Evaluating the Experimental Design of "EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification"

Simona Aksman, Aditya Kumar, Aravinth Paranan
UC Berkeley
{saksman, aditya_kumar, aravinth_paranan}@berkeley.edu

1. Introduction

Satellite imagery-based land use classification models have recently benefited from advancements in computer vision. In "EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification" [8], Helber et al. show that state-of-the-art deep convolutional neural networks (CNNs), such as ResNet-50 [7], GoogLeNet [14] and Shallow CNNs [10], can significantly outperform older image classification methods, such as Bag of Visual Words (BoVW), for the task of land use classification, obtaining an overall classification accuracy rate of 98.5%. However, Helber et al. use standard neural network architectures out of the box, and thereby do not tailor their models to best fit their data. In this paper, we use information theory-based experimental design and analysis methods for machine learning [3–6] to improve upon the experimental design of Helber et al.'s study.

In the following sections we describe the EuroSAT dataset and give some background on the neural networks used by Helber et al. Then we use information theory-based methods to measure capacity, generalization, and resilience for this study. Finally, we resize the networks Helber et al. used to better suit the data, and discuss our thoughts on how to make their study design more robust.

2. Background

The EuroSAT dataset that Helber et al. created and used in their study was originally derived from multi-spectral imagery obtained by NASA's Sentinel-2A satellite. Helber et al. then manually curated this data and defined 10 land use classes to use for a machine learning prediction task following the land use guidelines defined in the European Urban Atlas [1].

What is the variable the machine learner is supposed to predict? The machine learner is trying to predict the land usage class out of the 10 possible class types (Industrial Buildings, Residential Buildings, Annual Crop, Permanent Crop, River, Sea & Lake, Herbaceous Vegetation, Highway, Pasture, and Forest) for each satellite image. Examples of each class are provided in Figure 1.

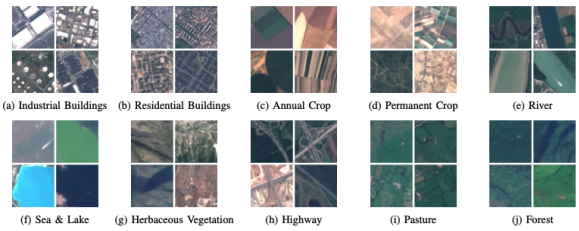

Figure 1. Sample images from each of the 10 classes in the EuroSAT dataset.

How accurate is the labeling? What is the annotator agreement (measured)? We believe that the labeling is highly accurate, as the paper stated that multiple authors manually reviewed the dataset of 27,000 images multiple times. However, the annotator agreement was not explicitly listed, leaving us unsure in terms of what is the upper bound of true accuracy on the dataset.

What is the required accuracy metric for success? Helber et al. evaluate the success of their methods using classification accuracy, which is defined as the percentage of images that are correctly classified overall (i.e. across all classes). In their original publication, they do not clarify whether this metric is evaluated on the validation set, training set, or both (they did not have a test set in their experiments). Following best practices for machine learning, we evaluate our models on training, validation, and test sets. After training several candidate models using the RGB data, Helber et al. set a classification accuracy benchmark of 98.5% for a ResNet-50, so we also expect to achieve this performance.

How much data do we have to train the prediction of the variable? There are 27,000 total images in the dataset spanning 10 classes, with 13 spectral bands per image. After experimenting with a range of training / validation splits, Helber et al. decided to randomly sample 80% of data for training and 20% for validation, so we use that training split

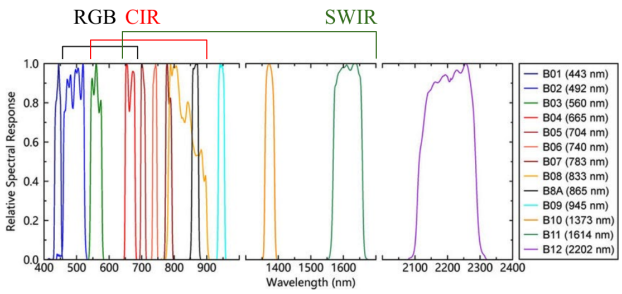

Figure 2. Spectral response signatures across the 13 Sentinel-2A spectral bands [11]. The wavelength ranges of the three modalities we analyze, RGB, CIR and SWIR, have been highlighted in the figure.

Class	Resilience to noise (dB)
SeaLake	-21.025684
Pasture	-16.225036
Forest	-14.678892
AnnualCrop	-11.703936
Industrial	-10.687796
River	-10.526201
Residential	-8.798462
PermanentCrop	-8.316015
Highway	-7.305676
HerbaceousVegetation	-6.375849

Figure 3. Noise resilience ratio per class in the RGB data.

in our experiments as well. However, instead of a 20% validation set, we use 10% of the data for validation and 10% for testing, such that it is a random 80%-10%-10% training-validation-test split. This ensures that we have a true out-of-sample dataset to evaluate on, the test set, since the validation set can be used for hyperparameter tuning.

Are the classes balanced? Classes are imbalanced, such that each of the 10 classes contains 2-3K instances (7-11% of the overall data).

How many modalities could be exploited in the data? As there are 13 spectral bands per image, any combination of these spectral bands could be exploited. In their study, Helber et al. primarily evaluate three spectral band combinations: visible light (RGB), color-infrared (CIR), and shortwave-infrared (SWIR). We also evaluate these three modalities, which vary significantly in the range of their spectral responses. Figure 2 shows how spectral response varies by wavelength across these three modalities. To construct images from these three modalities, we use the following spectral bands (where the name and the expected wavelength for that band are provided in parentheses):

- RGB: B02 (Blue, 490 nm), B03 (Green, 560 nm), B04 (Red, 665 nm)
- CIR: B08 (Near-Infrared, 842 nm), B04 (Red, 665 nm), B03 (Green, 560 nm)
- SWIR: B11 (Shortwave-Infrared 1, 1610 nm), B08A (Red edge 4, 865 nm), B04 (Red, 665 nm)

By combining these spectral bands, we produce three sets of 64×64 pixel images with 3 color channels each.

Is there temporal information? There does not appear to be temporal information, as the satellite images appear to be snapshots taken at a point in time. However, since satellites do revisit the same location periodically, it is technically possible that an image of a given location could be in the dataset multiple times but at different points in time. Helber et al. do not indicate that this has occurred, and so we assume that this is not the case, especially given that they manually validated each image in the dataset.

How much noise are we expecting? Helber et al. intentionally left in images with atmospheric effects, which add noise. We could apply an atmospheric correction prior to training to empirically determine how much noise this added to the images. However, a more standard approach to evaluate noise in images is to estimate the Gaussian standard deviation using median absolute deviation of the wavelet detail coefficients [2]. We use this approach to estimate the noise in the RGB data. This calculation assumes that the noise in our dataset follows a Gaussian distribution. After applying this calculation class-wise to the RGB dataset, it appears that the built-up land use classes (Residential and Industrial) have the highest mean noise, while Sea & Lake, Forest and Pasture classes have the lowest mean noise. This follows our visual intuition, as images in the less noisy classes are more uniform in appearance. Figure 1 exemplifies this result.

Next, we evaluate the noise resilience ratio to determine how much resilience is needed to handle noise in each class using the following equation:

$$\text{Noise resilience}_i\ (\text{dB}) = 20 * log_{10}(\frac{variance_i}{mean_i}) \quad (1)$$

where i indicates each class, and $variance_i$ and $mean_i$ are the class-wise variance and mean noise levels in the data, respectively. Results can be found in Figure 3. -n dB in noise indicates sensitivity to n dB of noise. The class least resilient to noise is Sea & Lake and the class most resilient to noise is Herbaceous Vegetation. This makes sense following some visual inspection of the data: images in the Sea & Lake class look highly similar in appearance and a model learning these features may therefore learn to expect this homogeneity (i.e. a blue body of water, sometimes sur-

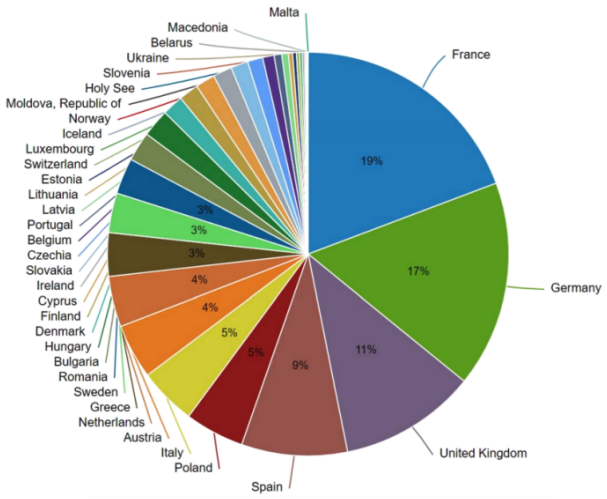

Figure 4. Distribution of countries included in the EuroSAT dataset.

rounded by land). The Herbaceous Vegetation class, on the other hand, contains many different types of visual patterns, as vegetation can take on a wide variety of different natural forms. A model learning this data may learn to map a broader set of patterns to this class than it would for the Sea & Lake class.

Do we expect bias? We expect bias at the population level as well as the class level. There is bias at the population level because the EuroSAT dataset only includes satellite imagery for 34 European countries. See Figure 4 for details. In particular, images of France, Germany, and the United Kingdom appear to represent 47% of the data, and so we would expect Helber et al.'s land classification models to perform especially well when evaluated on these regions. These models may not generalize well when evaluated on imagery from non-European countries. There is also bias at the class level due to class imbalances, which produce a slight bias towards the Annual Crop, Forest, Herbaceous Vegetation, Residential, and Sea & Lake classes, and away from the Pasture, Highway, Industrial, Permanent Crop, and River classes. Images from the pasture classes are least common, representing only 7.4% of the overall data. These class imbalances are likely a result of the distribution of land use within the countries included in the dataset, and are therefore likely to be linked to the population-level bias previously discussed.

3. Method and Results

What is the Memory Equivalent Capacity for the data (as a dictionary)?

To determine the memory equivalent capacity (MEC) of the data, we reproduce Algorithm 1 in [3] with one modification to the algorithm to enable it to handle multi-class data, as the original algorithm only handles binary data. To handle multi-class data, we initialize the reference class label that is used to count thresholds to be the label for the first row in the data. Then as we iterate through the data, we update this reference class label to be the class label for the latest row we counted a threshold for.

GoogLeNet AvgPool		
RGB	CIR	SWIR
14,560	14,604	14,586
ResNet-50 AvgPool		
RGB	CIR	SWIR
29,283	28,856	28,882
Shallow CNN MaxPool		
RGB	CIR	SWIR
43,156	42,398	41,465

Table 1. Data MEC in bits for various training datasets, assuming an Equilibrium Machine Learner.

See Table 1 for our analysis of data MEC using Algorithm 1. Note that we apply compression via the CNN layers from the three networks we worked with, GoogLeNet, ResNet-50, and Shallow CNN, and therefore compute data MEC on the data after the CNN layers have been applied (more details on how and why we did this are provided in the next section about network capacity). For reference, the MEC of the raw RGB training data is 172,252 bits, meaning that we would expect to need a much larger network to learn the data if we did not first use convolutional layers to compress the data.

We also compare the data MEC across three different modalities within the data, RGB, CIR, and SWIR, and find that all three have similar data MEC after CNN compression. Therefore, in the following sections we simplify our analysis to only consider RGB data after CNN compression.

Note that under the assumptions of an Equilibrium Machine Learner, the function being learned is the sum of a dot product of the input data and a set of weights, where all weights have been set to 1 to produce an equilibrium state. We can then simplify the learning problem to only training biases, where we empirically evaluate whether an additional bias threshold is needed whenever a class change occurs. This simulates building a fully connected 3 layer neural network with one hidden layer, which is all that is needed to learn any non-linear function according to the Kolmogorov–Arnold representation theorem [9]. These assumptions are further explained in [3].

What is the expected Memory Equivalent Capacity for a neural network? We evaluate the MEC of GoogLeNet [14], ResNet-50 [7], and Shallow CNN (AlexNet) [10] starting from the final pooling layer (which

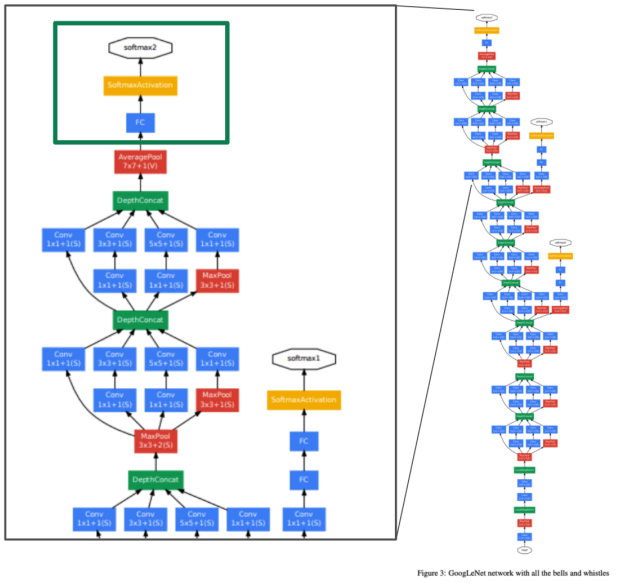

Figure 3: GoogLeNet network with all the bells and whistles

Figure 5. Section of the GoogLeNet [14] network that we train and count MEC for is outlined in green. The rest of the network is frozen so no new weights are learned and MEC is 0. Note that the version of GoogLeNet that we used has an additional FC layer.

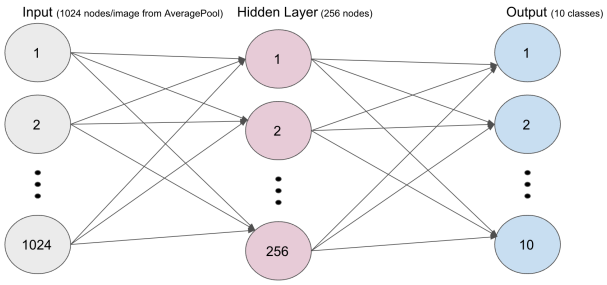

Figure 6. GoogLeNet's fully-connected layers that we count MEC for.

is an AvgPool layer for GoogLeNet and ResNet-50, and a MaxPool layer for the Shallow CNN). Figure 5 provides an example of the section of these networks that we actually train and therefore count MEC for. We then count MEC for the remaining fully connected (FC) layers for each network. As each of these networks has been pretrained, we can freeze all layers prior to the final FC layers during training. As a result of this freezing step, layers prior to the final FC layers contribute 0 MEC to our calculations. Note that all networks have been pretrained on ImageNet data [13] because this step improved model performance in [8]. Pretraining also contributes 0 MEC to our calculations, as it is a fixed input.

To calculate the MEC of each of these networks, we use

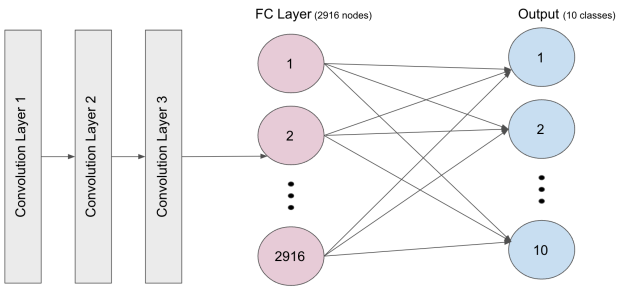

Figure 7. The shallow CNN architecture that we reproduced from the paper. We only count the MEC of the network after the convolutional layers.

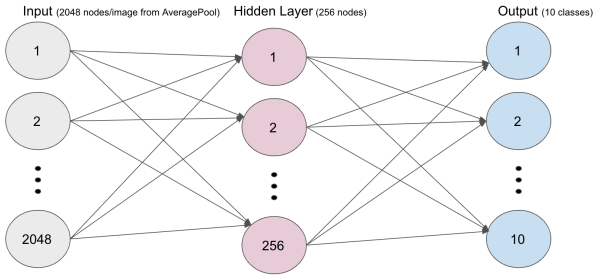

Figure 8. ResNet-50 FC layers that we count MEC for.

the following four rules established in [5]:

1. The output of a single perceptron is maximally 1 bit of information.
2. The maximum memory capacity of a perceptron is the number of parameters, in bits.
3. The maximum memory capacity of perceptrons in parallel is additive.
4. The maximum memory capacity of a layer of perceptrons depending on a prior layer of perceptrons is limited by the maximum output in bits of the prior layer.

Below are our network MEC calculations after applying these 4 rules.

GoogLeNet:

$$1024 * 256 + 256 + \min(256, (256 + 1) * 10) = 262,656 \text{ bits} \quad (2)$$

ResNet-50:

$$2048 * 256 + 256 + \min(256, (256 + 1) * 10) = 524,800 \text{ bits} \quad (3)$$

Shallow CNN:

$$2916 * 10 + 10 = 29,170 \text{ bits} \quad (4)$$

	GoogLeNet	ResNet-50	Shallow CNN
Generalization (bits/bit)	0.03	0.017	0.20
Average Resilience (dB)	-30.46	-35.39	-13.97

Table 2. Generalization and resilience for each network after training on RGB data.

See Figures 6, 7 and 8 for diagrams which show the final FC layers we evaluate MEC for.

What is the expected generalization in bits/bit and as a consequence the average resilience in dB?

Given multi-class data, expected generalization is calculated using the following equation:

$$G = \frac{\sum_{\mathrm{i}}^{\mathrm{C}} \mathrm{P_i log_2}(\frac{1}{\mathrm{P_i}})\mathrm{numCorrectlyClassified_i}}{\mathrm{MEC_{network}}} = \frac{[bits]}{[bit]} \quad (5)$$

where i is each class from 1 to C, and P_i is the probability of each class. We evaluate G on our training data. For a model to generalize when there are multiple classes, G must exceed $\frac{C}{C-1}$. Since $C = 10$ for this dataset, G must exceed 1.1 to achieve generalization.

As we show in Table 2, the generalization ratios of the three networks used by Helber et al. indicate that these networks are memorizing rather than generalizing, as each of the generalization ratios are less than 1.1. This matches our expectations regarding generalization for the ResNet-50 and GoogLeNet models, as we can see that these networks have much greater capacity than is required to memorize the data.

Average resilience is calculated with the following equation:

$$\text{Average Resilience (dB)} = 20 * \log_{10}(\mathrm{G}) \quad (6)$$

We calculated average resilience values for each of our three networks. Those values can be found in Table 2. We observe that, as the generalization ratio decreases, resilience decreases, indicating that a model is becoming more sensitive to learning noise. The ResNet-50 model is the largest in terms of network MEC, and therefore has the lowest generalization ratio and the lowest average resilience.

Is that resilience enough for the task? Since -n dB of resilience indicates that a model would be sensitive to n dB of noise, this means that GoogLeNet and ResNet-50 will be more sensitive to noise than the Shallow CNN. The Shallow CNN may therefore be less likely to overfit to the more noisy classes, such as the Residential or Industrial classes. In Figure 3, we determined that a learner would need between -21 dB and -6 dB resilience to noise for this dataset, depending on the class. We would expect that the Shallow CNN will handle this noise better than the GoogLeNet and ResNet-50, which will be more likely to memorize the noise.

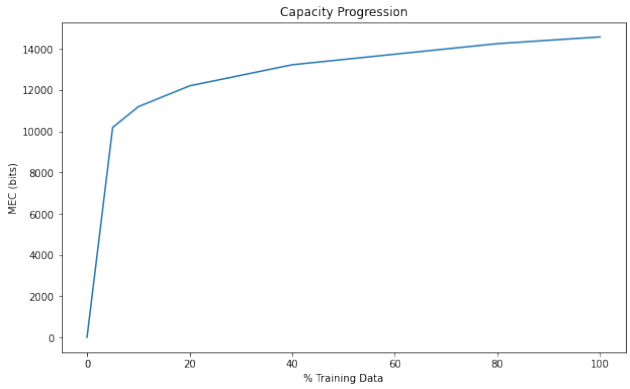

Figure 9. Capacity progression for the RGB training data after training GoogLeNet through to the final AveragePool layer.

How bad can adversarial examples be? Some adversarial examples were bad enough to be manually removed from dataset: images with clouds, mislabeled images, and blank images caused by dead pixels. Since they were already removed, it is unknown how bad it would be if these examples had been included. Other adversarial examples, such as color-casted images due to atmospheric effects, were left in to improve classifier robustness. Including color-casted images still produced out-of-sample accuracy rates of greater than 98.5 percent, so they do not appear to be too detrimental.

Do we expect data drift? We do not expect significant data drift because the data is satellite imagery of land, and we do not expect Earth's landscape to change significantly enough to affect the prediction accuracy of the model. Additionally, one of the use cases of this model is to monitor landscape changes over time, so the expectation is that the model can be used for long periods of time.

Is there enough data? How does the capacity progression look like? To determine if there is enough data, we replicated Algorithm 2 from [3], which calculates the capacity progression for an Equilibrium Machine Learner. We assume that this algorithm can achieve 100% training set accuracy given the assumptions in [3]. The capacity progression plot for the RGB training data after the GoogLeNet AveragePool layer is provided in Figure 9, and for the RGB training data after the ResNet-50 AvgPool in Figure 10. For both of these datasets it appears that an Equilibrium Machine Learner will continue to learn the data at every training data split, but it is a non-linear progression, with very little additional learning occurring after 80% of the data has been learned. Based on Figure 11 from [3], which outlines possible outcomes when plotting capacity progression, it appears that the Equilibrium Machine Learner is somewhat generalizing. Therefore, there should be enough data for generalization to occur for these datasets.

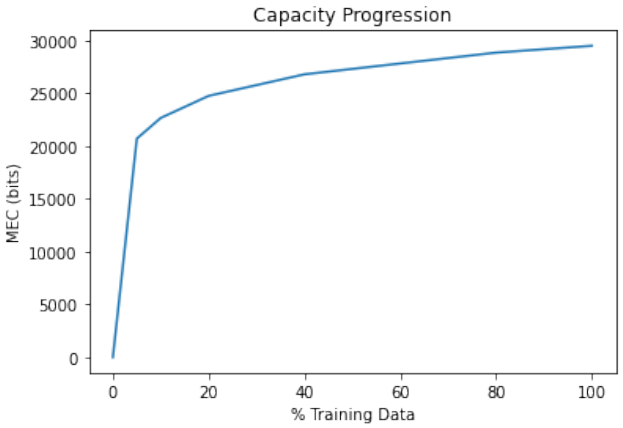

Figure 10. Capacity progression for the RGB training data after training ResNet-50 through the final AveragePool layer.

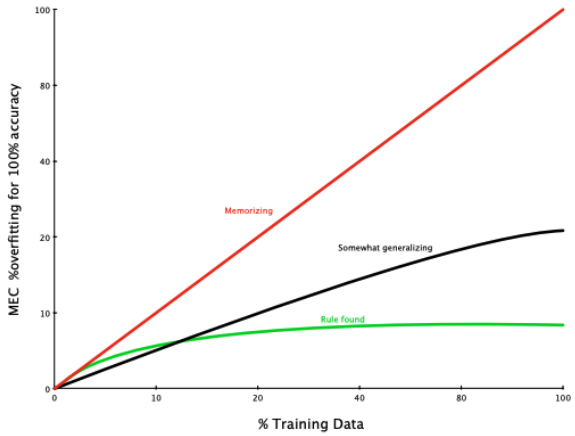

Figure 11. Three possible outcomes when plotting capacity progression [3].

Train your machine learner for accuracy at memory equivalent capacity. Can you reach near 100% memorization? If not, why (diagnose)?

We previously determined that the networks we are working with have capacity levels that differ from the capacity levels of their input data. GoogLeNet and ResNet-50 have network capacities that greatly exceed the capacity of the data, while the Shallow CNN actually has a network capacity that is less than the MEC of the data. We therefore need to resize our networks to match the capacity of our data to train for accuracy at MEC.

After increasing the capacity of the Shallow CNN to around 44,000 bits, we find that the training accuracy is 99% after 15 epochs. Hence, we can conclude that, for the Shallow CNN, we are able to memorize the data when data

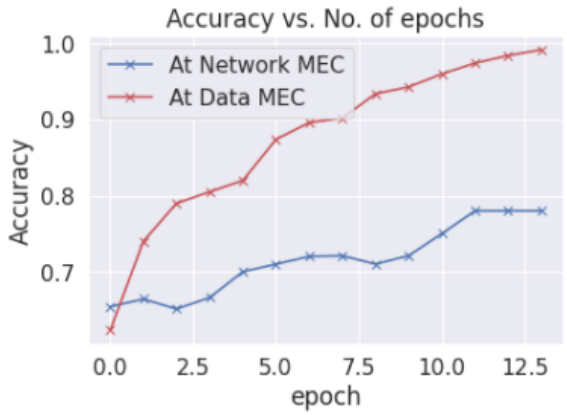

Figure 12. Shallow CNN training accuracy per epoch when training at network MEC and data MEC.

	GoogLeNet	ResNet-50	Shallow CNN
Training Accuracy (%)	99.5	98.4	99
Validation Accuracy (%)	97.6	98.0	98.1

Table 3. Training and validation accuracy of each network at memorization capacity (when dataset MEC is roughly equivalent to network MEC) after training for 15 epochs.

MEC is equivalent to network MEC. We share the results at data MEC and network MEC per epoch in Figure 12. Increasing the network's capacity to match data MEC greatly improves performance compared with training at the original network's MEC, from 78% to 99% training accuracy.

We also find that we can train GoogLeNet and ResNet-50 models to around 99% training accuracy at data MEC after decreasing the capacity of both of these networks to match the data MEC of each of the RGB training datasets (around 14,000 bits for GoogLeNet and 29,000 bits for ResNet-50). The results of these experiments can be found in Table 3.

Note that, in order to train for memorization, we had to remove an FC dropout layer from the GoogLeNet and ResNet-50 architectures, as this applies regularization that previously prevented us from reaching 100% training accuracy.

Train your machine learner for generalization: Plot the accuracy/ capacity curve. What is the expected accuracy and generalization ratio at the point you decided to stop? How well did your generalization prediction hold on the independent test data? Explain results. How confident are you in the results? Do you need to try a different machine learner?

Next we train each network for generalization. See Figures 14, 15, and 16 for the generalization curves we plotted for each network. To create these plots, we start by setting

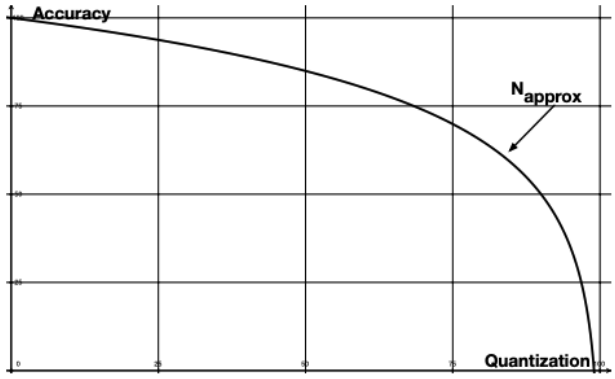

Figure 13. Expected quantization v. accuracy plot from [6]. We iteratively decrease MEC to simulate quantization.

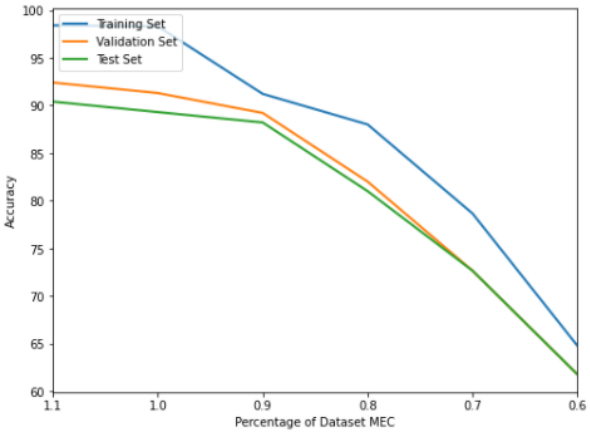

Figure 14. Training, validation and test set accuracy as MEC decreases for the Shallow CNN.

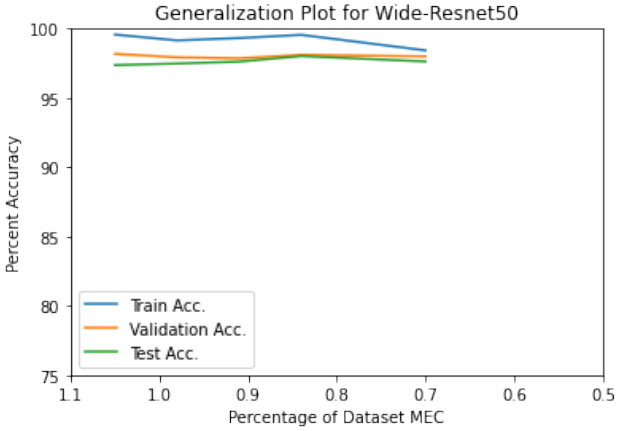

Figure 15. Training, validation, and test set accuracy as MEC decreases when training Resnet-50 FC layers. We were unable to decrease the MEC past a certain point due to the nature of high dimensional layers in neural networks.

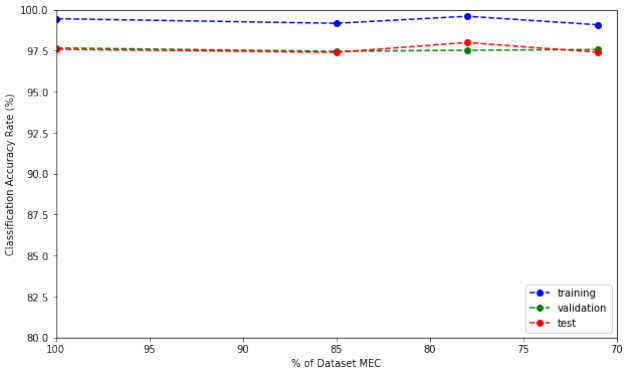

Figure 16. Generalization curve for a GoogLeNet. Training, validation, and test set accuracy does not diminish much as the % of dataset MEC decreases from 14,000 bits to 10,000 bits, indicating that generalization has not yet been reached.

the network's MEC roughly equal to the data's MEC, and then iteratively reduce the number of neurons for each network to simulate quantization i.e. information compression. The x-axis for each plot in Figures 14, 15, and 16 is the network's MEC as a percentage of the data's MEC, and the y-axis is the overall classification accuracy. We expected the results of this exercise to look like Figure 13, such that we can determine a value for N_{approx}, or the approximate noise factor, as per the theory established in [6]. At N_{approx}, the capacity of a network should be enough to handle the noise in the data, and we therefore expect generalization to occur. Among the three models we evaluated, however, only the Shallow CNN produces a generalization curve where we are able to vary quantization enough to determine a value for N_{approx}. For the Shallow CNN, it appears that this point may be at around 90% of the data's MEC, as this is where there is both an inflection point in the validation set accuracy rate, and the distance between the training and validation set accuracy rates is minimized. Beyond this point there is a drop-off in validation set accuracy. At a network capacity that is 90% of the data's MEC (around 37,800 bits), we find that our expected training accuracy is 98.3%, our expected validation accuracy is 91.3%, and our expected generalization ratio is about 0.31 bits/bit. It appears that our estimated value of N_{approx} occurs at an MEC value that exceeds the original network's MEC (29,170 bits), indicating that the original network used by Helber et al. was not correctly sized to handle noise, and therefore was unnecessarily reducing information content.

Despite the low generalization ratio of 0.31 bits/bit (we had expected this ratio to exceed 1.1), we are confident that

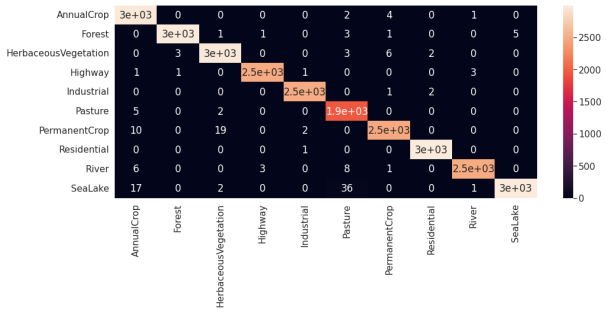

Figure 17. Confusion matrix for the Shallow CNN's predictions on the test set where the network MEC roughly equals N_{approx} (37,800 bits).

the Shallow CNN produces strong generalization given the strong test set accuracy, which closely mirrors the validation set accuracy. Note that test set accuracy is computed in a step that we keep entirely separate from the training and validation loop to ensure that it is truly an out-of-sample estimate. We also evaluate the class-wise performance on the test set via a confusion matrix for further validation of our results. Figure 17 shows that the model has strong test set performance across all classes, and is therefore highly robust within this setting. If, however, we were to transfer this model to a new geographic setting, such as a non-European country, we are not as confident that the performance would be as strong. Further testing is needed to determine how this model would perform in other geographic settings.

For GoogLeNet and ResNet-50, we do not see signs of generalizing within the ranges we have plotted. See Figures 15 and 16 for more details. However, we cannot further reduce the MEC of these networks to below around 70% of the data's MEC due to the topological constraints of these networks. This is due to how the FC layers are constructed: in each FC layer, every neuron in a given layer is connected to each of the input neurons from the previous layer. As a result, the number of input neurons multiplied by the number of output neurons in a layer plus the number of output neurons is the minimum MEC possible for a fully-connected network. For a GoogLeNet, this is about 10,250 bits ($1024 \times 10 + 10$). In addition, since we assume that the CNN layers are pretrained and frozen, and therefore fixed inputs, we cannot tweak these layers to further reduce the MEC of these networks. To further reduce capacity for these networks, we would need to train a new network which produces a smaller output from the CNN layers, which greatly increases the computational resources and time needed to train these networks. We briefly attempted to do this and found that, to produce a network with 8,690 bits of capacity by training CNN layers from scratch, it would take at least 150 epochs, at which point we are only able to reach around 68% training accuracy and 63% validation accuracy. Overall, we believe that using pretrained and frozen CNN architectures makes sense when these networks show very strong validation and test set accuracy rates, as we see is the case for GoogLeNet and ResNet-50, despite the limitations in generalization performance.

Parameter	**Shallow CNN**	**ResNet-50, GoogLeNet**
Initial Learning Rate	1e-3	1e-3
Epochs	15	15
Batch Size	16	64
Decay	1e-6	1e-3
Momentum	0.9	0.1
Optimizer	SGD	Adam
Loss	Categorical Crossentropy	Categorical Crossentropy

Table 4. Hyperparameters for reproducing neural networks in [8].

4. Discussion

Comment on any other quality assurance measures possible to take/ the authors should have taken. Are there application-specific ones? If time is present: How did you deal with it? Minor quality assurance measures were taken by the authors in terms of dataset quality, as the dataset was reviewed multiple times by multiple authors. However, no annotator agreement was listed, and no analysis was done regarding MEC calculations with relation to the large network size - the neural networks were primarily evaluated via accuracy/loss. As such, this analysis was conducted to evaluate the quality of the machine learning paper.

How does your experimental design ensure repeatability and reproducibility? With the release of [8], Helber et al. open-sourced their data but not their code. This means that the original experiments did not ensure repeatability. To ensure repeatability in our experiments, we have open-sourced our code, which is available here. In our GitHub repository we have included Google Colab notebooks for the models we evaluated across each modality evaluated, such that the results in this paper can be replicated by running these Google Colab notebooks.

Although Helber et al.'s original code was not available, others reproduced some of their results from their paper description and open-sourced those results on GitHub. We reproduced Helber et al.'s ResNet-50 results for RGB images using this Google Colab notebook that was open-sourced on GitHub [12]. Code for creating the GoogLeNet and Shallow CNN models, as well as CIR and SWIR datasets, were not open-sourced, so we reproduced those results from the descriptions provided in [8].

Our Shallow CNN consists of three layers each composed of a 3×3 convolutional layer with stride = 1, followed by a subsequent 4×4 max pooling layer with a stride = 2. All three layers use a ReLU activation function. The layers are followed by a fully connected layer. The hyperparameters used to train the model are given in Table 4.

In some cases, we used different parameters than were used in the original paper: for instance, we only trained for 15 epochs instead of 120, set the batch size to 64, and used an Adam optimizer. Despite these changes, we were still able to reproduce the benchmark performance for each of these models as reported in [8]. The full set of hyperparameters used for these models are also provided in Table 4.

5. Contributions

- **Aditya Kumar**: Questions 1, 3, 4, 6, 8; Resnet-50: data and network MEC, capacity progression, training for memorization and generalization
- **Simona Aksman**: Questions 2, 3, 4, 5, 6, 7, 9; GoogLeNet: data and network MEC, capacity progression, training for memorization and generalization; reproduced Algorithms 1 and 2 from [3]; LaTeX template setup; report editing and proofreading
- **Aravinth Paranan**: Questions 3, 4, 6, 7, 9; Shallow CNN: data and network MEC, capacity progression, training for memorization and generalization

6. Acknowledgements

Thank you to Professor Gerald Friedland and teaching assistant Eleanor Cawthon for the lectures, discussions, and office hours. We appreciate the several rounds of helpful input and feedback we received.

References

[1] European Commission. Mapping guide for a european urban atlas. https://ec.europa.eu/regional_policy/sources/tender/pdf/2012066/annexe2.pdf, 2012. 1

[2] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81.3:425–455, 1994. 2

[3] Gerald Friedland. *An Information View on Data Science: Experimental Design for Machine Learning*. 2021. 1, 3, 5, 6, 9

[4] Gerald Friedland and Mario Michael Krell. A capacity scaling law for artificial neural networks. *CoRR*, abs/1708.06019, 2017. 1

[5] Gerald Friedland, Alfredo Metere, and Mario Michael Krell. A practical approach to sizing neural networks. *CoRR*, abs/1810.02328, 2018. 1, 4

[6] Gerald Friedland, Jingkang Wang, Ruoxi Jia, and Bo Li. The helmholtz method: Using perceptual compression to reduce machine learning complexity. *CoRR*, abs/1807.10569, 2018. 1, 7

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1, 3

[8] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019. 1, 4, 8, 9

[9] Andrey Kolmogorov. On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. *Proceedings of the USSR Academy of Sciences*, page 179–182, 1961. 3

[10] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997, 2014. 1, 3

[11] Yingjie Li, Jing Chen, Qingmiao Ma, Hankui Zhang, and Jane Liu. Evaluation of sentinel-2a surface reflectance derived using sen2cor in north america. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, PP:1–25, 06 2018. 2

[12] Raoof Naushad. Land cover and land use classification using sentinel-2 satellite imagery with deep learning. https://github.com/raoofnaushad/Land-Cover-Classification-using-Sentinel-2-Dataset, 2020. 8

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. 4

[14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 1, 3, 4