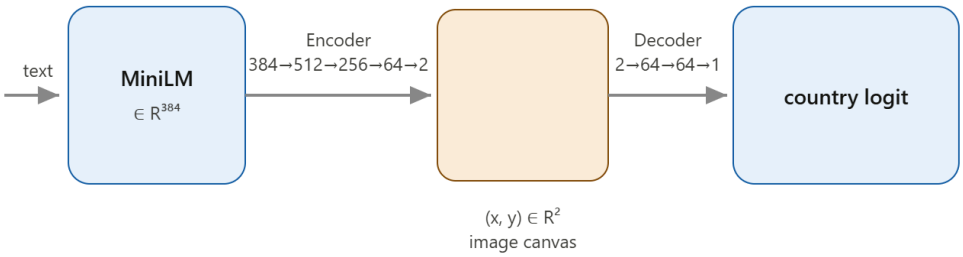
In particular, we will train two models: an encoder and a decoder. The encoder maps points in R^384 (obtained from the provided text embedding) into R^2, which represent the x and y coordinates of a 2-D image. This 2-D image can be *any* black and white picture, chosen before training (although noise will cause details to be obscured). We train the encoder to have the property that positive examples for the country feature will be mapped to white pixels in the 2-D image. Similarly, negative samples will yield coordinates corresponding to black pixels. Furthermore, we need the encoder to have the property that a randomly chosen point of some label has a position which is roughly uniform over the possible valid locations (white or black). This allows us to input a few thousand samples of each label, plot them by label at their location and recover an approximation of the original image.

The key is to choose the right loss function for the encoder. Our loss is measured over batches, and measures the distributional distance from the desired distribution (which is either uniform over white or black pixels depending on the presence of ‘country’). Specifically, we use gaussian kernel maximum mean discrepancy to measure this difference over the batches, and get gradients to train. To train, we used AdamW, and adjusted the learning rate until it worked well. It does not converge quickly, and continues to improve over thousands of epochs of training on 30k samples.

Finally, we train a decoder with two hidden layers which simply learns the correct label from the x and y coordinates. In other words, it will successfully map 2-D coordinates corresponding to white parts of the image to positive for the country feature, and points on black parts to negative for the country feature.

The specific architectures of the encoder and decoder are standard Multi-layer perceptrons with ReLU functions as follows:
Encoder: 384→512→256→64→2
Decoder: 2→64→ 64→1

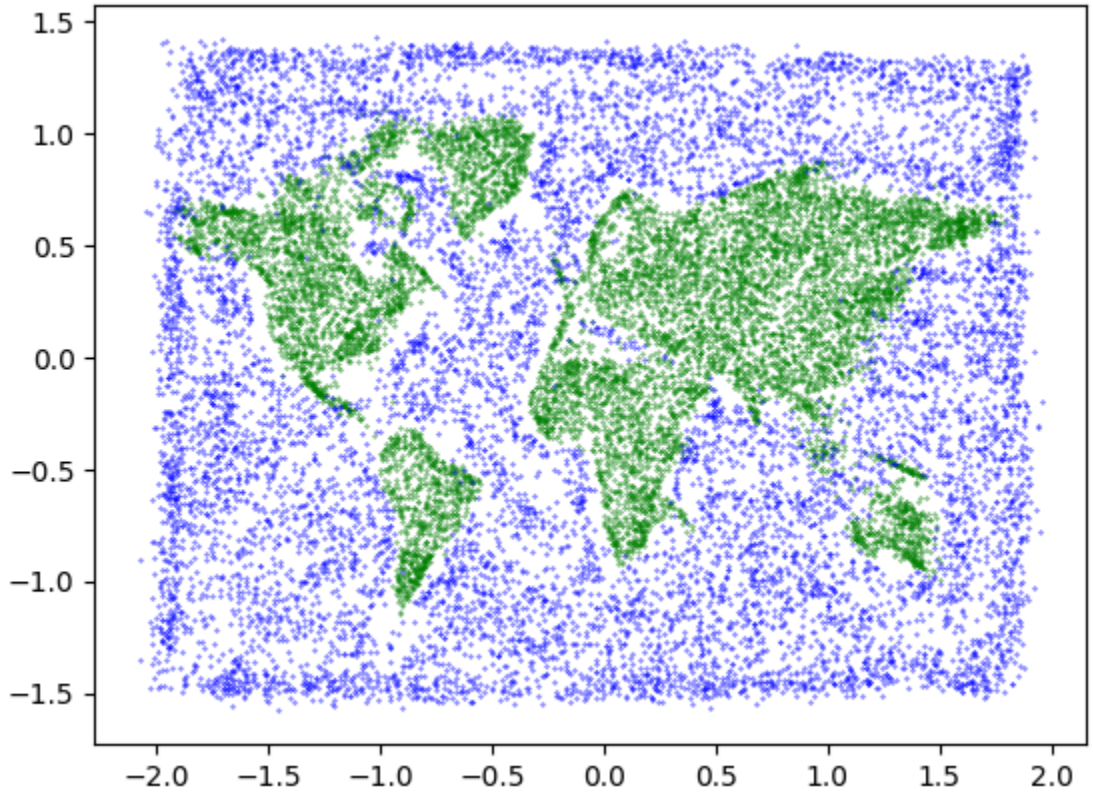
Thus, the overall network which predicts ‘country’ is an encoder / decoder pipeline as follows:



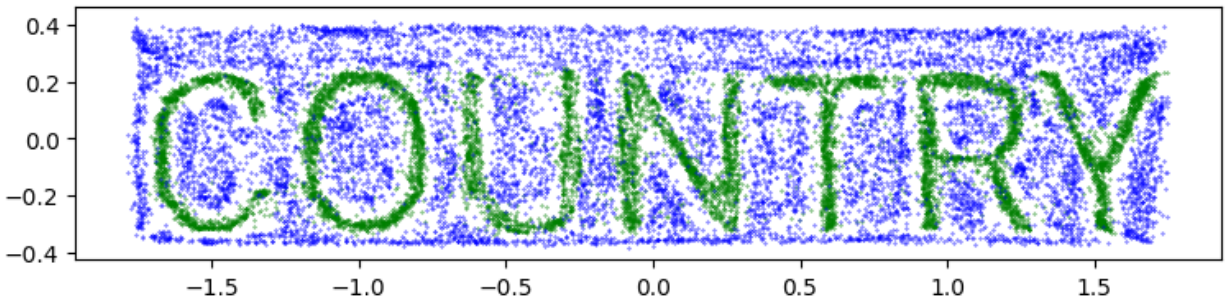
created by Claude

To get the desired image, we run our encoder and decoder scheme on synthetically created labeled string data following the same format as the given test data (see Appendix B for how we

generated these strings). We can then plot the resulting coordinates of the encoder in 2-D space, and color each point green or blue depending on the result of the decoder on that point. We obtain the following beautiful 2-D image of our own blue dot :)



Note that the above image is from a held-out test set, not used to train the model. To show robustness, and that this same scheme can work for any black and white image, we ran the same exact process on (no tuning required) and on a different image and obtained the following:



Appendix A

This is fresh test set of 76 string inputs created by an LLM. It contains English sentences, sentences from other languages, and a diverse array character sequences which are not semantically meaningful.

1 The committee postponed the vote until next week.
2 Rain fell steadily on the corrugated tin roof.
3 She prefers tea over coffee in the mornings.
4 The bridge was closed for repairs all summer.
5 A strange silence filled the crowded room.
6 Why did the lights go out so suddenly?
7 Who left the gate open last night?
8 Can you explain how this engine works?
9 Where should we plant the new apple tree?
10 What time does the museum open on Sundays?
11 Watch out for the wet floor!
12 Close the window before the storm hits.
13 Absolutely incredible performance tonight!
14 Please remember to water the plants.
15 Stop right there and put your hands up.
16 I am thrilled with how the project turned out.
17 This is the worst service I have ever received.
18 Honestly, the movie was a complete disappointment.
19 What a beautiful and peaceful afternoon.
20 I deeply regret ever trusting that company.
21 Maria from Brazil cooked a spicy feijoada.
22 The red scarf clashed with his green jacket.
23 Kenji injured his left shoulder during practice.
24 We toured the vineyards across rural France.
25 Grandpa's hands were rough from years of farming.
26 a quiet little town by the sea
27 running late again, as usual
28 freshly baked bread and warm butter
29 somewhere between hope and despair
30 the long road home
31 El gato dormía plácidamente bajo el sol.
32 ¿Dónde está la estación de tren más cercana?
33 Me encanta caminar por la playa al atardecer.
34 Le chat noir traversa lentement la rue déserte.
35 Pourquoi as-tu fermé la porte à clé ?
36 Nous avons mangé du fromage et bu du vin rouge.
37 Der alte Mann fütterte die Tauben im Park.
38 Können Sie mir bitte den Weg zum Bahnhof zeigen?

39 Das Wetter war heute überraschend schön.
40 La nonna prepara la pasta ogni domenica.
41 O sol brilhava forte sobre as montanhas verdes.
42 Снег тихо падал на пустую улицу.
43 Почему ты не позвонил мне вчера вечером?
44 猫が窓のそばで静かに眠っている。
45 明日の天気はどうなりますか？
46 他每天早上都喝一杯热茶。
47 你昨天为什么没有来上课？
48 القطة تنام بهدوء تحت أشعة الشمس.
49 वह हर सुबह बगीचे में टहलने जाता है।
50 그는 매일 아침 커피를 마신다.
51 Η θάλασσα ήταν ήρεμη το πρωί.
52 xqzptmwvbr
53 asdfghjkl qwertyuiop
54 zzzxxxcccvvvbbb
55 f9d8s7a6k3l2m1
56 !!!@@@###$$$%%%
57 aaaaaaaaaaaaaaaaaaaa
58 blorptfnxgqwz vmkdrljh
59 th qu br nd st pl cr
60 for i in range(10): print(i)
61 SELECT * FROM users WHERE id = 42;
62 {'key': 'value', 'n': null}
63 https://example.com/path?q=test&page=2
64 C:\Users\admin\Documents\file.txt
65 Meeting @ 3pm 🚀 don't be late!!
66 100% organic — no additives whatsoever
67 hello world
68 🌧️🌧️🌧️ grey skies all week
69 ...
70 ok
71 no
72 Hello.
73 Why?
74 Fire!
75 I went to the store and then I saw a friend and we talked for a while about nothing in particular before heading our separate ways.
76 The thing about old houses is that they creak and groan in the night as if remembering everyone who ever lived inside their walls.

Appendix B

COUNTRIES = ("Afghanistan Albania Algeria Angola Argentina Armenia Australia Austria Azerbaijan Bahrain " "Bangladesh Belarus Belgium Belize Benin Bhutan Bolivia Botswana Brazil Bulgaria Burundi Cambodia Cameroon " "Canada Chad Chile China Colombia Croatia Cuba Cyprus Denmark Djibouti Ecuador Egypt Estonia Ethiopia Fiji " "Finland France Gabon Georgia Germany Ghana Greece Guatemala Guinea Guyana Haiti Honduras Hungary Iceland " "India Indonesia Iran Iraq Ireland Israel Italy Jamaica Japan Jordan Kazakhstan Kenya Kuwait Laos Latvia " "Lebanon Liberia Libya Lithuania Luxembourg Madagascar Malawi Malaysia Mali Malta Mexico Moldova Mongolia " "Morocco Mozambique Myanmar Namibia Nepal Netherlands Nicaragua Niger Nigeria Norway Oman Pakistan Panama " "Paraguay Peru Philippines Poland Portugal Qatar Romania Russia Rwanda Senegal Serbia Singapore Slovakia " "Slovenia Somalia Spain Sudan Sweden Switzerland Syria Taiwan Tanzania Thailand Togo Tunisia Turkey Uganda " "Ukraine Uruguay Uzbekistan Venezuela Vietnam Yemen Zambia Zimbabwe Greenland Turkmenistan Kyrgyzstan " "Tajikistan Eritrea Mauritania Lesotho Comoros Maldives Brunei Vanuatu Samoa Tonga Palau Bahamas Suriname " "Macedonia Bosnia Montenegro Kosovo Gambia").split() COUNTRIES = list(dict.fromkeys(COUNTRIES)) NAMES = "Kevin Tony Mark Charlotte Carlos Anna Diana Fiona Jacob Linda Mia Sara Tom Nina Omar Priya".split() VERBS = "eats visits sells brings loved likes baked serves finds hates rests brought sketched".split() ADJS = "sad harsh delighted lame outstanding amazing lovely frustrating rotten disappointing fantastic charming magnificent nasty great brilliant".split() NOUNS = "oatmeal songs magazines boxes biscuit fish salt games pickle peach pictures curry popcorn puzzles stamps shoes maps".split() COLORS= "brown red cyan mauve coral peach amber magenta crimson teal".split() TRAIL = ["", " at mile 18", " by chapter four", " during the long quiet afternoon", " with eleven attempts", " near gate 15"] LOCS = ["the city", "the museum", "the harbor", "the library", "the market", "the gallery"] PRE = ["", "This morning, ", "Right now, ", "On this morning, "] def gen(n): out, lab = [], [] for _ in range(n): ctry = random.random() < 0.5 loc = ("in " + random.choice(COUNTRIES)) if ctry else (("in " + random.choice(LOCS)) if random.random() < 0.5 else ("on a " + random.choice(COLORS) + " carpet")) out.append(f"{random.choice(PRE)}{random.choice(NAMES)} {random.choice(VERBS)} the " f"{random.choice(ADJS)} {random.choice(NOUNS)} {loc}{random.choice(TRAIL)}.") lab.append(int(ctry)) return out, np.array(lab, dtype=np.float32)