**Todo List:**

1. Download data to google drive (so that it can be mounted to colab) ✅
2. Preprocess and store the data as a numpy array ✅
3. Setup the training process for some basic models ✅
4. Choose a model or architecture to work on ✅
5. Tune hyperparameters and collect data (umap/plots)
6. Write the report (29th to review and submit)

**Project Guidelines:** <https://eecs442.github.io/proj.html>

* Report Guidelines: <https://docs.google.com/document/d/1TSldaRQqPyA-9SmgcZWWWVJkap1njzc1HZcAoXSjtKU/edit>
* Project Showcase: <https://web.eecs.umich.edu/~justincj/teaching/eecs442//WI2021/showcase.html>
* Geoguessr Deep Learning guesser
  + <https://medium.com/@tef1/geoguessr-guesser-98e01efb5235>
  + <https://www.kaggle.com/datasets/ubitquitin/geolocation-geoguessr-images-50k>
  + <https://github.com/Stelath/geoguessr-ai>
* DenseNET121 Implementation: (To build our own based on this)
  + <https://medium.com/deepkapha-notes/implementing-densenet-121-in-pytorch-a-step-by-step-guide-c0c2625c2a60>

Filtered countries with only 100+ photos

60-80% training data, 10-20% validation data, and 10-20% test data

<https://www.kaggle.com/datasets/annaglass1/geoguessr-55countries?source=post_page-----98e01efb5235-------------------------------->

Geoguessr dataset (Original)

<https://www.kaggle.com/datasets/ubitquitin/geolocation-geoguessr-images-50k>

Access Kaggle dataset from Colab

<https://www.kaggle.com/discussions/general/74235>

Convolutional neural network

* <https://jonascleveland.com/best-image-classification-models/> for 2023

Google street view

* <https://developers.google.com/maps/documentation/streetview/overview> APIs that return static image

Pytorch tutorials:

* <https://pytorch.org/vision/main/models.html>
* <https://www.kaggle.com/code/pinocookie/pytorch-dataset-and-dataloader>

UMAP:

* <https://medium.com/mcd-unison/umap-an-alternative-dimensionality-reduction-technique-7a5e77e80982>
* Use this for analysis of the model, we will be able to tell which country are being mistaken for each other, and which are distinct from the rest

64x148x3 (these are the resized images that we are preloading)



**Sharing Code**

def topk\_eval(model, loader, k): # Evaluate accuracy on validation / test set

model.eval() # Set the model to evaluation mode

correct = 0

with torch.no\_grad(): # Do not calculate grident to speed up computation

for input in tqdm(loader):

batch, label = input['image'], input['label']

batch = batch.to(device)

label = label.to(device)

pred = model(batch)

label = torch.reshape(label, (label.shape[0], 1))

correct += (label == torch.topk(pred, k)[1]).sum().item()

acc = correct/len(loader.dataset)

print("\n Evaluation accuracy: {}".format(acc))

return acc

topk\_eval(model, test\_loader, k=5)

**Older Stuff**

* Winter 2021 rubric
  + <https://web.eecs.umich.edu/~justincj/teaching/eecs442/WI2021/assignments/project.pdf>
* Last semester rubric
  + [https://www.eecs.umich.edu/courses/eecs442-ahowens/fa23/psets/guidelines.pd](https://www.eecs.umich.edu/courses/eecs442-ahowens/fa23/psets/guidelines.pdf)f
* Try finding some kind of base
  + Previous paper that tests a similar question
* Diagnostic image-detection
  + Breast cancer - has benign and cancerous
    - <https://bcdr.eu/>
  + Brain-map
    - <https://portal.brain-map.org/>
  + National cancer institute imaging data commons
    - <https://portal.imaging.datacommons.cancer.gov/explore/>
* Tree/rock climbing rank
  + Take pics of trees/boulders and rank their climbing difficulty
  + We could also generate an optimal climbing route