GeoGuessr Country Classifier: Analyzing Regional and Economic Biases in Image-Based Country Detection

Sarika Ahire sa3937[at]princeton.edu
Bryan Zhang bz5989[at]princeton.edu

INTRODUCTION

GeoGuessr is an online game in which players attempt to guess the geographic location presented in an image from streetview maps. The closer the guess to the actual location, the more points are awarded. In this project, we create a GeoGuessr country detector that, after training on a selection of images, attempts to guess which country the images in a test set are from. To add our own twist on things, we also determine what sorts of categories our classifier tends to be wrong on, by labeling the images — using the countries as proxies — by their economic rating by the World Bank, as well as by region of the world. This allows us to create a model that attempts to predict the country based on various image parameters, as well as test to see if our model tends to perform worse on specific areas of the world or on different countries' economic level. Identifying bias can help us take steps to reduce it and develop fairer image classification models.

1. PROJECT DESCRIPTION

We first format and label the countries in our dataset by region and economic level. Then, using the ResNet CNN architecture, we predict the country, region, and economic level that a photo is taken from. We retrain our CNN by training the fully-connected layer and freezing the rest. Finally, we analyze the results quantitatively and qualitatively.

1.1. Hypotheses

We predict that countries with high regional variation, such as those containing a range of forests, beaches, mountains, and cities, will perform worse. These include the United States and New Zealand, for example. We expect Europe and North America to perform better than other regions since there is more data on them. For economic levels, we predict that high-income countries will perform better since there is also more data on them and there are more distinctive features to be found in pictures from these areas.

1.2. Challenges

The challenges of this project include data quality and quantity, intraclass variability, interclass similarity, and computational complexity. Some images from the dataset have very few relevant features, which makes it difficult for our model to accurately identify and classify them. For example, an image may just contain a street and field (Fig. 1). Another may just contain water and faint outlines of land (Fig. 2). Another may have extremely low illumination (Fig. 3).

Fig. 1: Image from Argentina, canvas_1629261479.jpg

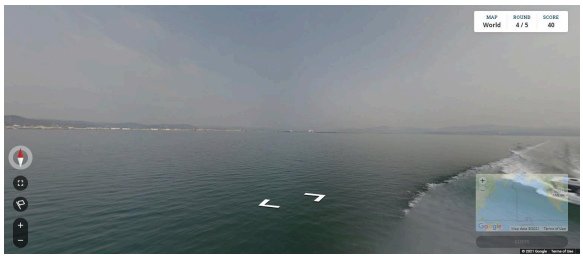

Fig. 2: Image from Japan, canvas_1629273876.jpg

Fig. 3: Image from Germany, canvas_1629300847.jpg

Moreover, the original dataset is unbalanced in terms of the number of images in each class; a class contains between 1 and 12k images. We modify the dataset to remove classes with too few images and remove images from classes with too many images, but even after these modifications, each class size ranges from around 80 to 800. The modifications also result in the loss of certain regional and economic categories. Finally, training and testing our model on the dataset, even after reducing its size, takes a significant amount of time — slightly over an hour per epoch.

2. PREVIOUS WORK

Our project is inspired by "CNN Plays Geoguessr: Transfer Learning on ResNet50 for Classifying Street View Images," which, similar to ours, creates "a computer vision (CV) model to play a modified version of Geoguessr where, instead of guessing the coordinates of a given street view image, the model would learn to predict the country of the location" (Dayton et al.). However, we use a different number of countries and images, and additionally look for bias according to location and income.

"Pigeon: Predicting Image Geolocations" (Haas et al.) contains similar ideas to this project, but focuses on more specific geolocalization and uses different datasets and methods.

3. DESIGN AND IMPLEMENTATION

3.1. Dataset

We used the GeoLocation — Geoguessr Images dataset from Kaggle. The original dataset contains approximately 50,000 images from the GeoWorld challenge and 124 folders, each containing images from that country. Images can contain people, buildings and landmarks, natural features, and more (Fig. 4, Fig. 5, Fig. 6). According to the dataset creator, GeoGuessr games aren't truly random: certain countries appear more than others, leading to a nonuniform distribution with each folder containing a range from 1 to 12k images. For example,

around a fifth of this dataset contains images from the United States.

Fig. 4: Image from India, canvas_1629417579.jpg

Fig. 5: Image from Mexico, canvas_1629577731.jpg

Fig. 6: Image from South Africa, canvas_1629259760.jpg

We altered the dataset for the purposes of this project by removing any countries that contained less than 80 images. We removed a total of 61 countries that way. We also created a cutoff of 800 images (after randomizing the order of input images) for each country, leaving us with 63 countries and 24,686 images for our country analysis.

We also added our own labels based on region of the world and income, according to World Bank Categories. The 12 location categories are as follows, with Europe split according to EuroVoc, a thesaurus maintained by the Publications Office of the European Union (Table 1).

Region	**Label**	**Number of Countries**
Africa	A	6
Central Asia	CA	3
Central/Eastern Europe	CE	10
East Asia	EA	6
Latin America	LA	7
Middle East	ME	3
North America	NA	3
Northern Europe	NE	3
Pacific	P	5
South Asia	SA	3
Southern Europe	SE	5
Western Europe	WE	9

Table 1: Region categories

Due to the 80 image minimum, while there existed countries within the World Bank's "Low" income category in the original Kaggle dataset, there weren't sufficient images to have any countries labeled "Low" in our training/validation/test datasets. Thus the three economic level categories are as follows (Table 2).

Economic Level	Label	Number of Countries in our Dataset	Number of Countries in World Bank
High	H	35	83
Lower middle	LM	14	54
Upper middle	UM	15	54

Table 2: Economic level categories

The evaluation metric for our dataset consists of splitting our modified dataset into 80 percent for training, 10 percent for validation, and 10 percent for testing. We then determine the loss and accuracy for training, validation, and testing.

3.2. Architecture

We first tried to predict the country that a photo is taken from with the EfficientNet CNN architecture, which achieves higher accuracy and efficiency by using a scaling method to uniformly scale depth, width, and resolution using a compound coefficient. This results in a parameter-efficient model that still maintains high accuracy; the model tries to learn the fewest necessary parameters to capture necessary concepts.

However, due to the high computational costs of EfficientNet, as we were running into epochs taking over nine hours, we decide to use the ResNet model, which uses residual learning with shortcut connections to mitigate the vanishing gradient problem, allowing for the training of deeper networks. We took the ResNet50 model (explicitly the IMAGENET1K_V2 trained weights) from the *torchvision* model library, removed the last fully-connected layer, then replaced it with our own modified fully-connected layer, which uses two linear layers each followed by a ReLU and dropout layer, before a final linear layer that maps to our classes (countries, econ, or region, depending on which model we were training). This allowed us to make minimal modifications to the original ResNet architecture (as we were not modifying the weights throughout the base model), while still leaving enough trainable parameters so that our model can map learned parameters in the later levels of ResNet to properties in our input images.

For our loss function, we used *torch.optim.Adam*, with our hyperparameters being learning rate and weight decay (regularization).

3.3. Algorithm

build_dataset.py: Splits each dataset type (based on country) into training, validation, and test datasets, at a 8:1:1 split ratio. Also caps the number of images used for specific countries in order to balance the dataset (since the given dataset strongly overweighs countries like the United States, Canada, the U.K., and Brazil, which provides a possibility of bias/overfitting on those specific sub-datasets)

build_dataset_econ.py: Splits each dataset type (based on economic rating) into training, validation, and test datasets, at a 8:1:1 split ratio. Also caps the number of images used for specific economic ratings to keep them generally in the same ballpark range (since we started off with an excessive

amount of images from countries rated "High" economically by the World Bank. This allows us to have a more generally accurately trained model.

build_dataset_region.py: Splits each dataset type (based on region) into training, validation, and test datasets, at a 8:1:1 split ratio. Also caps the number of images used for specific regions in order to balance the dataset, especially because some regions had a comparatively lower number of images per country (notably the South Asia region) whereas others had a excess of data (i.e. Central/Eastern Europe)

config.py: Contains the base (starting) parameters used in our models

confusion_countries.py: Gets test accuracy, as well as creates the confusion matrix for countries (hard to comprehend this matrix since it is a 63x63 matrix)

confusion_econ.py: Gets test accuracy, as well as creates the confusion matrix for economic levels

confusion_region.py: Gets test accuracy, as well as creates the confusion matrix for regions

create_dataloaders.py: A program that generates a dataset/DataLoader tuple for a dataset (used for respective tuples for the training, validation, and test datasets).

main.py: Contains the main structure for training our model by country

main2.py: Contains the main structure for training our model by country, given our own partially trained model as the starting point

main_econ.py: Contains the main structure for training our model by economic rating

main_region.py: Contains the main structure for training our model by region

param_finding.py: Contains a similar version of main, except used to try various values of our main two hyperparameters, learning rate (randomly selected using $10^{uniform(-4, -2)}$) and regularization value (called weight_decay) (randomly selected using $10^{uniform(-4, 0)}$).

paths.py: A program that reads through a downloaded dataset that uses folders to keep track of labels, and returns a list of paths to those images (effectively combining images and their labels into one unit)

4. RESULTS

4.1. Quantitative Evaluation

4.1.1. Predicting by Country

When predicting by country, we first train our model over 10 epochs (Fig. 7), leading to a reasonable reduction in loss for both validation and training, as well as small but good increases in model accuracy (Table 3).

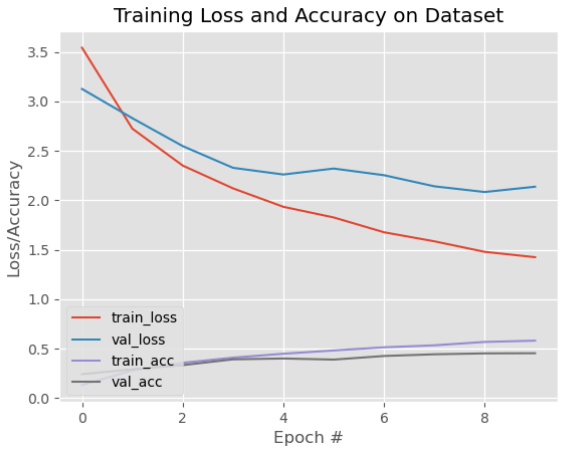

Fig. 7: Graph of loss and accuracy by country

Afterwards, we decide to train our model for another 10 epochs (Fig. 8), but here we find that our training and validation accuracy diverge. We start to see strong signs of overfitting on our data: as our training accuracy rises, our validation accuracy stays just under 50 percent (Table 3).

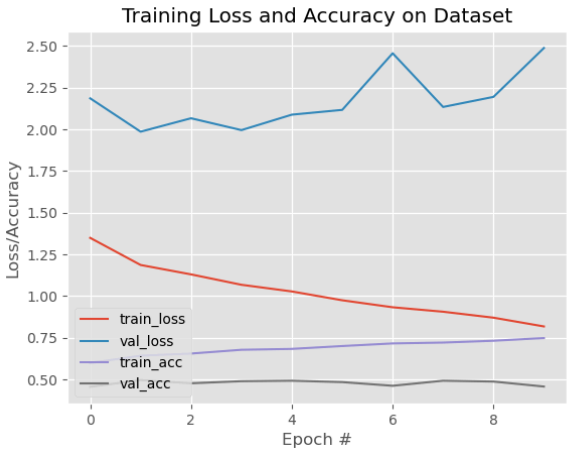

Fig. 8: Graph of loss and accuracy by country after further training attempt

Iteration	Train Accuracy (%)	Validation Accuracy (%)
1	13.1	24.2
2	28.1	29.1
3	35.7	33.3
4	40.9	39.2
5	44.9	40
6	48.2	39
7	51.4	42.6
8	53.4	44.3
9	56.8	45.2
10	58	48.4
Second run		
11	59.9	45.6
12	64.2	49.6
13	65.5	47.7
14	67.8	48.9
15	68.3	49.2
16	70	48.4
17	71.6	46.2
18	72.1	49.2
19	73.2	48.7
20	74.8	45.7

Table 3: Accuracy by country

Running on our test selection, we get a test accuracy of 54.63 percent, which is quite reasonable taking our validation accuracy into account. We use the first iteration model over the overfitted model to minimize how much our model focuses on specific qualities in the training data and uses major features the model finds in the images.

4.1.2. Predicting by Economic Level

When predicting by economic level, we see a relatively steady increase in both validation and training accuracy and reduction in their losses (Fig. 9, Table 4). Overall, this model seems to give the best accuracy, likely because there are only three labels to choose from (high, lower-middle, and upper-middle).

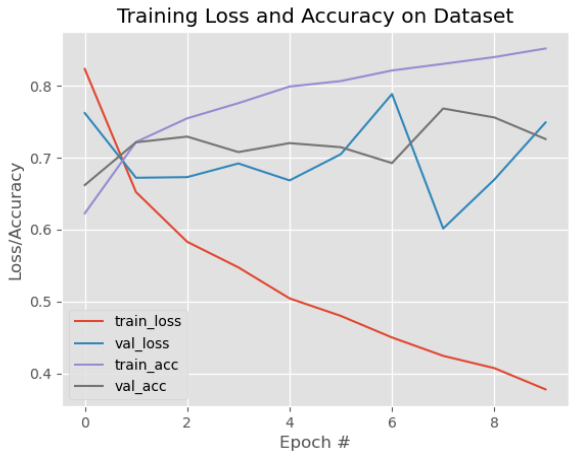

Fig. 9: Graph of loss and accuracy by economic level

Iteration	Train Accuracy (%)	Validation Accuracy (%)
1	62.3	66.2
2	72.2	72.1
3	75.5	72.9
4	77.6	70.8
5	79.9	72
6	80.7	71.5
7	82.1	69.3
8	83	76.8
9	84	75.6
10	85.2	72.6

Table 4: Accuracy by economic level

We get a reasonable test accuracy of 72 percent, which almost matches our validation accuracy.

4.1.3. Predicting by Region

For the region separations, we run into a similar issue as our countries dataset, where our train and validation accuracy rise to a certain level and then validation stops growing. This likely can be resolved to some extent using regularization, but that likely wouldn't increase validation accuracy by much; it would only stop train accuracy from growing so rapidly, as we see a very early divergence (starting from just the second epoch), showing that we already captured most of the special image features in the early training cycles to distinguish between the 12 regions.

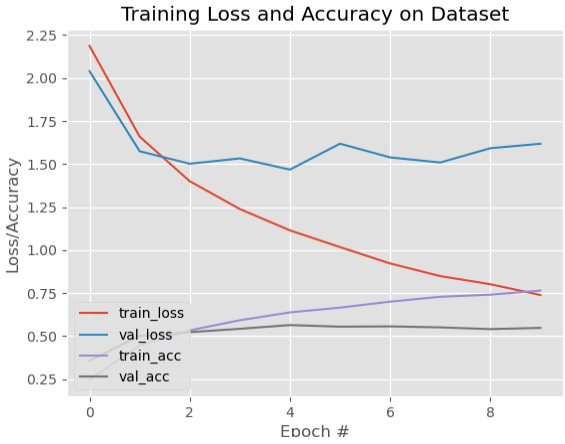

Fig. 10: Graph of loss and accuracy by region

Iteration	Train Accuracy (%)	Validation Accuracy (%)
1	24.5	35.65
2	44.58	50
3	53.3	52.2
4	59.1	54.2
5	63.7	56.3

6	66.5	55.4
7	70	55.6
8	72.8	55
9	74	54
10	76.4	54.7

Table 5: Accuracy by region

With regions, we get a test accuracy of 55.5 percent, which matches up well with our validation accuracy. This is better than randomly categorizing regions, as there are 12 regions to categorize and random chance would've only got us to around 8 percent accuracy.

4.2. Qualitative Evaluation

4.2.1. Predicting by Country

Although we were aware of the large number of countries we were testing, we did not anticipate this degree of unreadability in our confusion matrix when viewed from afar (Fig. 11). Zooming in, however, we can more reasonably interpret the diagram (Fig. 12). Germany ends up having the most number of correctly predicted labels, 75. The fact that it is in Europe and thus has more data likely contributes to this. Generally, the trend we see in the confusion matrix is the colorful nature of the diagonal line, which does mean that in general, we label images mostly to the right countries.

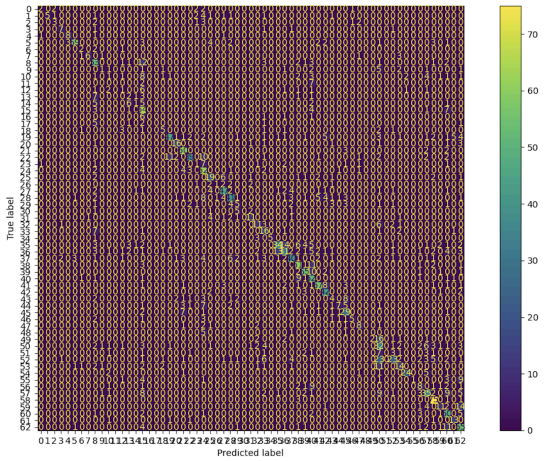

Fig. 11: Confusion matrix by country

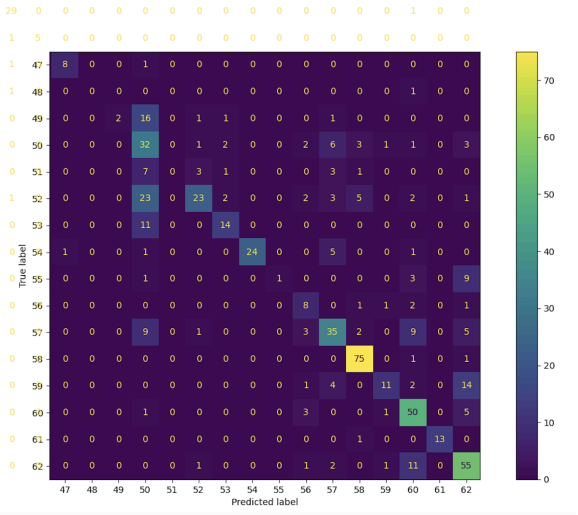

Fig. 12: Enlarged confusion matrix by country

4.2.2. Predicting by Economic Level

For economic level, our model appears to perform well for high (label 0) and upper-middle (label 2) incomes, and rather poorly for lower middle incomes (label 1) (Fig. 13). This is sensible and supports our hypothesis, as we see much more accurate economic level predictions for high and upper-middle income countries than for low income countries. Our model also does differentiate between high and lower-middle

income countries quite well, while having a reasonable mixup between H/UM and UM/LM country images, as there would still be reasonable overlap between images taken in different parts of said countries.

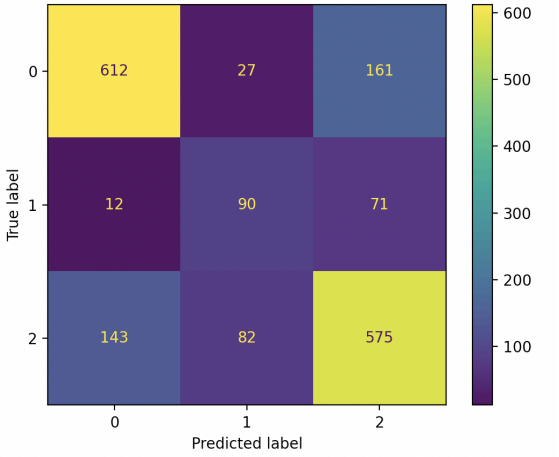

Fig. 13: Confusion matrix by economic level

4.2.3. Predicting by Region

Our model accurately predicts the label for regional categories most of the time (Fig. 14). It seems to perform the worst for the Southern Europe region, in the sense that our model often falsely predicts images to be from the Southern Europe region. A possible cause is the variety of cultures and terrains within this group of countries since images from this region vary from mountainous regions to areas near the sea, as well as more desert-like areas and full cities. This explains especially why pictures from East Asia and Latin America are so often mispredicted as Southern Europe, as they share similar varieties of terrain. In general, there is confusion between the European regions and North America. Additionally, there appears to be confusion between Central Asia and Central/Eastern Europe, which is likely due to their proximity and similar terrain.

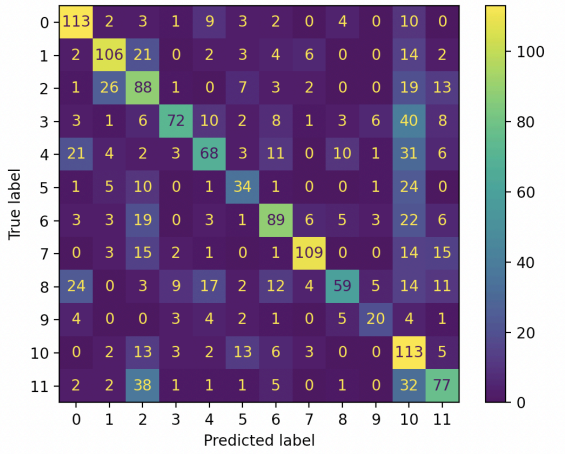

Fig. 14: Confusion matrix by region

4.3. Modifications

We made multiple modifications to the dataset, as described above, as we ran our model to increase accuracy and reduce the time spent running the model.

One modification we made was switching to ResNet from the EfficientNet CNN architecture, which uses techniques to scale up models to obtain increased accuracy and efficiency, to predict the country that a photo is taken from. Despite the name, our EfficientNet implementation proved to be extremely time inefficient despite achieving a relatively high accuracy of ~40 percent on the first epoch, so we decided to use ResNet. The cause of this inefficiency was likely its complexity from its scaling techniques, as it may have been efficient in terms of the number of parameters, but not in terms of time spent (our ResNet implementation takes slightly over an hour per epoch, whereas EfficientNet implementation took ~9.6 hours for a single epoch).

Another modification was increasing the regularization for training by region,

which ended up having minimal effect in boosting validation performance. This is possibly because regularization didn't have a significant impact relative to other training parameters, or the model wasn't having issues with overfitting to begin with.

5. DISCUSSION

5.1. Strengths

Strengths of our ResNet model include its capability of training deeper networks due to its residual connections, which help resolve the vanishing gradient problem. This allows it to achieve a better learning ability and improved performance. ResNet can also generalize well and is flexible, which allows us to use transfer learning and train our model on our GeoGuessr dataset. Relative to EfficientNet, it is also considerably faster, a

5.2. Weaknesses

Weaknesses of our ResNet model include high computational costs and optimization issues. Despite being more time efficient than EfficientNet, one epoch in ResNet still takes over an hour running purely on CPU. As a result, we weren't able to do sufficiently in-depth hyperparameter tuning and optimizing, which would have improved our losses and accuracies, especially for the countries and regions datasets.

6. CONCLUSION

Our choice of categories for the dataset images creates a model that helps identify regional and economic disparities. In the scope of the GeoGuessr game, a player can play into these biases to determine which locations and economic levels appear more often, and make their guesses accordingly. On a broader scale, in noting these disparities, we can take steps to reduce bias by modifying training and testing algorithms and improving data collection for underrepresented regions or economic levels. This would allow for more equitable models in classifying images of locations around the world.

Acknowledgements

We would like to thank Professor Heide, Professor Ramaswamy GS '23, and our advisor Julian for their help with this project.

REFERENCES

1. Papers

Dayton, F., Heo, J., & Werner, E. CNN Plays Geoguessr: Transfer Learning on ResNet50 for Classifying Street View Images.

Haas, L., Alberti, S., & Skreta, M. (2023). Pigeon: Predicting image geolocations. arXiv preprint arXiv:2307.05845.

2. Code

Rosebrock, A. (2021, October 11). Pytorch: Transfer Learning and Image Classification. PyImageSearch. https://pyimagesearch.com/2021/10/11/pytorch-transfer-learning-and-image-classification/.

PyImageSearch. Imutils/imutils/paths.py at master · pyimagesearch/imutils. GitHub. https://github.com/PyImageSearch/imutils/blob/master/imutils/paths.py.

Pytorch. vision/torchvision/models/efficientnet.py at main · pytorch/vision. GitHub. https://github.com/pytorch/vision/blob/main/torchvision/models/efficientnet.py.

Pytorch. vision/torchvision/models/resnet.py at main · pytorch/vision. GitHub. https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py.

3. Dataset

EuroVoc. EUR. https://eur-lex.europa.eu/browse/eurovoc.html?params=72%2C7226.

K., R. (2021, September 1). *Geolocation - Geoguessr images (50k)*. Kaggle. https://www.kaggle.com/datasets/ubitquitin/geolocation-geoguessr-images-50k/data.

World Bank Country and lending groups. World Bank Country and Lending Groups – World Bank Data Help Desk. https://datahelpdesk.worldbank.org/knowledgebase/articles/906519-world-bank-country-and-lending-groups.