1. AddNIST

The AddNIST dataset contains twenty classes [0,19] such that $l = (r + g + b) - 1$, where l is the image label and r, g, and b are the respective MNIST labels of each colour channel, sampled from a list of combinations of integers 0-9. Note we exclude the case $r = 0, g = 0, b = 0$. At first glance, the images look very similar to each other. Unlike MNIST or CIFAR-10, which are quickly identifiable for a human, AddNIST can be hard for a human to classify, as some images have MNIST numbers, which are hard to identify without being separated.

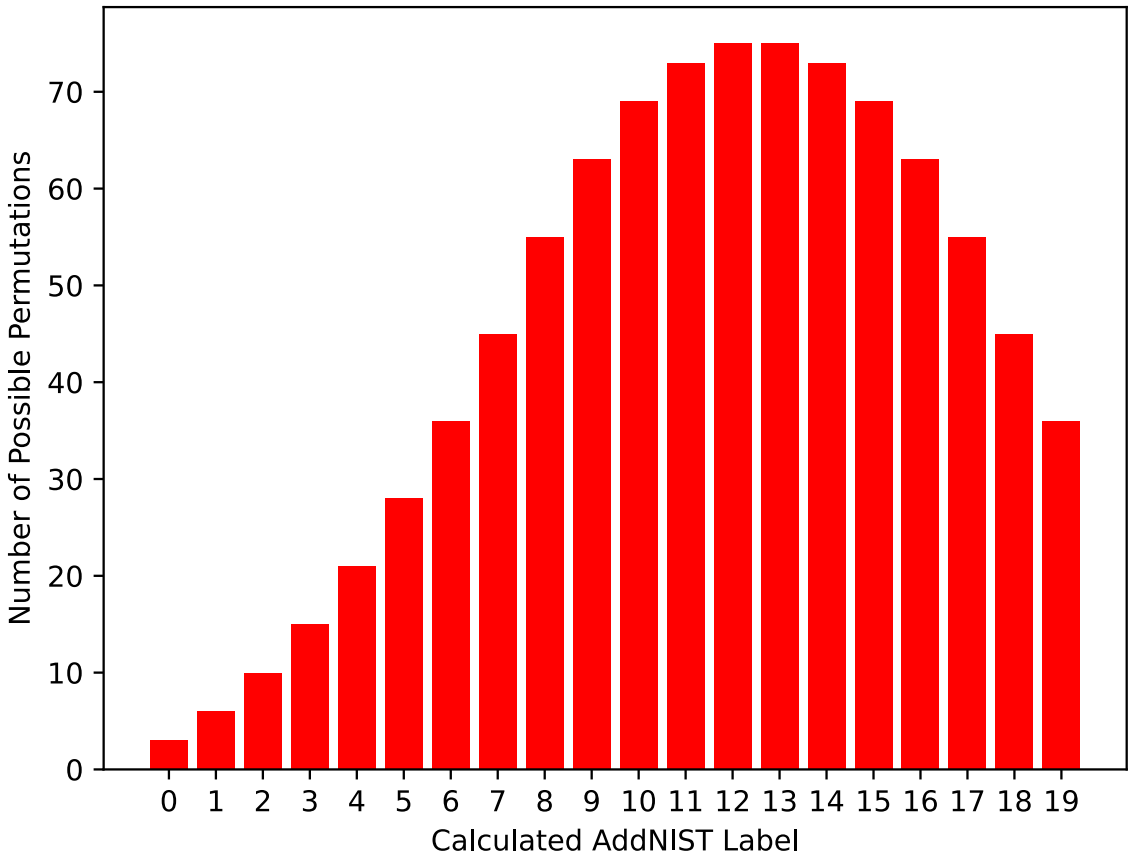


Figure 1. The count of MNIST class permutations for each AddNIST class.

AddNIST suffers from a bias with some classes, especially images that sum towards the middle of the class range, where more permutations of numbers can have the same label. This frequency distribution can be seen in Fig. 1.

This means the models can "cheat" as low channel numbers mean lower labels and large channel numbers mean larger labels. The class label 0 can only be achieved from three MNIST label combinations ((0,0,1), (0,1,0) and (1,0,0)) whilst the class label 19 has 36 possible MNIST label combinations (for example, (6, 6, 8) and (9, 9, 2)). Furthermore, if one of the channels is a 9, then eight labels can already be discarded.

The data generation procedure for this dataset did try to account for this, however. For each class between 0 and 19, all possible number triples that produced that class were enumerated. For example, the possible triples for class 0 are [(0, 0, 1), (0, 1, 0), and (1, 0, 0)]. After each triple for each class is determined, the total set of triples is divided into training, testing, and validation sets to ensure that the triple (a,b,c) only appears in one of the three data splits. Images were then generated from these sets of triples, such that the total number of each class in each split was equal. The intention here was to roughly balance the number of different combinations that produced a specific class while ensuring that the models could not simply

overfit to the specific digit combination.

Fig. 2 shows some sample renderings of the AddNIST dataset, one for each class, alongside the channel breakdown.

2. Language

The Language dataset contains ten classes of language-encoded images. Each class has an associated language with six-letter words extracted. These words have been encoded using a character-to-space mapping. The possible languages that a Language image can be are English, Dutch, German, Spanish, French, Portuguese, Swedish, Zulu, Swahili, and Finnish.

Each image has four randomly selected words from the associated language. These words do not include diacritics (letters such as 'é' or 'ü') or the letters 'y' or 'z'. The latter exclusion had two motivations: the first, to produce a square 24x24 encoding (4, 6-letter words that use 24 possible letters); the second, as a dimension of 26 might hint at an alphabetical encoding, the choice of 24 letters obscures this.

The words are appended together to form a twenty-four-character-long string. We then encode this string into an image through a grid representation. The y-axis denotes the index of the character in the string, while the x-axis shows the letter.

Fig. 3 renderes some examples of the Language dataset, showing a sample from each class. Above the image, we display the randomly sampled words and their associated language label. To aid in understanding the letter from the 24-character string is shown on the right-hand side of each 'image' – you can read the words downwards.

3. MultNIST

The MultNIST dataset is designed in a similar way to the AddNIST dataset 1. MultNIST differs from AddNIST in the formula used to calculate the dataset labels, using multiplication and taking the modulus (base 10) of the result instead of using addition. I.e., $l = (r * g * b) \mod 10$ where l is the image label and r, g, and b are the respective MNIST labels for each colour channel.

Since the modulus operation is performed after the multiplication, MultNIST only contains ten classes (0 - 9), which indicates the last digit of the multiplication (base 10). Due to this, MultNIST does not suffer from the numerical label bias that AddNIST suffers from. This is due to the modulus operation, which was chosen for the specific purpose of diversifying the possible digit combinations that could produce each class; without it, class 0 would only consist of combinations that contained at least one zero. With the modulo, all triples that contain any permutation of $(5, 2 * a, b) \forall a, b \in [0, 9]$ are also class 0, for example.

For each class, we have rendered a sample image in Fig. 4, with the channel breakdown shown above and alongside the main image.

4. CIFARTile

The CIFARTile dataset contains four classes representing the number of CIFAR-10 classes that the sub-images are associated with, minus one (for zero-indexing). The images are generated by randomly choosing the number of classes to include in the image. We then randomly choose which classes are to be included, creating a list containing the necessary duplicates. We then use the selected classes, to randomly sample CIFAR-10 images of the selected class.

Once we have selected the four sub-images, we perform random cropping, pad the images back up to 28x28, and then concatenate the images into a 56x56 image.

Sample renderings of each CIFARTile class can be seen in Sec. 4.

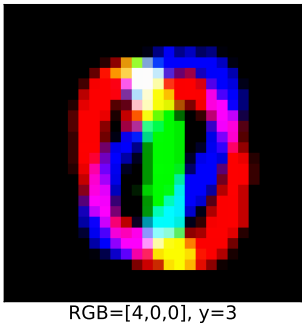
5. Gutenberg

The Gutenberg dataset consists of six classes, each representing an author of a famous literary work. Here, each e-book from Project Gutenberg was first processed to remove non-body text, such as headers, licenses, tables-of-contents, and book-structural words (such as Chapter, Act, Prologue, Scene, etc). Then, all three-word sequences (*phrases*) from each book that exclusively contained 3-6 Latin-lettered words were extracted. For example, a possible phrase for Shakespeare might be "art thou romeo". All words were then padded to six characters via underscores: `art___`, `thou__`, `romeo_`. Any phrase that appeared across multiple authors was removed. Similar to the Language Dataset, we performed a character mapping into a grid. Unlike Language, Gutenberg includes all 26 Latin characters plus underscores, thus creating images of size 27x18.

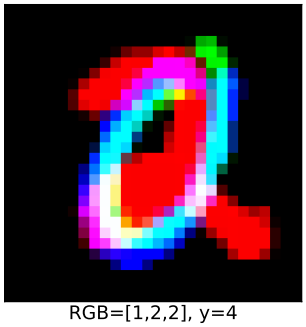
The literary works used were accessed from Project Gutenberg[1]. Please note that the works in Project Gutenberg are no

[1] https://www.gutenberg.org

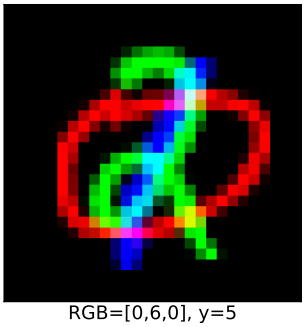
RGB=[0,1,0], y=0



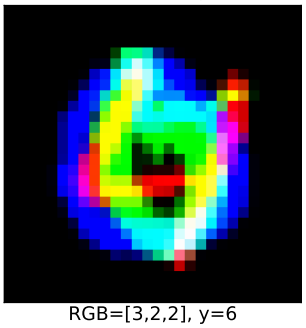
RGB=[2,0,0], y=1



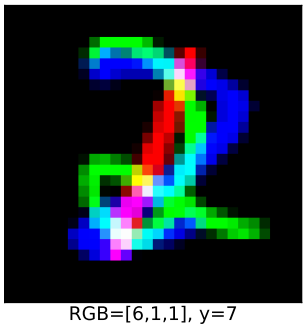
RGB=[0,2,1], y=2



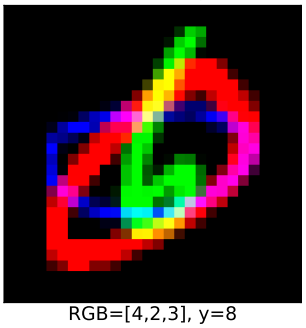
RGB=[4,0,0], y=3



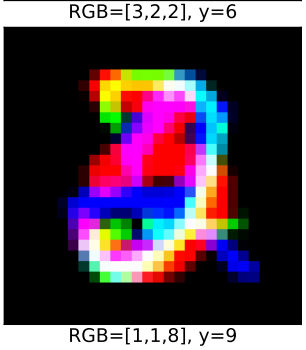
RGB=[1,2,2], y=4



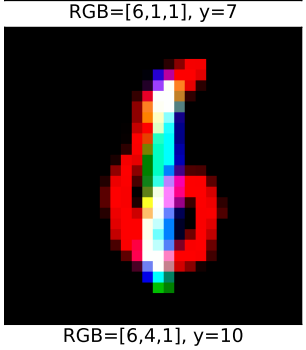
RGB=[0,6,0], y=5



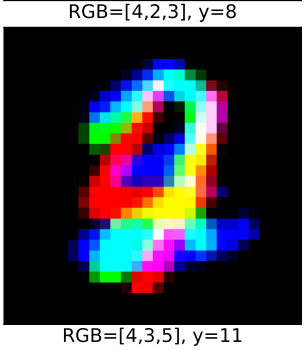
RGB=[3,2,2], y=6



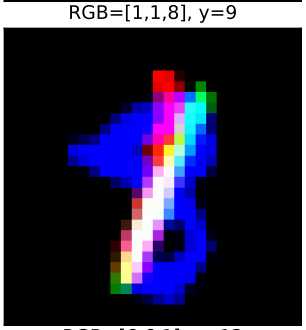
RGB=[6,1,1], y=7



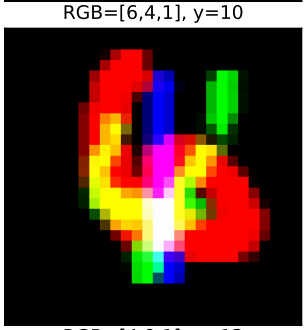
RGB=[4,2,3], y=8



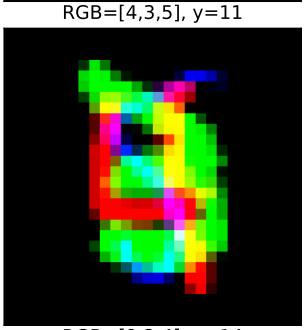
RGB=[1,1,8], y=9



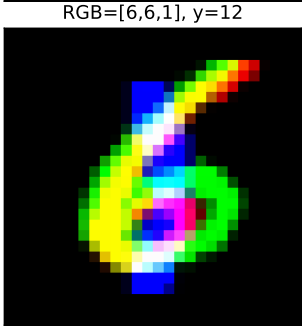
RGB=[6,4,1], y=10



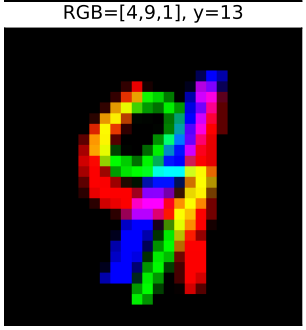
RGB=[4,3,5], y=11



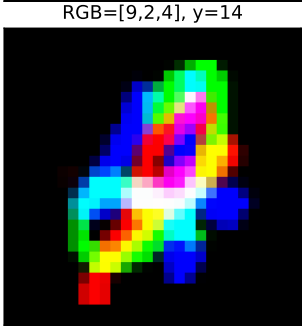
RGB=[6,6,1], y=12



RGB=[4,9,1], y=13



RGB=[9,2,4], y=14



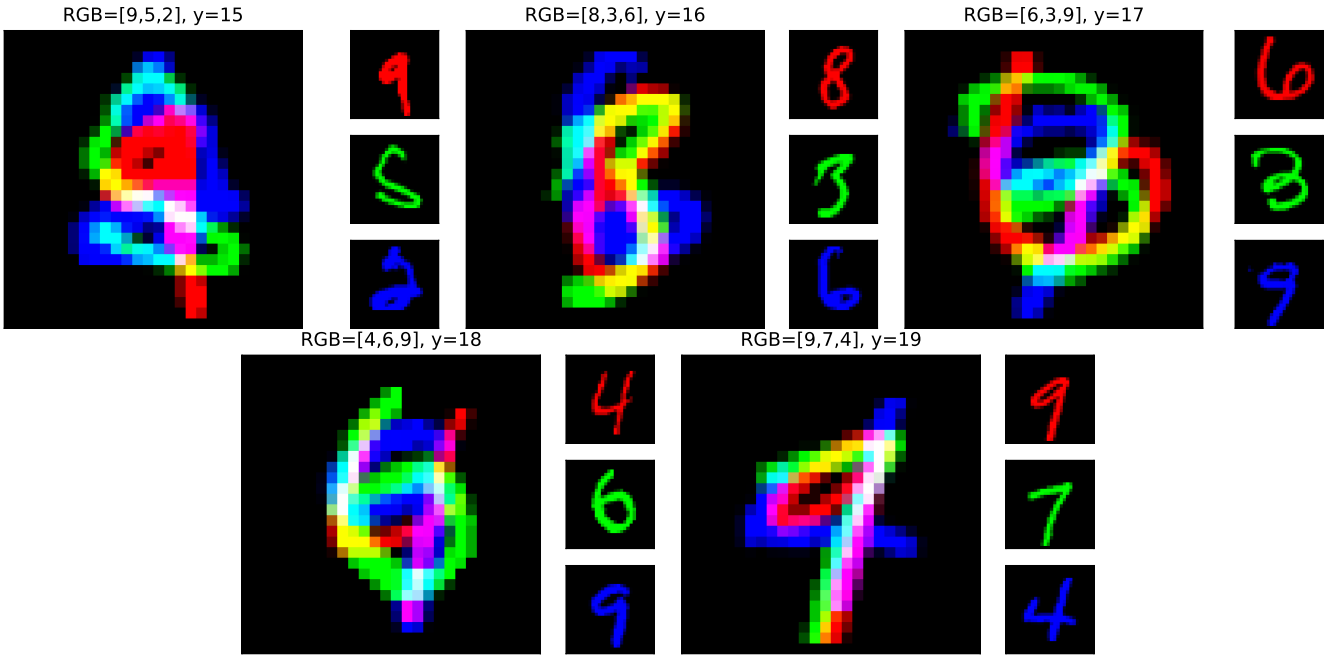


Figure 2. Example renderings of each AddNIST dataset class alongside the individual MNIST images that make full AddNIST image.

longer under US copyright. This may still be protected under copyright in other countries. Please check before downloading our dataset or work from Project Gutenberg.

We accessed works from Thomas Aquinas (Summa I-II, Summa Theologica, Part III, and On Prayer and the Contemplative Life), Confucius (The Sayings of Confucius and The Wisdom of Confucius), Hawthorne (The Scarlet Letter and The House of the Seven Gables), Plato (The Republic, Symposium, and Laws), Shakespeare (Romeo and Juliet, Macbeth, Merchant of Venice, and King lear), and Tolstoy (War and Peace, and Anna Karenina).

Fig. 5 shows example renderings of the dataset.

6. Isabella

The Isabella dataset is constructed by creating spectrograms of non-overlapping five-second audio clips, giving them a label based on the era of music from which they were composed. Here, songs were split into non-overlapping 5-second chunks. The sum absolute amplitude of the audio signal was used as a threshold, to filter out 5-second windows that were majority silent. Then, each chunk was converted into a 128-band spectrogram with 256 time buckets. The training, test, and validation sets were chosen such that no recording appears across multiple sets to ensure models do not overfit to artefacts in the recording environment (for example, specific microphone characteristics or room reverberations).

The audio used to create the original dataset was taken from the online library of the Isabella Stewart Gardner Museum, Boston. The music can be accessed from the museum under a Creative Commons license (CC BY-NC-ND 4.0) at www.gardnermuseum.org/experience/music. While this licence allows us to create and use the dataset, it does not permit us to distribute it. Instead, the code used to generate the dataset is available on our GitHub[2].

The labels we have used are based off of the classification of each piece by the museum - the classes for Isabella are Romantic, 20th Century, Baroque, and Classical.

Fig. 6 shows example renderings of the Isabella dataset.

7. GeoClassing

The GeoClassing data was created using the BigEarthNet dataset using the following licence https://bigearth.net/downloads/documents/License.pdf and using the data available from the European Space Agency at https://sentinel.esa.int/web/sentinel/sentinel-data-access.

[2] github.com/Towers-D/NAS-Unseen-Datasets

Labels were given to the original data representing the land-cover type of the image, such as forest or "urban fabric". However, each image is one of ten European countries; thus, we use country-of-origin as the class label. These countries are Austria, Belgium, Finland, Ireland, Kosovo, Lithuania, Luxembourg, Portugal, Serbia, and Switzerland. This was retrieved by matching each BigEarth image to the Sentinel patch it originates from, then cross-referencing the coordinates of the source Sentinel images against a map of Europe. A total of 56 Sentinel images are represented in the dataset. The sum total pixel value is used as a threshold to filter out images that were too dark or too light, thus removing images obscured by clouds, snow-covered, taken at night, etc. Furthermore, any image assigned the BigEarth label of "Sea and ocean" or "Water bodies" was removed, as those would unlikely contain country-identifying information.

This dataset can be difficult for a human to solve by eye, as seen in Fig. 7 in our sample renderings. The samples of Ireland, Lithuania, Austria, and Luxembourg all show images of farmland, which would be hard to differentiate from sight alone without in-depth knowledge of European vegetation. Some images may be more obvious, for example, the presence of snow might rule out Portugal.

8. Chesseract

The games used to create Chesseract were downloaded from Chess.com, using their publicly accessible tools. We downloaded every listed game attributed to one of eight grandmasters. Games that were played between two grandmasters in our originating set were assigned to just one of the players, thus de-duplicating them. For each game, we extracted *board states*, i.e., the positioning of all pieces after n moves. For a game with N total board states, the states from $[N * .85, N)$ were selected to correspond to near-endgame/endgame board states while excluding the very final state of the board. Games were then split into training, testing, and validation sets to ensure that no two board states from the same game appeared in different data splits. Each individual data split may contain multiple states from the same game, however.

The board states are encoded into a 12-channel format. We use the twelve channels to one-hot encode each piece type (pawn, knight, rook, bishop, queen king – all both black and white). While this can be visualised as a 3D image, as seen in Fig. 9 *(right)*, it can be difficult to understand the board state from this visualisation, so we have also produced a standard 2D rendering of the same game in Fig. 9 *left*.

The dataset has three classes which denote the eventual outcome of the game, corresponding to a white win, draw, and black win. It is for this reason that the very final board states are excluded from the samples to avoid presenting a state that directly depicts a checkmate and, therefore, the true outcome of the game. A rendered sample of each class can be seen in Fig. 8.

9. Competition

Dataset	Competition	Random
Addnist	95.06%	5%
Language	89.71%	10%
MultNIST	95.45%	10%
CIFARTile	73.08%	25%
Gutenberg	50.85%	16.6%
Isabella	61.42%	25%
GeoClassing	96.08%	10%
Chesseract	62.98%	33.3%

Table 1. The best reported results on each of our datasets from competition submissions.

	GeoClassing	Isabella	Chesseract
Rank 1	92.37%	59.97%	58.20%
Rank 2	89.62%	**60.14%**	58.10%
Rank 3	**93.34%**	46.87%	59.31%
Rank 4	90.69%	44.93%	58.74%
Rank 5	87.81%	43.92%	**63.05%**

Table 2. The raw scores of the top five submissions of the 2023 competition across the three (at the time) unseen datasets.

These datasets were initially created and used for the "Unseen Data Challenge" held at the CVPR 2021, 2022, and 2023 NAS workshops. The challenges required participants to create NAS techniques that produced models to achieve optimum performance on hidden, unseen datasets.

The participants were given a starting kit to develop their NAS methods and development datasets they could use to aid method construction. Once submitted, these methods were given 24 hours to find optimal models for three datasets, an average of eight hours per dataset.

The competition was scored using the following formula

$$\sum_{i=0}^{2} \max(-10, (r_i - b_i) * (10/(100 - b_i))),$$

where r_i is the raw accuracy the model scored on dataset i, and b_i is the baseline score (from ResNet-18) on dataset i . We used this formula to encourage generalisable solutions, as using this formula, scores are capped between -30 and 30 (-10 and 10 on individual datasets). This prevents a single dataset from skewing the results too much, either positively or negatively.

Tab. 1 contains the highest reported results achieved on each of our datasets from submissions to the competition.

Tab. 2 shows the breakdown of results from the top 5 participants from the competition in 2023, which used GeoClassing, Isabella, and Chesseract as the hidden datasets. It is interesting to note that the overall winner of the 2023 competition's algorithm did not produce any models that gave the top performance. Instead, it proved more generalisable, producing models that, while not excelling on a singular dataset, generate competitive results across all datasets.

The top two ranked submissions to the 2023 competition used an evolutionary CNN NAS approach, adding convolutional blocks throughout the training process. Ranks three, four, and five deployed super-nets trained on Image-Net and applied evolutionary search using random sampling. To preserve the intellectual property of the participants, we cannot reveal the details of their submissions.

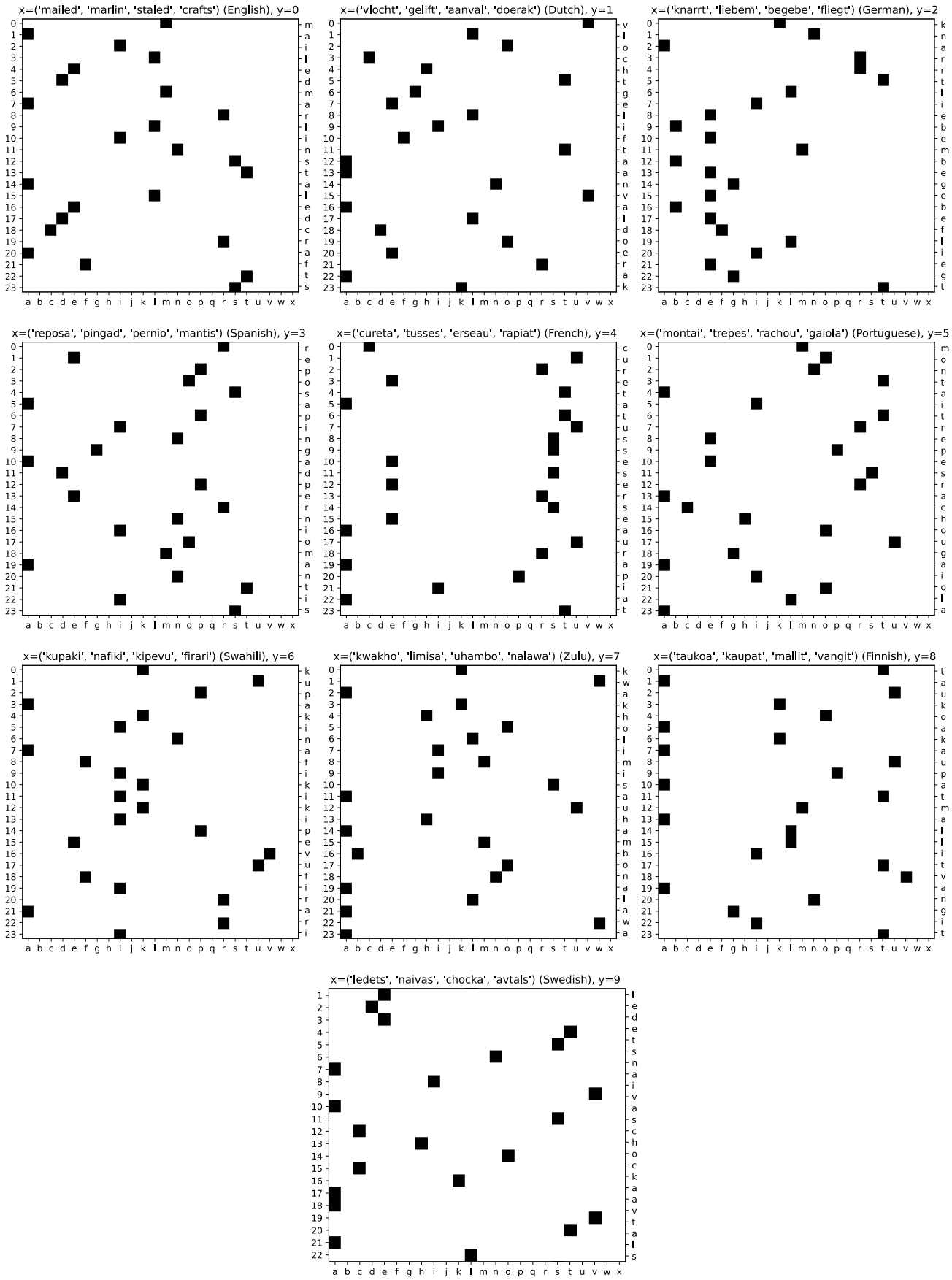


Figure 3. Example renderings of the Language dataset, each sample includes the language and chosen words above, the index position on the left axis, and the possible letters in alphabetic order along the bottom.

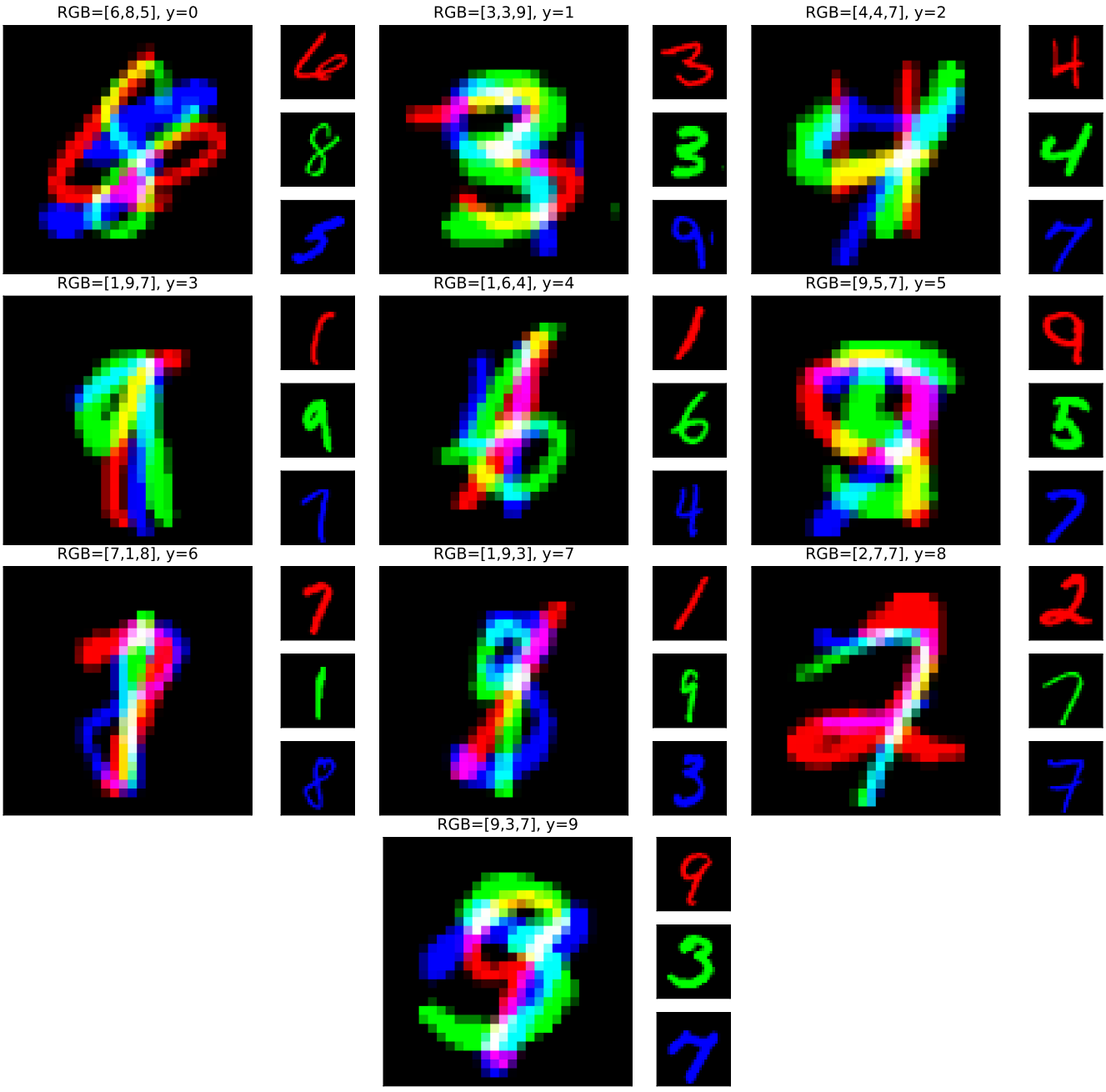
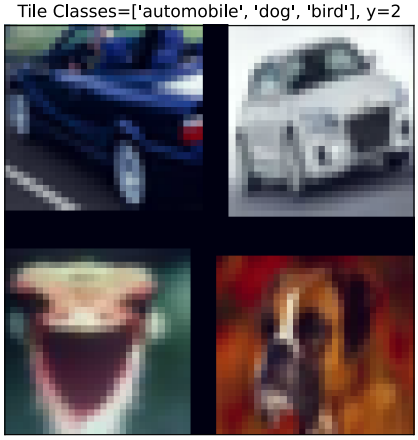


Figure 4. Example renderings of each MultNIST class alongside the MNIST images that make the full MultNIST.

Tile Classes=['dog'], y=0

Tile Classes=['cat', 'automobile'], y=1

Tile Classes=['automobile', 'dog', 'bird'], y=2



Tile Classes=['bird', 'ship', 'airplane', 'cat'], y=3

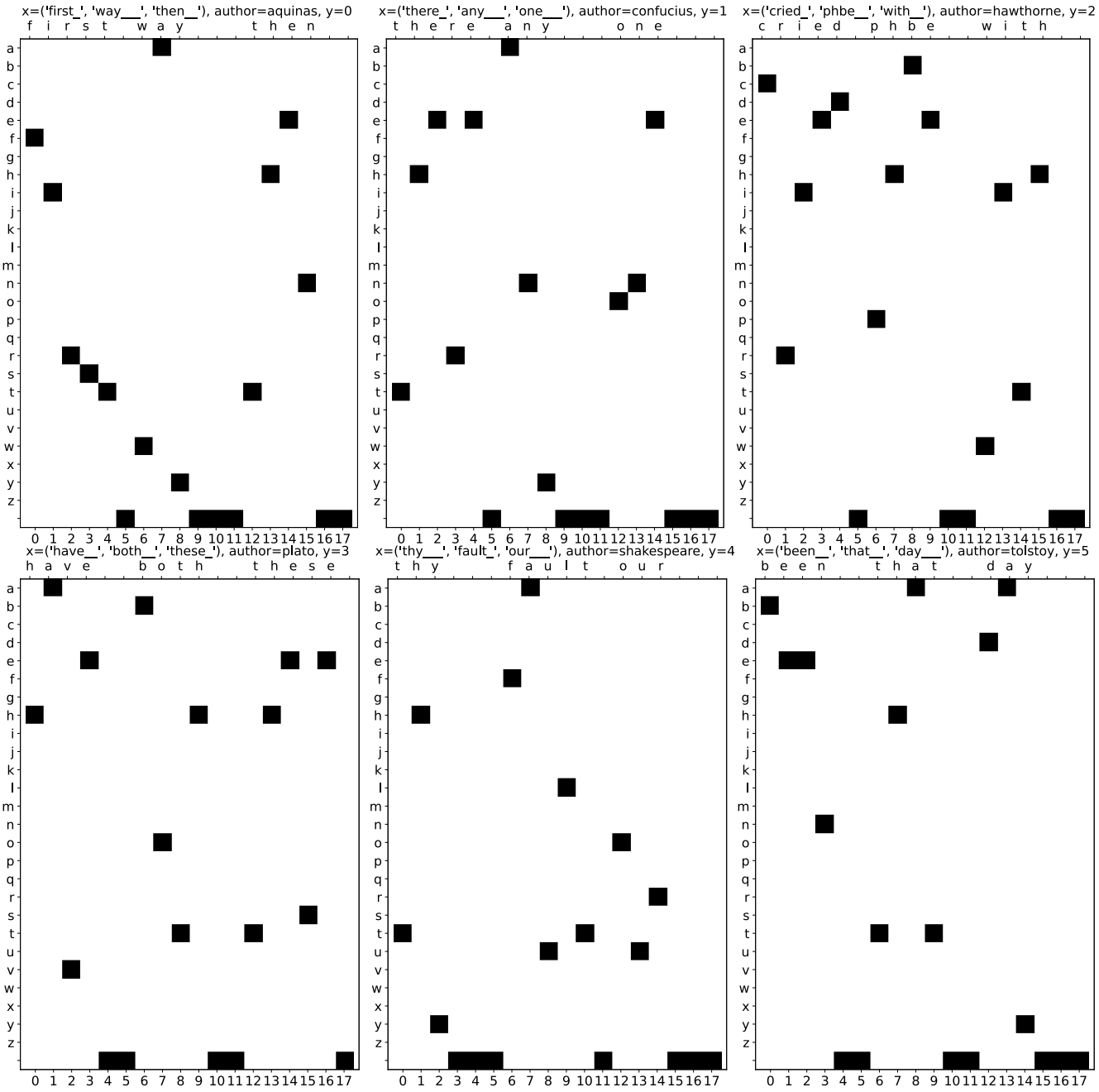


Figure 5. Example renderings of the Gutenberg dataset. The x-axis shows the index position in the string and the y-axis shows the character at the corresponding position.

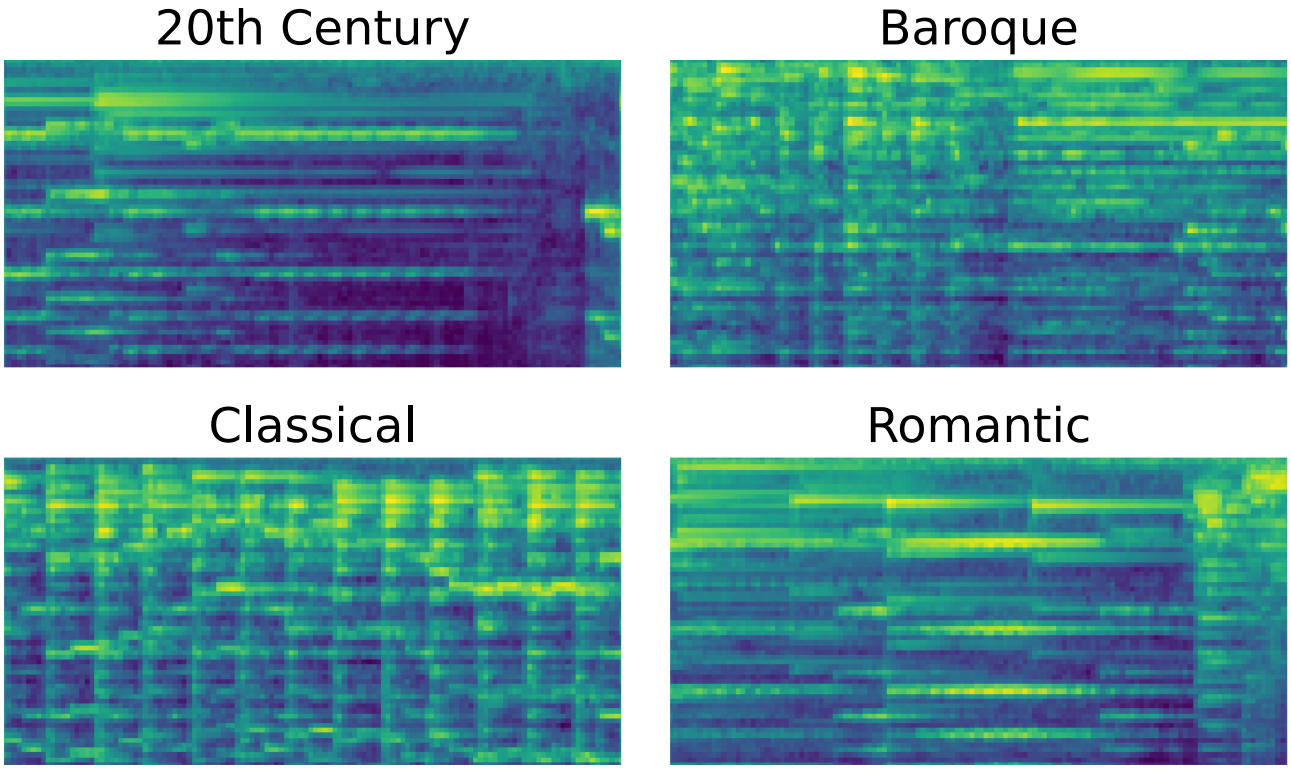


Figure 6. Rendered examples of the Isabella dataset.

Figure 7. Rendered examples of the GeoClassing Dataset.

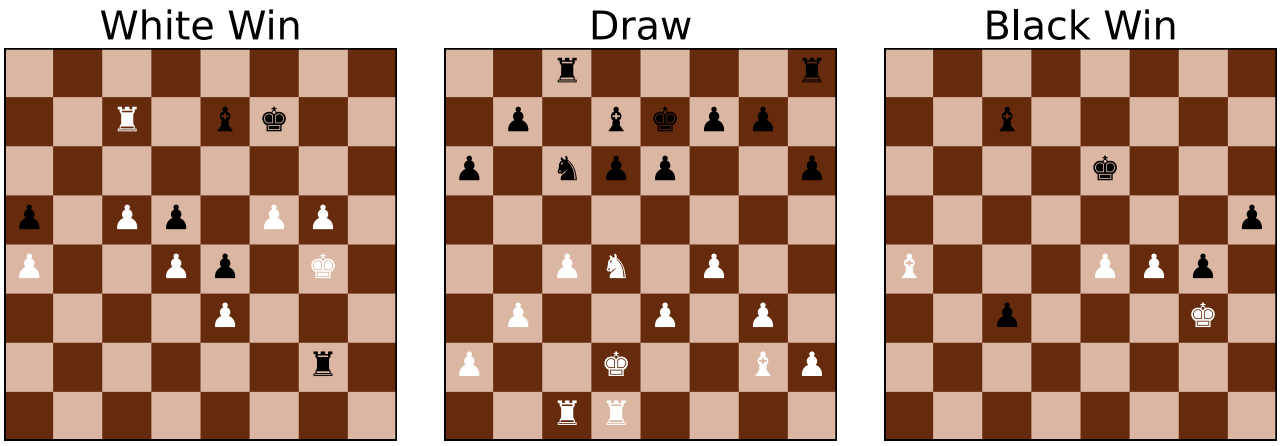


Figure 8. Examples of the Chesseract dataset converted into a standard board format.

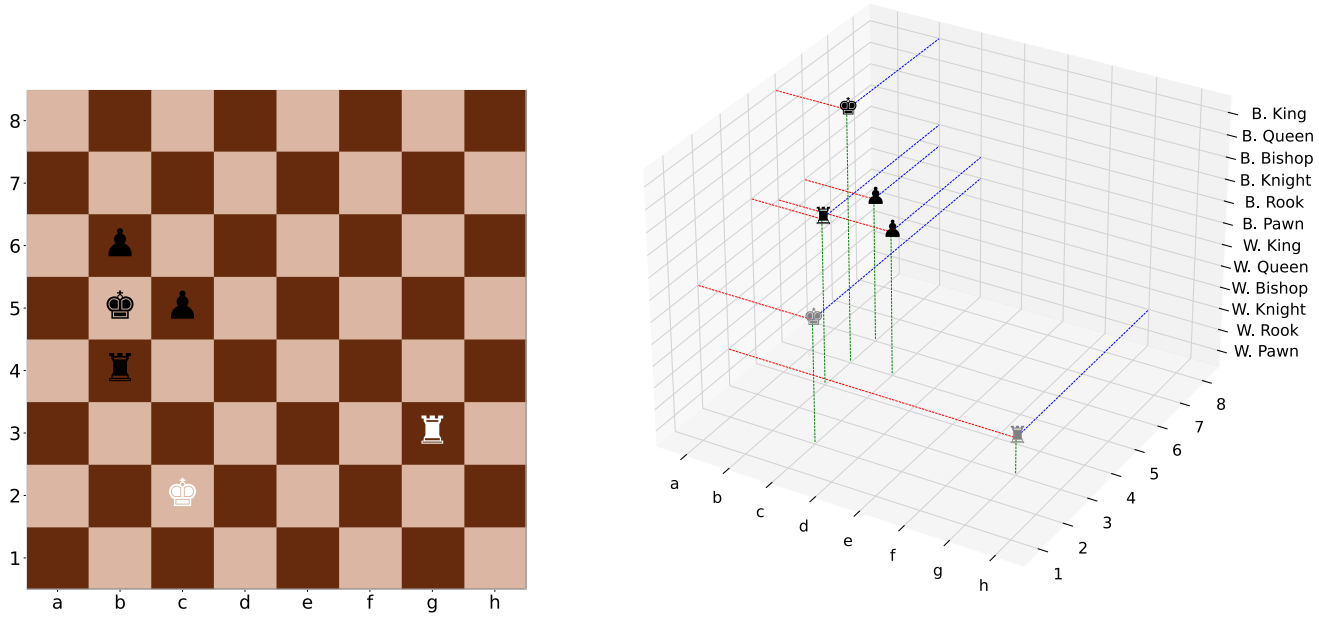


Figure 9. As Chesseract uses a 12-channel one-hot encoding to represent a board state, the board can be viewed as a 3D object. *left*: shows an example game that results in a Draw, where we have flipped the board so white is at the bottom as it would be in traditional notation. *right*: shows a 3D rendering of the *left* image. The white pieces are shown as grey in the 3D rendering for easier viewing.