Characterizing Image Classification Difficulties through Reduced-Dimension Class Convex Hull Analysis

Shawn McGrory

A Thesis

In the Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Electrical and

Computer Engineering) at

Concordia University

Montreal, Quebec, Canada

November 2020

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Shawn McGrory

Entitled: Characterizing Image Classification Difficulties through Reduced-Dimension Class Convex Hull Analysis

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Hassan Rivaz

_____ External Examiner
Dr. Charalambos Poullis

_____ Internal Examiner
Dr. Hassan Rivaz

_____ Thesis Supervisor(s)
Dr. Krzysztof Skonieczny

_____ Thesis Supervisor(s)

Approved by _____
Dr. Akshay Kumar Rathore Chair of Department or Graduate Program Director

Mourad Debbabi, Interim Dean
Gina Cody School of Engineering and Computer Science

ABSTRACT

Characterizing Image Classification Difficulties through
Reduced-Dimension Class Convex Hull Analysis

Shawn McGrory

The ability to correctly recognize natural terrain is regarded as a critical actor in autonomous path planning success. Following recent promise shown by deep learning algorithms, much research has directed focus towards conveying these successes to visual terrain classification, however the scarcity of work directing informed neural network design has proved limiting to these efforts. This work presents an algorithm that can be used to quantify the difficulty of specific image classification tasks and to investigate the characteristics of particular difficulties and trends in a way that is interpretable by humans. An accompanying analytical procedure characterizes such image classification difficulties; identifying what makes some images easily distinguishable exemplars of their class and what makes others readily confused with other classes. Case studies are presented of insights identified through selected example analyses of terrain image classification datasets: discussing relative intensities of terrain classes from images taken by Mars rovers, and the impact of color gradients in separating sand from bedrock in color images of terrain, and its implications for remote sensing hardware used to supply classifier input. Additional investigations cover more general image datasets: the MNIST hand written digits dataset, and key background color features in CIFAR-10. We validated the technique's potential to architecture design through comparisons to various neural networks, discovering characteristics mutual between predicted difficulty and classification error. The results presented in this paper provide a jumping off point for the analysis of terrain classification difficulty, and can inform designers of autonomous vehicles how challenging it can be to distinguish classes of terrain and provide insight into the risk associated with making traverse decisions.

Acknowledgements

While I will begin by expressing my gratitude towards my supervisor, Dr. Krzysztof Skonieczny, however there are no words to convey this sentiment in full. Thank you for your endless patience, encouragement, and dedication to helping your students. Without your guidance and support, it is unlikely I would have yet settled on a research topic, let alone have completed this Thesis.

I must also express my deepest thanks to Dr. Michael P. Furlong, whose expertise inspired this project and has been deeply involved since its conception. I would also like to thank Mission Control Space Services Inc., from which members consistently provided invaluable collaboration, serving as a guiding force throughout this work.

To my colleagues, Dr. Meysam Effati, Jean Sebastien Fiset, Amin Haeri, Adriana Daca, Parna Niksirat, Dr. Amir Nassiraei, Tyson Boer, and all those met through my studies: I thank you for your encouragement and advice throughout these years, but most of all, I thank you for the endless source of inspiration you have provided; while it may not have been realized, you have each helped to steer this research.

Lastly, I express my thanks to my family, and girlfriend, whose support throughout these years have made this all possible.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Deep learning and convolutional neural networks (CNNs) have shown particular promise in general image classification tasks. Starting with AlexNet [5], CNNs have dominated competitions that evaluate performance on standardized image datasets, such as the ImageNet Large-Scale Visual Recognition Challenge. In response, autonomous robotics has seen considerable interest directed at transferring these successes to their own applications, fueling active research into deep learning algorithms for visual terrain classification [6][7][8][9][10]; terrain classes typically include sand, rocks, gravel, and sometimes bedrock, asphalt, or grass. Example images are shown in figure 1.1.

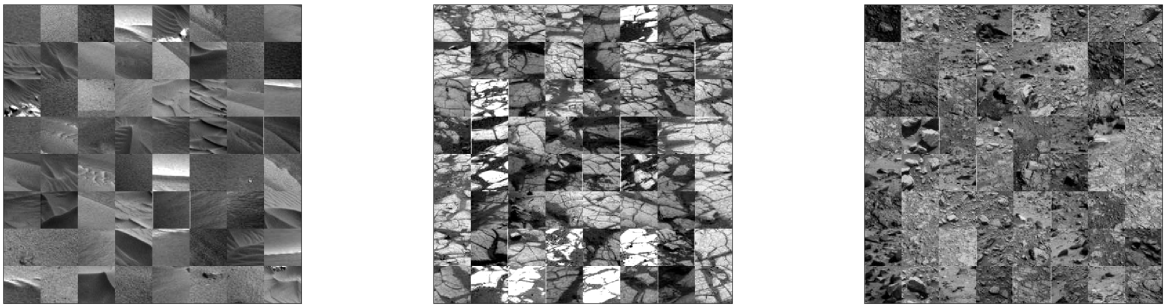


Figure 1.1: Examples of terrain-images within the Martian Terrain dataset [1]. Sand (left), bedrock (center), and rock-strewn (right).

Terrain classification has yet to consistently reach the prediction performance of general

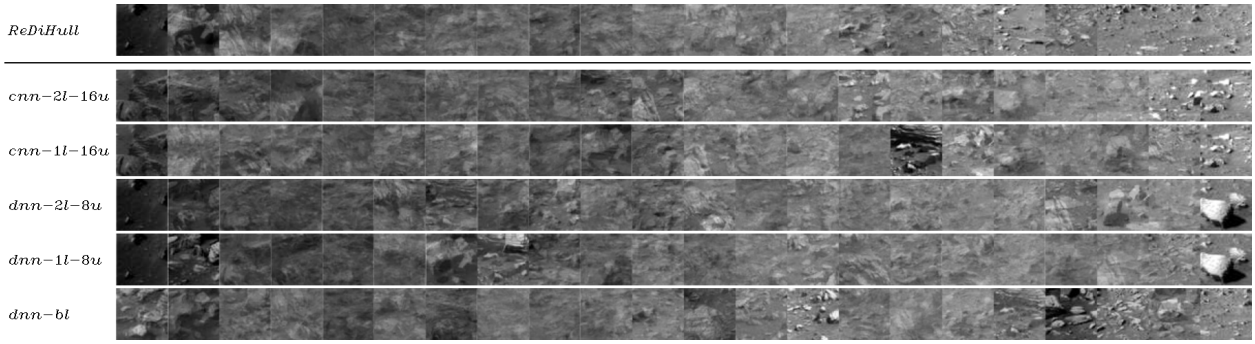


Figure 1.2: Increasing difficulty, from left to right, of *rock-strewn* terrain samples (true label) w.r.t *bedrock* terrain samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

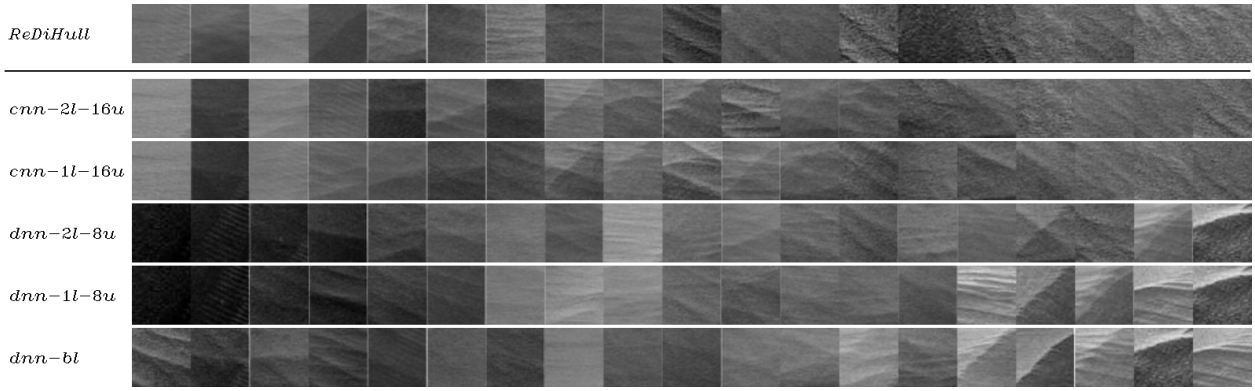


Figure 1.3: Increasing difficulty, from left to right, of *sand* terrain samples (true label) w.r.t *rock-strewn* terrain samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

object recognition, and there is no large publicly available common benchmark dataset, akin to ImageNet, specifically for terrain classification making comparing approaches problematic. In addition to the dearth of available training data, terrain classification may fundamentally be a more difficult task than general object recognition.

Particularly relevant to this issue is the scarcity of knowledge regarding network design, and even further limited knowledge of how to design neural networks for specific tasks such as terrain classification. Instead, typically manual or automated neural architecture search are employed, requiring the training and evaluation of many architectures [11][12][13]. Further, there is not yet a widely accepted metric for quantifying how difficult or complex a particular

Figure 1.4: Increasing difficulty, from left to right, of digit Eight samples (true label) w.r.t digit Seven samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

dataset is to classify using neural networks. There is at least consensus that any finite network's performance may be constrained by its capacity [14]; the capacity of a network is a quantity related to the number of architecture design parameters. Without a difficulty metric for comparing to previously evaluated datasets, though, each new classification task again requires training and evaluating many architectures of varying capacity.

This work presents an analysis technique that can be used to quantify the difficulty of specific image classification tasks and to investigate the characteristics of particular difficulties and trends in a way that is interpretable by humans. It then applies this technique specifically to terrain classification for planetary rovers, as well as demonstrates preliminary work targeting general image classification. Finally, we produce quantitative and qualitative comparisons of difficulties predicted via our method and difficulties observed for various deep learning architectures. A preview of the types of results that will be developed are shown in figures 1.2 through 1.4.

1.1 Literature Review

In order to guide the design of classifiers to accomplish a task, it is necessary to understand the difficulty of that classification task. Datasets of different complexity may be more efficiently learned by classifiers of differing capacity. Estimating dataset complexity can guide the search for neural networks that are simple enough to fit in constrained operational hardware, but complex enough to solve the problem.

As noted by [15], estimating dataset complexity is an underdeveloped problem. Prior work in dataset complexity estimation can be broken down into two categories. The first category analyzes image classification complexity in terms of engineered features, such as texture and colour, or higher-level abstractions like "objectness" and clutter. The second category consists of data-driven approaches to modelling the difficulty of learning a task.

The first category relies on features that humans can observe in the dataset that could explain confusing or ambiguous data points. Features include, but are not limited to, image quality [16], object variations [17][18][19], and scene information [20][21]. These approaches rely on the measurement or identification of features to predict the difficulty of a recognition task. [17] developed a metric which considers difficulty as a function of images' clutter - a quantity based on the number of possibly-object containing windows sampled before a true class-object is found. Others who explored difficulty prediction through measurement of image characteristics include [19] who considered difficulty as related to various pixel value statistics, as well as [22] who also considered somewhat similar statistics, however relating them to time required by a human to produce an image's annotation.

These limitations are of particular significance when considered for predicting terrain classification difficulties: in general, samples within terrain classes cannot be accurately described by a single pattern or feature as they lack commonality. To illustrate this, consider the samples of digit Eight presented within figure 1.4. It is clear that variation exists between samples, however, we may accurately presume that any and all samples within this digit class can be described as demonstrating a self-intersecting loop-like structure. In contrast, terrains classes are not likely to demonstrate these types of unifying features. As consequence of this property, characteristics and scenarios widely regarded as relating to difficulty, e.g.

occlusions, cannot be confidently presumed as introducing difficulties to terrain classification and similar data.

While using human-identifiable characteristics may allow for more interpretable architecture design, these methods make strong assumptions regarding the content and meaning of image characteristics and statistics that relate to difficulty. First, they assume the characteristics are static and globally relevant. Human operators may select features that satisfy their preconceived notions of difficulty, and may not truly represent the data in the dataset. Secondly, features that are designed to work in a dataset may not be generally applicable across different kinds of datasets, creating a new design burden with each new dataset analyzed. Given that any collection of predetermined features are unlikely to capture task-agnostic image classification complexity, researchers have developed complexity metrics that are derived from empirical analysis of the datasets.

The second category of complexity metrics use measures of difficulty of working with the dataset directly, instead of relying on features identified through a qualitative assessment of images. [15] estimates dataset complexity through the performance of different "probe networks" on the classification task. The complexity of the networks required to achieve good performance are used as a proxy for dataset complexity. [18] used human response time in a visual search task as a predictor of image difficulty. While the approach could be conceivably applied to classification problems, the metric really measures the complexity of individual images and not the dataset as a whole. Naturally, the complexity of the dataset could be estimated through an analyses of the complexity of the constituent images, but recruiting sufficient human labour to complete this task may be prohibitive. It should be noted that although these methods do not explicitly consider visually observable image attributes, visual similarity between difficult images is commonly seen.

As mentioned earlier, measures of dataset complexity based on hand-engineered features have the advantage of being interpretable to humans designing classifiers, but they may not generalize across datasets. Conversely, data-driven measures of dataset complexity can generalize across datasets, but they don't elucidate what about the dataset makes it complex to classify.

1.1.1 Attributes Considered Explicitly

Russakovsky et al. [17] developed a metric for estimating the difficulty of a joint classification and localization task. The authors propose a technique to obtain an image's difficulty using a measure of "clutter" - a quantitative description of how many possibly-object containing windows must be sampled before sufficient localization is achieved, using intersection over union as this criterion:

$$\text{IOU}\left(\hat{B}, B\right) = \frac{\text{area}\left(\hat{B} \cap B\right)}{\text{area}\left(\hat{B} \cup B\right)} \geq 0.5$$

Where B is the true object-region and $\hat{B}$ is a predicted region, such as those described as possibly-object containing windows.

Next, the authors sample one thousand windows $W^m = \{ W_1^m, W_2^m, ...W_{1000}^m \}$ for each image m then rank them based on their probability of containing any object, i.e. *objectness*. The type of object potentially contained within a window or whether the potential object corresponds to any of the provided classes is not considered in this step.

With W^m as an ordered set, they count the k number of windows which must be evaluated before $\text{IOU}\left(W_k^m, B_i^m\right) \geq 0.5$ where the image m contains $B_1^m, B_2^m, ...,$ labelled objects. This process corresponds to the OBJ measure of the image:

$$\text{OBJ}\left(m\right) = \min\{k \ : \ \max_i \text{IOU}\left(W_k^m, B_i^m\right) \geq 0.5\}$$

Finally, the clutter of a class may then be computed as:

$$\text{CLUTTER} = \log_2\left(\frac{1}{M}\sum_m \text{OBJ}(m)\right)$$

Conceptually, the resulting metric returns a higher value when applied to images or scenarios which are more difficult to localize. While this is less likely to be problematic in an object detection setting, their application does not consider how dissimilar an object is from its labelled class. In other words, it may be straightforward to identify that a labelled region indeed contains an object however this does not guarantee that a correct prediction of the object's class may be obtained with similar effort.

Vijayanarasimhan and Grauman [22] explored image difficulty estimation to improve the efficiency of an active learning routine for training a multi-instance multi-label (MIML)

[23] system. Active learning is a training paradigm in which the "student" classification algorithm may query a "teacher," in this case a human, for additional information regarding a given training example. Multi-instance multi-label problems differ from traditional image classification in that each image may contain multiple classes, as well as multiple instances of a given class.

In their work, the active learning system may query a user to obtain one of three possible annotation types: segmentation and labelling of any and all instances within an image, segmentation of a single instance, or an image-level label describing whether any such object is contained within the image. These three scenarios correspond to tasks demanding a decreasing amount of manual effort, and moreover, may result in varying amounts of benefit to the system. As such, the authors devise a method to estimate the *value of information* of a potential query - a quantity they obtain based on the trade-off between required effort and information benefit.

An image's required effort is defined as the (normalized) time required by human annotators to complete the relevant segmentation task. Using the acquired measurements, they create a system which when given an image, returns a cost estimation for the candidate annotation(s) in terms of predicted time required for the task(s).

This cost function does not map the image directly to a time prediction, rather, the authors extract low-level features to serve as predictors. Considered features include a histogram of oriented gradients, edge density, color histogram, and grayscale histogram. Using multiple kernel learning, they find that edge density significantly outweighs other features as a predictor.

The total cost of a dataset is computed as the sum of the annotation costs and prediction risk carried by unlabelled, labelled, and partially labelled images. Using this, they determine the information benefit, or *value of information* as the predicted change in total cost.

Within the perspective of our work, measurements of human subjects' activities is seen as significantly limiting. With that being said, their system, particularly their designation of the value of information attributed to a certain representation, is of great interest.

Work done by Liu et al. [19] attempted to predict the cost of segmenting an image. To do so, authors considered segmentation difficulty as being linearly related to an image's low level

statistics and texture properties. For each image within the studied Berkeley segmentation dataset, four representations are obtained - CIE LAB color, grayscale, local binary pattern (LBP), and log gradient. From each of these representations, they extract the mean and variance of pixel values. Denoting an image representation x flattened into a $1 \times m$ sized vector $x = \langle p_1, p_2, ..., p_m \rangle$ the mean μ and variance σ^2 may be computed as follows:

$$\mu = \frac{1}{m} \sum_{p \in x}$$

$$\sigma^2 = \sqrt{\frac{1}{m} \sum_{p \in x} (p - \mu)^2}$$

Next, they construct histograms for the color, grayscale, and LBP representation, as well as a log histogram for the log gradient representation. For each of these four histograms, they compute the entropy and variance.

Additionally, they determine the maximal bin-count within the LBP's histogram for use as a feature. The final features considered are slopes and intercepts obtained by approximating the log histogram of log gradients as a piecewise function comprised of two linear equations - one positive slope and one negative slope.

Concatenation of all listed features produces a 29-variable length feature vector X for each image. Using the images' f-measures [24] as the output variable y they model a basic linear equation of the form:

$$y = X\beta + \epsilon$$

As the authors' work is focused on object segmentation, the intermediate recognition task exists, albeit implicitly; the problem does not require prediction of a class label, solely correct segmentation of the depicted object. With that being said, the algorithm must be able to identify the correct object within the image. Thus while their technique lacks consideration of the individual class definitions, they are still capable of obtaining relevant difficulty estimates. Moreover, each image generally contains a single object-class, further reducing the need for incorporation of class definitions and their individual difficulties.

Rahman and Fairhurst [2] proposed a process for measuring dataset complexity based on the similarity between class' binarized image content. Binarization is achieved through the use of 1-bit, single channel images. In other words - a pixel value of 0 denotes "off" and 1 denotes "on". They express class similarity S_{kl}^{ij} by determining the number of shared, positive pixel values between binarized images, where the pixel value of an image at the xth column and yth row is obtained as $P(x, y)$. Given a pixel position, the kth image of the ith class, and the lth image of jth class, they simply compare pixel values between images:

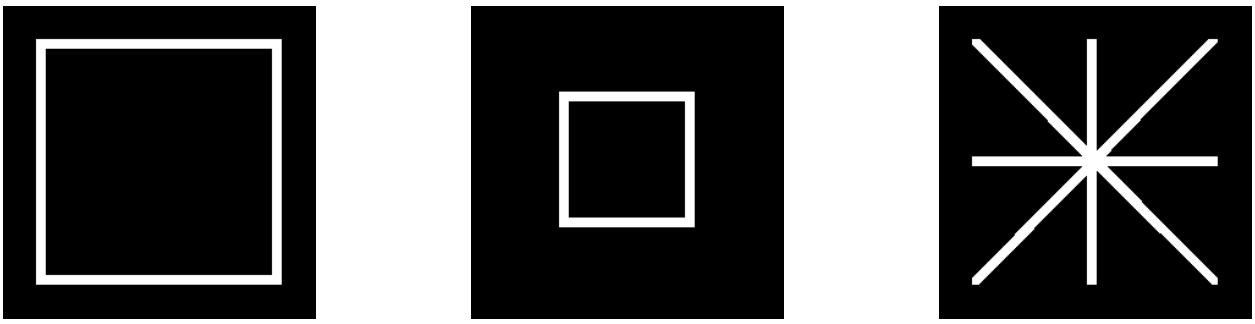
$$\zeta\left(P_{i,k}\left(x,y\right),P_{j,l}\left(x,y\right)\right)=\begin{cases}1 & P_{i,k}\left(x,y\right)=P_{j,l}\left(x,y\right)=1\\ 0 & otherwise\end{cases}$$

The function ζ is essentially a direct true-false comparison between pixel values at identical positions. Obtaining this comparison over all possible combinations of image pairs for the two selected classes allows them to obtain the between-class similarity measure:

$$S_{kl}^{ij}=\sum_{x=0}^{n_{cols}-1}\sum_{y=0}^{n_{rows}-1}\zeta\left(P_{i,k}\left(x,y\right),P_{j,l}\left(x,y\right)\right)$$

The chosen method of determining similarity between images is too naive for most problems; neglecting to consider any information besides mutual, positive pixel values results in a metric which may yield questionable statistics when applied to a task with more varied perturbations of size, position, and/or orientation. As example, consider the images shown in figure 1.5, where 1.5(a) and 1.5(b) are images sampled from the same class I and 1.5(c) is sampled from class J. Although 1.5(a) and 1.5(c) both depict image-concentric squares, the similarity produced for this pair of images is zero-valued. In contrast, the similarity between 1.5a and 1.5c is non-zero, despite being significantly different.

In summary, while work was demonstrated for datasets of handwritten and machine-printed alphanumeric characters, these datasets characterized as having minimal signal noise and/or within-class variations. Images having such characteristics are less likely to contain instances afflicted by the potential limitations of the similarity measure used.



(a) **(b)** **(c)**

Figure 1.5: Hypothetical scenario illustrating potential vulnerability of the similarity measure applied in [2]

1.1.2 Attributes Considered Implicitly

Ionescu et al. [18] present image difficulty as related to the time required by a human to complete an accompanied recognition task. This is somewhat similar to [22], however rather then measure the time required to segment the image, they measure the time required to detect whether an object is in an image. The response times of human-annotators were recorded using images from the PASCAL VOC 2012 dataset. Subjects were presented an image and object name from 20 possible objects, then required to determine whether the given object is visible within the image.

To gain insight into the influence of observable image properties on visual search difficulty, they compute Kendall's τ correlation between images' rankings produced by the measured response times and seven attributes potentially inducing challenge: (i) number of objects, (ii) objects' area relative to image size, (iii) objects' position relative to the image center, (iv) number of different classes, (v) number of truncated objects, (vi) number of occluded objects, (vii) number of objects marked as difficult. These statistics were computed using information supplied in the PASCAL dataset's annotation files.

The more strongly correlated attributes (i), (ii), (iii), and (iv) resulted in $\text{abs}(\tau) \approx 0.30$ while (v), (vi), and (vii) resulted in $\text{abs}(\tau) \approx 0.22$. To consider the statistics jointly, they train a ν-support vector regression model. This yields the highest correlation $\tau = 0.36$ among these tests.

These results were then used as a performance baseline to design a system for obtaining

higher-confidence predictions of image complexity. The proposed system utilized two CNN models to obtain deep features of an input image. These features were then concatenated and further processed to obtain complexity predictions with a Kendall's $\tau = 0.47$.

Of course, the process of measuring response times for humans will be unfeasible for many scenarios. It remains to be seen how this difficulty measure can be applied to other datasets, especially those which are significantly different than that which was used.

The work by Scheidegger et al. [15] determines dataset complexity using "probe nets" - this technique basically translates to the performance achievable by various sized deep learning architectures. The authors estimate image classification difficulty, or required effort, as a quantity found in relation to the networks' size and accuracy. The authors also considered k-means and silhouette scoring as comparative methods for predicting dataset complexity.

They found that the probe nets were significantly better at estimating dataset complexity ($R^2 = 0.89 - 0.99$) compared to k-means and silhouette scores ($R^2 \approx 0.31$). It is important to note that these results, while promising, are limited by the fact that the authors applied a single network as reference. In further detail, each methods' correlation coefficient was computed in reference to the performance of a ResNet-20 neural network architecture evaluated on the tested datasets.

Another potential concern is that the degree to which the measured complexity depends on the utilized architecture(s) is unknown. Such a system may predict a certain dataset as being complex, when in reality, an architecture or algorithm more suited to the evaluated dataset was not considered.

Earlier work by Ho and Baird [3] proposed a method for estimating the *intrinsic difficulty* of an image classification problem - that is, estimating the minimum achievable error dictated by the task, regardless of the applied classification system. This fundamental limit is known as the Bayes error [14]. To illustrate an instance of how and why this limit may manifest, consider a hypothetical image classification dataset for which labels were produced by human annotators. During this process, a few images from class J were accidentally duplicated and assigned to class K. Although they likely account for a minute amount of the total sample, the dataset now contains identical images with differing class assignments. Without this additional knowledge, any system, no matter its capacity, will achieve a non-zero error rate.

Labelling error is, of course, not the only potential source of inherent error. Common among any such source is that they may be described as an insufficiency of the given training data in representing the class.

(a) **(b)**

Figure 1.6: Examples of the characters generated by Ho and Baird [3] used to predict the Bayes error rate

While a task's performance ceiling is valuable knowledge, it does not provide sufficient information to fully characterize the task's difficulty. To elaborate, the intrinsic error of a problem may advise whether to continue expanding the capacity of a given system, however, is likely incapable of directing which type of classification algorithm to use or revealing behaviors which guide more computationally efficient design modifications. Moreover, using irreducible error alone is unlikely to yield accurate representations of dataset difficulty.

1.2 Contributions

We attempt to consolidate the positive aspects of both approaches noted in the literature review, seeking to estimate dataset difficulty. The following are the primary contributions of our work:

- We propose a novel algorithm for organizing images with respect to their classification difficulty, as defined by the number of principal components required to construct a space in which the latent representation of an image of class A is separated from the convex hull of latent images of another class B.
- We detail a framework for identifying human-interpretable characteristics related to the classification difficulty of particular image classes.
- We present case studies of our method applied to terrain classification on Martian terrain and Mars analogue terrain, object classification, and handwritten digit classification.We demonstrate our method's potential through comparative analysis with various deep learning models.

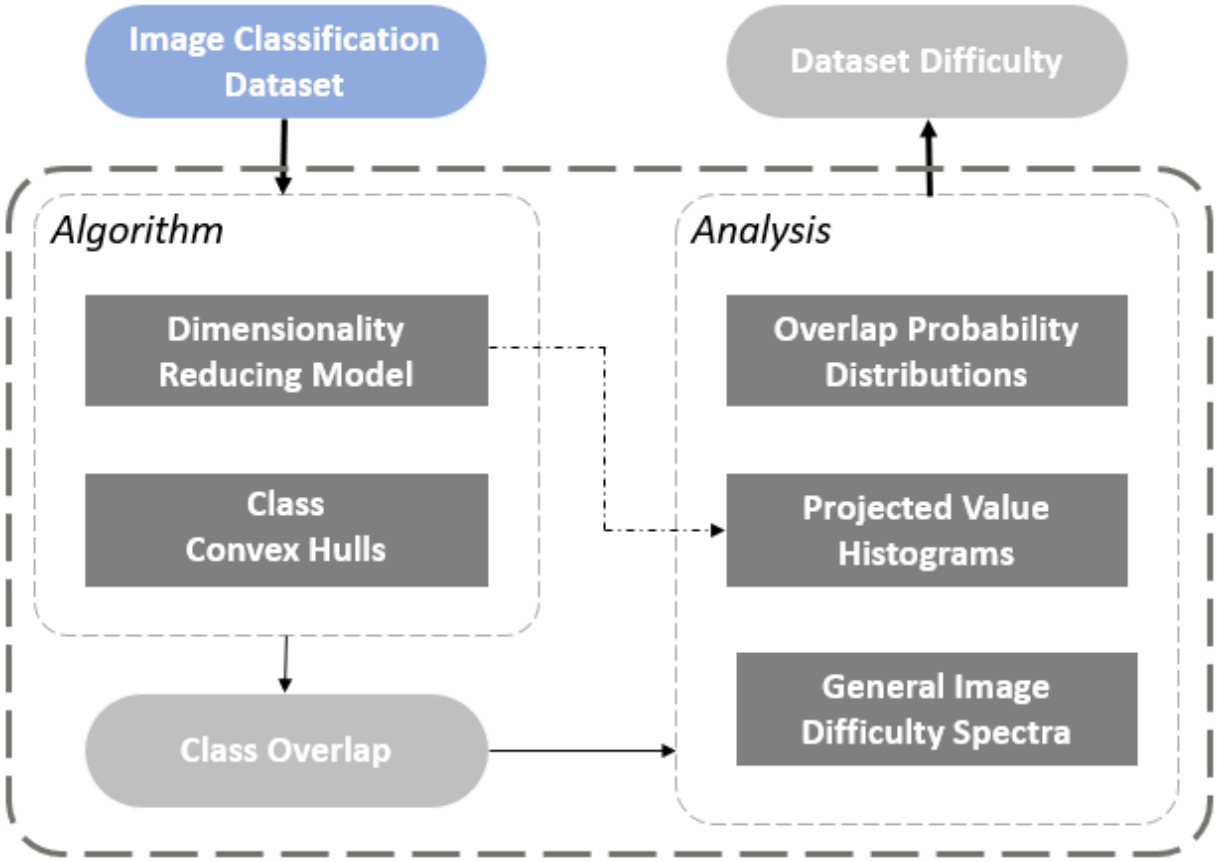


Figure 1.7: Block diagram of the proposed system. Chapter 2 discusses the details concerning the algorithm block, including its output; Chapter 3 outlines the analytical procedure developed for interpreting the algorithm outputs; Chapter 4 details the datasets applied as inputs to this system, as well as the resulting outputs.

1.3 Thesis Outline

In the next chapter, we outline the ReDiHull algorithm, its main components - a dimensionality reducing method and class model, and its output data. Chapter 3 provides detailed instructions for producing and analyzing derived data to learn characteristics of a given image classification dataset, and outlines a method to quantify difficulty between classes. Chapter 4 both illustrates and discusses select results obtained by applying the algorithm and analysis for four different image classification datasets, and where applicable, includes comparisons of statistics resulting from several deep learning architectures.

Chapter 2

ALGORITHM

Our proposed algorithm is an iterative search for a reduced dimensionality representation where all classes in a dataset are contained in non-overlapping subspaces. In this chapter, we explain the ReDiHull (reduced-dimensionality class-convex hull) algorithm underlying our method. The algorithm begins by learning a reduced-dimensionality representation of the input data (subsection 2.1.2). Within this representation, we examine each class-subspace (subsection 2.2), recording the data points contributing to between-class overlap. We repeat the process, incrementing the dimensionality of the reduced-representation, until between-class overlap is eliminated or a maximally-sized space is reached. We use the number of dimensions needed to separate out the different classes as the measure of classification complexity.

For sections within this chapter, we will use $\mathbf{X}$ to refer to the input dataset. We assume $\mathbf{X} \in \mathbb{R}^{n \times m}$ is a sequence of n number of samples $\{x_i\}_{i=0}^{n-1}$, produced by flattening images of equal height, width, and depth, into m-length vectors. We also assume $\mathbf{X}$ is normalized by subtracting the empirical mean $\bar{x} = \frac{1}{n} \sum_{i=0}^{n} x_i$ from each flattened image.

2.1 Dimensionality Reduction

Incremental analysis of the latent representation $\mathbf{Z}$ produced for input data $\mathbf{X}$ allows us to make inferential observations of both the amount and type(s) of information required to separate classes. We will denote the transformation f which maps our input $\mathbf{X} \in \mathbb{R}^{n \times m}$ to a latent representation $\mathbf{Z}_d \in \mathbb{R}^{n \times d}$ as:

$$f(\mathbf{X}, d) : \mathbf{X} \rightarrow \mathbf{Z}_d \tag{2.1}$$

In order for interpretable results to be extracted, the dimensionality reducing model and resulting representation have the following requirements:

1. Latent variables and/or the corresponding latent representations are ordered with respect to their score, given by a predefined metric,

2. The dimensionality reducing method should be reproducible, such that the order obtained in 1 is consistent between trials,

3. The change in distance between any two points produced by a change in their representation's dimensionality $\mathbb{R}^d \rightarrow \mathbb{R}^{d+1}$ must be greater than or equal to zero.

We necessitate criteria 1 and 2 to ensure the validity and robustness of the resulting method. Without these requirements, extracted characteristics cannot be confidently relied upon as they may be the result of trial-dependent factors (e.g. variables initialized using random values). As consequence of these two criteria, dimensionality reduction techniques based on convex functions are more likely to be found suitable; techniques whose computation produces a unique solution are guaranteed to satisfy these requirements.

The final requirement 3, while critical to the proposed method, is perhaps the easiest to satisfy; it is only relevant to rarely-encountered, abstract scenarios. For example, if the chosen function for (2.1) is stochastic, or the implementation does not follow the definition of a "pure function[1]," then this requirement may be of concern. This is imposed to ensure coherency of the methods results as well as to further constrain the system's outputs to being a property of the input dataset, rather than a property of the chosen method.

[1]A pure function describes a function having immutable properties: for any given succession of function calls which use identical input values, the function will return identical output values

2.1.1 Analysis of Dimensionality Reducing Methods

To help guide our selection, we began by comparing the efficacy of several dimensionality reducing methods. Considerable motivation for performing this analysis was owed to uncertainty regarding the expressive power of non-linear methods / non-linear units compared to linear counterparts.

Experiments were conducted using (i) MNIST and (ii) CIFAR-10 to train and evaluate the following dimensionality reducing models: PCA, non-negative matrix factorization (NMF), an autoencoder (HAE) with the configuration presented in [25], independent component analysis (ICA) using the logarithm of the hyperbolic cosine as the neg-entropy function, ICA using the natural exponential function as the neg-entropy function, and ICA using a third-degree polynomial as the neg-entropy function.

Shown in figure 2.1, the deep autoencoder out-performs other methods for a latent representation using fewer than thirty variables. At this point, the deep autoencoder and PCA share the same reconstruction error.

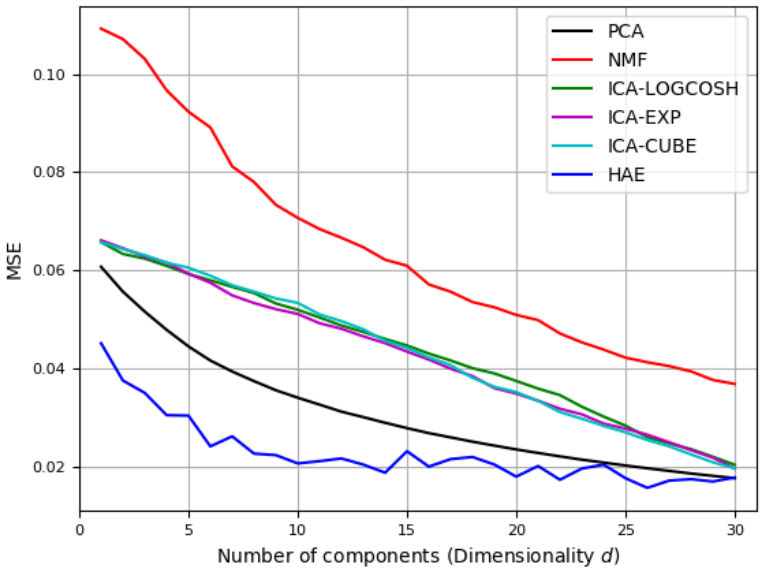


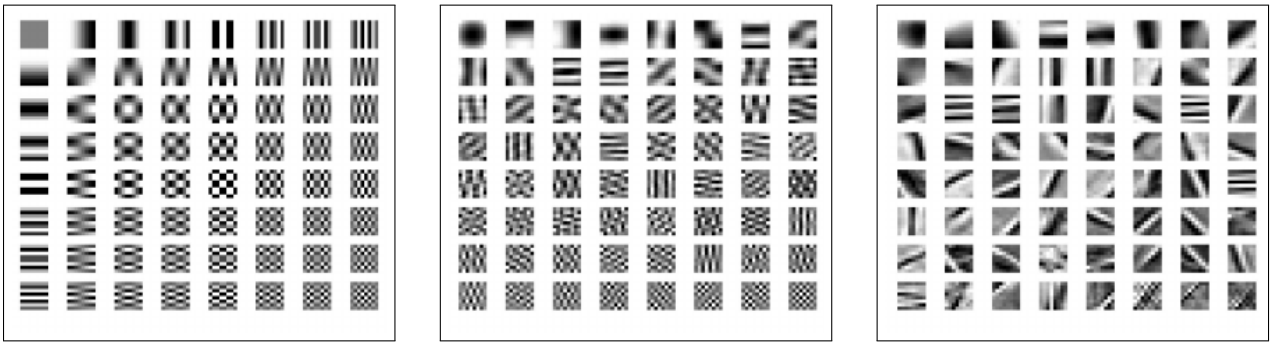
Figure 2.1: Comparison of reconstruction error per number of latent variables. Evaluated using the MNIST's test subset of 10 000 images.

While the autoencoder was found to produce a superior dimensionality reducing function,

its performance compared to PCA was not significant enough to justify its potential to introduce uncertainty. Autoencoders, like other deep learning algorithms, constitute a non-convex optimization problem; subsequent trials are not guaranteed to produce similar performance or even converge to local optima. Further, we found the networks to demonstrate capricious performance in response to the introduction of minor architectural variation.

2.1.2 Principal Component Analysis

For this study, we use principal component analysis (PCA) for dimensionality reduction. PCA is an orthogonal, linear transformation, that projects samples into a coordinate system whose axes encode the directions of greatest variation within the applied training data. PCA satisfies the first imposed requirement by ranking principal components with respect to the amount of variation they capture. Compliance with the second requirement is guaranteed by the transformation's orthogonality. PCA is one of several dimensionality reduction techniques that satisfy the requirements, and is one that is widely used and for which multiple efficient implementations exist.



(a) **(b)** **(c)**

Figure 2.2: (a) DCT basis functions, (b) PCA loading vectors, (c) ICA basis; Sourced from [26].

As PCA is the linear transform, optimal for retained variance, it is inherently useful for applications where the most important information is undefined / unknown. Being linear, we surmise that there may be non-linear methods capable of retaining more variance using fewer variables, however preliminary experiments suggest that the particular choice of

dimensionality reduction technique does not produce much difference in the context of our algorithm.

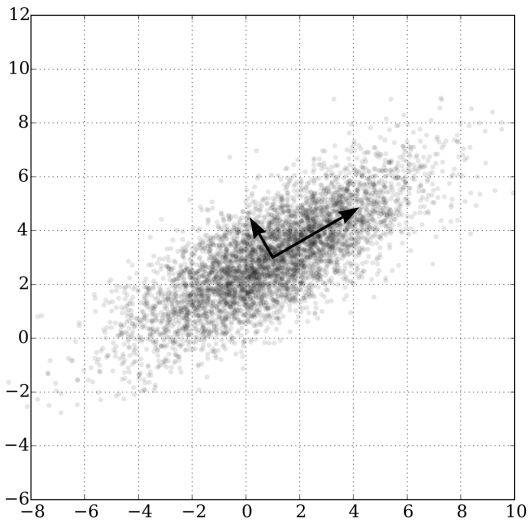


Figure 2.3: Example of the directions of maximal variance computed using PCA.

Other conceptually relevant properties lie in the ranking and types of features learned for applying PCA to image data; We observed that under this application, PCA uncovers 2D-frequency features, ranked with the lowest frequencies first. Referring to figure 2.2, we are shown that these features and rankings are in fact quite similar to the basis functions of the discrete cosine transform [26]. As consequence of this property, PCA leads to similar findings as other works, such as [18], which found that cluttered images - in other words, images with higher frequency content - resulted in more challenging tasks. An exemplar of this relation is shown in figure gained 2.4, obtained by applying PCA-based ReDiHull to CIFAR-10.

Definitions

We will denote the linear transformation obtained through PCA as:

$$\mathbf{Z} = \mathbf{X}\,\mathrm{W}^{\intercal} \tag{2.2}$$

Where each of the d columns in matrix $\mathrm{W}^{\intercal}$ represents a computed loading vector $\mathrm{w}_{(d)}$

Figure 2.4: Image samples from the CIFAR-10 *Bird* class ordered by increasing difficulty (left to right) according to ReDiHull.

with the first column as the first obtained. The number of possible loading vectors is finite, limited by either the number of samples or number of features $d \leq \min(n,\ n)$. We define this limiting value as:

$$k = \max(d) \tag{2.3}$$

In the context of (2.1), a projection onto k loading vectors is equivalent to (2.2), i.e $f(\mathbf{X}, k) = \mathbf{Z}_k = \mathbf{X}\,\mathrm{W}^\intercal$. We may also achieve a latent representation in terms of $d < k$ number of latent variables by substituting the matrix W with the submatrix obtained by removing its $k - d$ last columns, i.e. $\mathbf{Z}_d = \mathbf{X}\,\mathrm{W}_d^\intercal$ where

$$\mathrm{W}_d = \begin{bmatrix} \mathrm{w}_{1,1} & \cdots & \mathrm{w}_{d,1} & \cdots & \mathrm{w}_{k,1} \\ \mathrm{w}_{1,2} & \cdots & \mathrm{w}_{d,2} & \cdots & \mathrm{w}_{k,2} \\ \vdots & & \ddots & & \vdots \\ \mathrm{w}_{1,m} & \cdots & \mathrm{w}_{d,m} & \cdots & \mathrm{w}_{k,m} \end{bmatrix} \longrightarrow \begin{bmatrix} \mathrm{w}_{1,1} & \cdots & \mathrm{w}_{d,1} \\ \vdots & \ddots & \vdots \\ \mathrm{w}_{1,m} & \cdots & \mathrm{w}_{d,m} \end{bmatrix}$$

Another operation which will be frequently used is obtaining a reduced representation specific to a single latent variable located at index (d). We achieve this as the projection onto $\mathrm{w}\,(d)$ a single loading vector having the dth highest ranking:

$$\mathbf{Z}_{(d)} = \mathbf{X}\,\mathrm{w}_{(d)}^\intercal \tag{2.4}$$

Computing PCA

To begin, we obtain the first loading vector as the unit vector which satisfies:

$$\mathrm{w}_{(1)} = \underset{||\,\mathrm{w}\,||=1}{\arg\max}\ \left(\mathrm{w}\mathbf{X}^\intercal\,\mathbf{X}\,\mathrm{w}^\intercal\right) \tag{2.5}$$

Since $\mathrm{w}_{(1)}$ is a unit vector, (2.5) is rewritten as,

$$\mathrm{w}_{(1)} = \underset{||\,\mathrm{w}\,||=1}{\arg\max} \left(\frac{\mathrm{w}\mathbf{X}^{\intercal}\,\mathbf{X}\,\mathrm{w}^{\intercal}}{\mathrm{w}\,\mathrm{w}^{\intercal}} \right) \tag{2.6}$$

The solution to equation (2.6) is obtained as the eigenvector having the largest eigenvalue for $\mathbf{X}^{\intercal}\,\mathbf{X}$. This is satisfied when w corresponds to this eigenvector. With $\mathrm{w}_{(1)}$ obtained, we may now remove the information it captures from the input dataset, allowing us to compute $\mathrm{w}_{(2)}$. We proceed to removing the variance which has been accounted for by $\mathrm{w}_{(1)}$. This produces a new representation of the input data, denoted as $\tilde{\mathbf{X}}_d$. for which the process is repeated by replacing instances of $\mathbf{X}$ with $(\tilde{\mathbf{X}})$. i.e.

$$\tilde{\mathbf{X}}_1 = \mathbf{X} - \mathbf{Z}_{(1)}\mathrm{w}_{(1)}$$

Prior to computing each subsequent loading vector $\mathrm{w}_{(d)}$ the variance-removed representation must be updated as follows:

$$\tilde{\mathbf{X}}_d = \tilde{\mathbf{X}}_{d-1} - \mathbf{Z}_{(d)}\mathrm{w}_{(d)} \tag{2.7}$$

Where for $d = 1$, we use $\tilde{\mathbf{X}}_{d-1} = \mathbf{X}$. As a final step, we write (2.6) in a general form:

$$\mathrm{w}_{(d+1)} = \underset{||\,\mathrm{w}\,||=1}{\arg\max}\, \mathrm{w}\tilde{\mathbf{X}}_d^{\intercal}\tilde{\mathbf{X}}_d\mathrm{w}^{\intercal} \tag{2.8}$$

While the formulation given only allows for computation of single loading vectors, sequentially, methods exist permitting the computation all columns of matrix W simultaneously. In fact, recalling that the solution to (2.6) was given as the eigenvector having the largest eigenvalue for $\mathbf{X}^{\intercal}\,\mathbf{X}$ we may extend this logic, realizing that eigenvalue decomposition of the covariance matrix of $\mathbf{X}$ would yield all possible loading vectors.

It follows that the entire matrix W can also be determined using singular value decomposition (SVD), however, considering both the size of images and number of samples contained within a typical image dataset, the computation of $\mathbf{X}^{\intercal}\,\mathbf{X}$ demands a significant amount of resources which are not always available.

Moreover, we have found that for the majority of datasets applied to our algorithm, the stopping criteria is reached using a very small number of loading vectors. Computational resources can be consumed more efficiently by determining loading vectors on an as-needed basis. As consequence, we have found that solving (2.8) to obtain loading vectors sequentially

is preferable. Our implementation solves each w_d using the randomized SVD algorithm proposed by Halko et al. [27]

2.1.3 Discovering the Effects of Sample Variation using PCA

We summarize the conceptual role of PCA in our algorithm as well as introduce a visual reference leading to next section (2.2) on convex hulls through presentation of a case study: In this case study, we examine the relational effects induced by variation within class samples. This is carried out using the *rock-strewn* terrain class from the Martian terrain dataset [1]. Specific details regarding this dataset and its preprocessing can be found in 4.1.

The images shown in figure 2.5 depict four neighboring patch samples assigned to the *rock-strewn* terrain class. While they all share the *rock-strewn* terrain label, they are certainly not identical: differences in samples' horizontal and/or vertical positions has introduced within-class variation.

Given the observed variation between samples, determining its effect on the class may prove informative. Such an analysis requires (i) a method of reducing the images' dimensionality, and (ii) a technique to model the class boundaries. The combination of these will allow us to study the variation's effect on the class boundaries with respect to a few, isolated variables.

As this section has already determined PCA as a suitable answer to (i), let us begin by obtaining the reduced dimensionality representation of the samples. This is achieved by computing PCA against the entire Martian terrain training set, then transforming the samples through projection onto the resultant PCA loading vectors. Projecting the four *rock-strewn* samples from figure 2.5 onto the first 5 PCA loading vectors yields[2]:

[2]While the absolute values are conserved, projections' signs were flipped to avoid confusion within figure 2.6

$$x_1 \mathrm{W}_5^\intercal = \begin{bmatrix} 11.40 & -2.28 & 3.68 & 0.40 & -0.03 \end{bmatrix}$$
$$x_2 \mathrm{W}_5^\intercal = \begin{bmatrix} 11.22 & 2.39 & 2.99 & 0.76 & 0.17 \end{bmatrix}$$
$$x_3 \mathrm{W}_5^\intercal = \begin{bmatrix} 11.01 & -2.35 & -3.57 & 0.62 & 0.02 \end{bmatrix}$$
$$x_4 \mathrm{W}_5^\intercal = \begin{bmatrix} 11.76 & 2.13 & -2.91 & 0.78 & 0.37 \end{bmatrix}$$

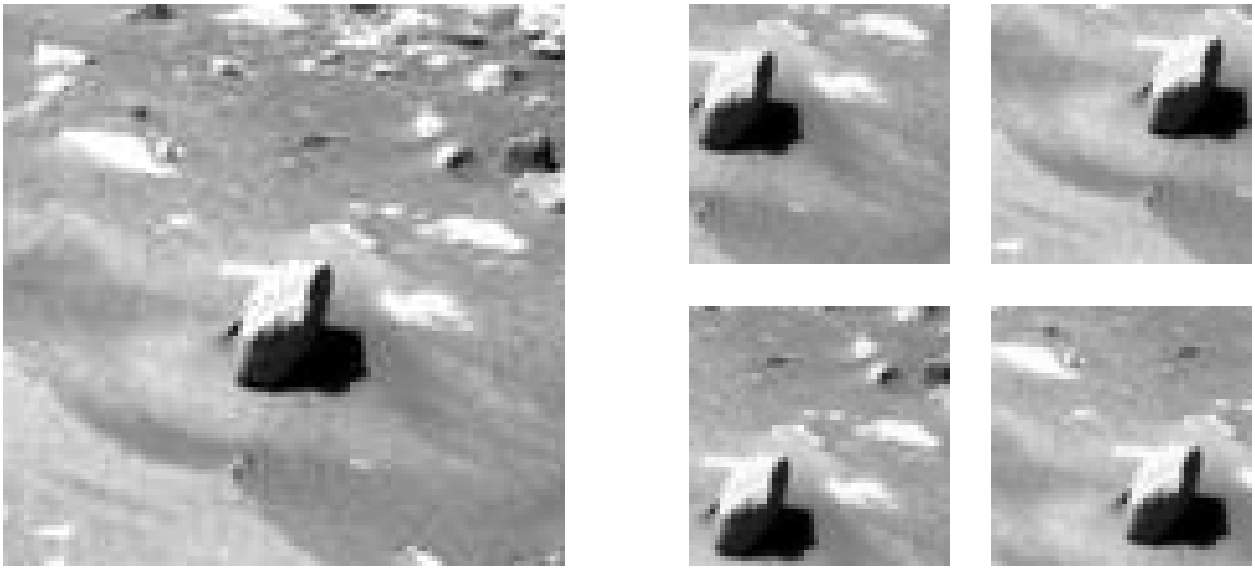
Returning to figure 2.5, a simple, visual comparison between samples allows us to associate the given action (shifted sampling positions) with an effect (qualitative differences between images). Now, we may compare samples quantitatively, allowing us to obtain a similar, cause-effect association, within the context of the images' projections. Proceeding with this comparison, we note that for the first loading vector, only minor differences between samples' projections are observed. Continuing to the second loading vector, we observe low variation within the paired samples x_1 and x_3, and similarly low variation within the pair formed by samples x_2 and x_4. In contrast, comparisons made *between* these pairs (i.e. comparing x_1 to x_2) demonstrate considerable difference.

By repeating this procedure for the remaining values, we may associate the shifts to sampling positions as affecting projected values onto loading vectors w_2 and w_3. To better visualize the identified effects, we provide the result of plotting samples with respect to these two components in figure 2.6.

The identified variation in projected values does reveal a potentially limiting property of PCA: its lack of invariance with regard to simple perturbations when applied to images. As all pixel values are considered simultaneously - that is, without concern for local inter-pixel relations - small deviations in an object's position or rotation may translate into large differences between projected samples. For many applications, this presents significant concern - PCA may be described as an optimal transform for minimizing reconstruction error, however the differences in projected values may transfer into the model's efficacy in reconstructing the original image. With that being said, while an invariant transform may produce a more efficient reduced representation, the invariance may also limit or even remove our ability to study the effects of the variation introduced. For example, replacing PCA with a technique invariant to positional changes may result in all four images having similar projections. Re-

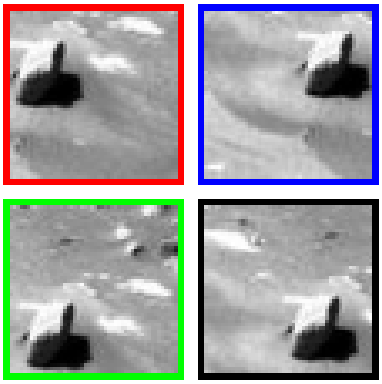
producing this example hypothetically, the use of some invariant transform might lead to the conclusion that these positional changes will have no influence on dataset difficulty as they have little impact on the projected values.

(a)

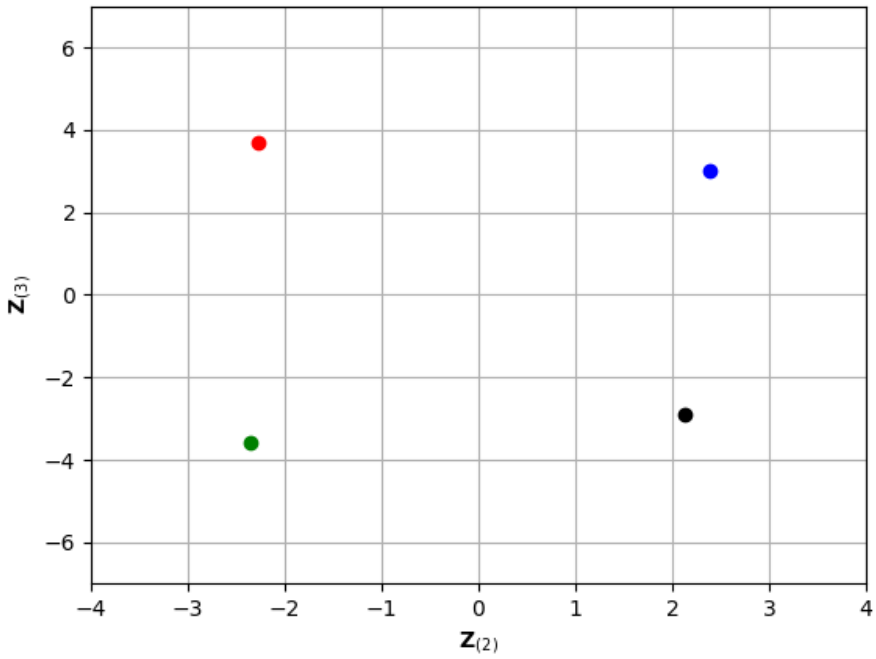


(b) **(c)**

Figure 2.5: (a) An image captured by the "Opportunity" rover on Sol 2174 depicting rock-strewn Martian terrain. (b) A zoomed-in section of the original image. This section contains (c) the four neighboring patch samples extracted from this scene.



(a)



(b)

Figure 2.6: (a) The four *rock-strewn* terrain samples and corresponding plot color. (b) The projection of the four samples onto two PCA loading vectors; projections onto w_2 are shown on the horizontal axis while projections onto w_3 are shown on the vertical axis.

2.2 Convex Hull

At each incrementally higher-dimension latent representation, the overlap (or conversely the separation) of classes is determined. We model a class-occupied subspace as the convex hull produced by its constituent samples. Employing this definition allows us to evaluate our ability to distinguish between classes within a latent space, regardless of the corresponding transformation's relation to this criterion.

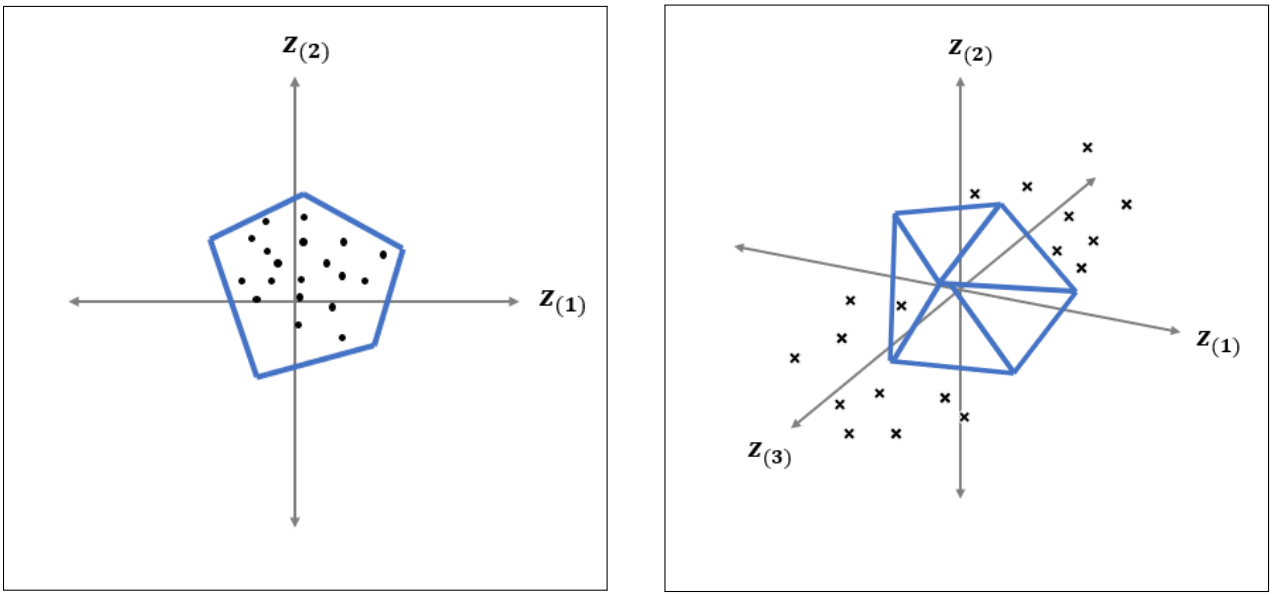


Figure 2.7: A change in dimensionality affects the overlap of classes in the latent space. The black points encompassed by the blue convex hull in a 2-dimensional space are no longer found to be overlapping with the convex hull when examined within a 3-dimensional space.

Whenever a convex-hull is referenced, we refer to its corresponding class as the *hull-class* ψ whose subset of dataset samples is given by $\mathbf{X}^{\psi}$. The convex-hull itself may be determined as the set of all convex combinations of points $x \subset \mathbf{X}^{\psi}$. As our analysis is performed exclusively on the latent representation $f(\mathbf{X})$ we provide the convex-hull's definition [28] accordingly, written in terms of $z \subset \mathbf{Z}^{\psi}$.

$$\text{Conv}(\mathbf{Z}_d^{\psi}) \equiv \left\{ \sum_{i=0}^{|\mathbf{X}^{\psi}|-1} A_i z_i \;\middle|\; \forall i : A_i \geq 0 \;\wedge\; \sum_{i=0}^{|\mathbf{X}^{\psi}|-1} A_i = 1 \right\} \tag{2.9}$$

2.2.1 Determining Overlap

Now, with a *hull-class*' role defined, the next step it to form a *class-pair* (ω, ψ) for which to compute overlap, through selection of a *reference-class* ω. It follows from (2.9) that if a point $q \subset \mathbf{Z}^{\omega}$ is not within the set of possible convex combinations produced by $\text{Conv}(\mathbf{Z}^{\psi})$, it is not contained within the convex hull.

Due to a combination of the algorithms' time complexities and inefficiencies related to implementation, we determine whether a point overlaps with a convex hull using two methods, whose selection is made based on the current value of d. We found that computing the convex hulls is done in near linear time for dimensions 1 through 5, however computation time increases rapidly when past this range. The linear programming approach follows a log-like curve, thus cumulative time increases rapidly for the lower dimensions.

$d \leq 5$ For scenarios within this condition, we determine overlap by first computing the convex hull vertices for points $\mathbf{Z}^{\psi}$, then computing the vertices of $\text{Conv}\left(q \cup \bar{\mathbf{Z}}^{\psi}\right)$, where $\bar{\mathbf{Z}}$ represents the vertex subset of $\mathbf{Z}$. If q overlaps with the convex hull of $\mathbf{Z}^{\psi}$, then it will **not** be within the set of vertices produced by $\text{Conv}\left(q \cup \bar{\mathbf{Z}}^{\psi}\right)$.

$d > 5$ While the process described for $d \leq 5$ is relatively quick for small values of d, computation times for larger dimensional spaces increases quite rapidly. As a result, we follow a different procedure for $d > 5$. In theory, we pose this as a linear programming problem of the following form:

$$\begin{aligned} &\text{Find } q \\ &\text{subject to} \quad q = \sum_{i=0}^{n^{\psi}-1} A_i z_i \\ &\text{where,} \quad \sum_{i=0}^{n^{\psi}-1} A_i = 1 \\ &\text{and } A_i \geq 0 \quad \text{for all} \quad i = 0, ..., n^{\psi} - 1 \end{aligned}$$

This may be recognized as a linear feasibility problem, where q is a point belonging to the projected reference class ω and z_i is a point belonging to the projected hull class ψ. To implement this in practice, we formulate a linear optimization using the interior point method described by [29]:

$$\begin{aligned} &\text{Minimize} && \mathbf{c}^{\intercal}\mathbf{A} \\ &\text{subject to} && \mathbf{A}\hat{\mathbf{z}} = \hat{\mathbf{q}} \\ &\text{and} && \mathbf{A} \geq 0 \end{aligned}$$

However, we do not need to perform the optimization; we want to determine the feasibility of the problem with regards to its constraints. Feasibility indicates whether or not the point is within the convex hull. The variable $\mathbf{c}$ is set to zero as it can be any arbitrary vector.

$$\hat{\mathbf{z}} = \begin{bmatrix} z_{1,1} & z_{2,1} & \cdots & z_{N,1} \\ z_{1,2} & z_{2,2} & \cdots & z_{N,2} \\ \vdots & \vdots & \ddots & \vdots \\ z_{1,m} & z_{2,m} & \cdots & z_{N,m} \\ 1 & 1 & \cdots & 1 \end{bmatrix} \text{ where } z_{i,j} \in \mathbf{Z}^{\psi}$$

$$\hat{\mathbf{q}} = \begin{bmatrix} q_{1,1} & \cdots & q_{1,m} & 1 \end{bmatrix} \text{ where } q_{1,j} \in \mathbf{Z}^{\omega}$$

While the linear programming formulation spares us from having to compute the convex hulls, feasibility is determined at minimum $|\{\mathbf{X}\} - \{\mathbf{X}^{\psi}\}|$ times for each class. We have found this is prohibitive for datasets containing many classes, where each class is comprised of many points. To overcome this challenge, we reduce each set of hull-points $\mathbf{Z}^{\psi}$ to its set of vertices, relevant to each of the examined projective spaces.

The use of convex hulls for dataset modelling has seen success in various classification and image processing solutions [30][31][32][33]. Given the role of convex hulls within our outlined technique, it is important to bring attention to nearest convex hull (NCH) classifiers. Introduced in [34], these types of systems consider classes as the representation gained by their convex hull(s), allowing unlabelled samples to be assigned to their nearest convex hull. As result, NCH classifiers share a high degree of conceptual overlap with both k-nearest

neighbor and support vector machines (SVM) algorithms. In fact, the formulation of SVMs and convex hulls are so closely related that the SVM separating plane for a binary class problem can be interpreted geometrically as the plane located at the midpoint of the two.

2.2.2 Determining Vertices

Here we describe the process employed for determining the convex hull vertices of a class. In short, this method determines class vertices simply by following the routine described in section 2.2.1, using the input class pair for which the hull-class has the same label as the reference-class i.e. $\psi = \omega$. At a given dimensional space, the samples within this *self-test* determined non-overlapping are equivalent to the convex-hull vertices within that space. Furthermore, samples found to be vertices at some dimension d are also vertices for all subsequent dimensions.

Now, it may be apparent that if the hull-class and reference-class share the same definitions, then for any $x_i \subset \mathbf{X}^{\omega}$ it is also true that $x_i \subset \mathbf{X}^{\psi}$. As consequence, it will be impossible to separate any x_i from the convex-hull Conv $\left(\mathbf{X}^{\psi}\right)$, regardless of their representations. To circumvent this, we calculate overlap for each x_i using the convex-hull which has this specific point removed from its hull-defining set of points Conv $\left(\{\mathbf{X}^{\psi}\} \setminus x_i\right)$.

2.2.3 Special Case for Vertices - Class Generalization

While described in the context of vertex determination, following this procedure allows us to simultaneously obtain statistics regarding the class' generalization. Discussed in further detail later on, determining overlap at each dimension using a ***self-test*** $\psi = \omega$ provides the ability to gain a sense of how generalized a class is. The premise is that the effort required to separate an image from its own convex-hull relates to the class' generalization.

It should be noted that some aspects concerning the class generalization are still not fully understood; as we are redefining the hull for each tested point Conv $\left(\{\mathbf{X}^{\psi}\} \setminus x_i\right)$ we are guaranteed to have separable points, no matter how well our training points represent the class. Now, while this is the correct behavior for extracting vertices, it is less desired for obtaining measures of class generalization. Consider we instead had two subsets for each class

- one for defining the hull and another for use as reference points. Under the scenario $\psi = \omega$ we would have $|\mathbf{X}^{\psi} \cup \mathbf{X}^{\omega}| = 0$ thus the convex-hull may be defined consistently between each tested $x_i \subset \mathbf{X}^{\omega}$. The counter argument to this, however, is that partitioning class examples into two distinct sets will reduce the reliability of each class' extracted statistics through decreased number of samples. To elaborate, the subset of points which define $\mathbf{X}^{\psi}$ will now be smaller than it was previously. Additionally, the same class' subset of reference points $\mathbf{X}^{\omega}$ will also have a reduced number of samples.

While there are likely other methods capable of modelling the class-subspaces with higher fidelity, the technique itself is not meant to be a classifier - it is meant to highlight modes of potential difficulty which may be used to design a classifier. Selecting a more advanced method may dampen the effects or possibly eliminate them; the consequence being the loss of ability to associate such flaws with the original data. By using a naive method, the underlying issues are more likely to be kept in tact, thus allowing the ability to identify whether or not they are present.

2.3 ReDiHull Algorithm

The previous sections discussed the two central pieces of our ReDiHull algorithm: a dimensionality-reducing technique and class convex hull model. Here, we summarize the algorithm, allowing for understanding of its output.

As the dimensionality of the reduced-dimension representation is incrementally increased, between-class overlap can decrease, as illustrated in figure 2.7. The number of dimensions needed to separate out the different classes is a measure of classification complexity.

ReDiHull is summarized in pseudo-code, shown as algorithm 2.8. We have as input (i) a reference-class subset, either training or testing, and (ii) a hull-class subset. We initialize the algorithm at dimension $d = 1$, with the set of overlapping points initialized with the entire reference-class subset, i.e., $\mathbf{\Theta}_{\text{o}}^{(\psi\,\omega)} = \mathbf{X}^{\omega}$. The algorithm is presented in a form which produces full enumeration of overlaps at each d for a given class pair, by operating recursively.

Application of the algorithm results in the sequence $\mathbf{\Theta}$ which stores the computed sets of overlapping points in order of increasing space-dimensionality. From this definition, it appears that for $d = 0, ..., k$ we have $\mathbf{\Theta}_d \subseteq \mathbf{X}^{\omega}$ thus for a worst case scenario translates to storing k copies of the $\mathbf{X}^{\omega}$ subset of images. This is an inefficient allocation of resources and should be avoided in practice. Instead, it is much more desirable to implement $\mathbf{\Theta}$ as containing the *indices* of overlapping samples from $\mathbf{X}^{\omega}$ at each dimension.

While algorithm 2.8 is presented as a binary-class problem, expanding its use is quite simple; in order to apply the algorithm to datasets having more classes, one need simply apply ReDiHull to all possible class-pair permutations. The order in which class-pairs are analyzed does not matter in general, unless "reduced" convex hulls are used.

Recall that a convex hull may be reduced to its vertices without loss of accuracy - we leverage this property to decrease computational burden. Given a class ψ we may obtain its reduced convex hull as its subset of convex hull vertices within a d-dimensional representation. Under this circumstance, the selection of any ψ which constitutes a first-time occurrence should analyze the class-pair satisfying $(\psi, \omega) = (\psi, \psi)$ before any other. In other words, whenever a certain class "J" is selected to serve as the hull-defining class and the class "J" has never been selected as the hull-defining class in previous iterations, then

Algorithm 1: ReDiHull Algorithm

Input: $\boldsymbol{\Theta}_{d-1}^{(\psi\,\omega)},\ \mathbf{X}^{\psi},\ d$

begin

$\mathbf{Z}_{d}^{\psi} \longleftarrow f\left(\mathbf{X}^{\psi}, d\right)$

$\boldsymbol{\Theta}_{d}^{(\psi\,\omega)} \longleftarrow \left\{q \mid \forall\, q:\ q \subset \boldsymbol{\Theta}_{d-1}^{(\psi\,\omega)} \text{ and } f\left(q, d\right) \in \text{Conv}\left(\mathbf{Z}_{d}^{\psi}\right)\right\}$

if $\left|\boldsymbol{\Theta}_{d}^{(\psi\,\omega)}\right| > 0$ **and** $d < k$ **then**

ReDiHull $\left(\boldsymbol{\Theta}_{d}^{(\psi\,\omega)},\ \mathbf{X}^{\psi},\ d+1\right)$

end

end

Figure 2.8: ReDiHull algorithm presented for a binary classification task.

the reference-class should be chosen as class "J" as well.

Chapter 3

ANALYSIS

The ReDiHull algorithm, detailed in chapter 2, awards us $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_0, ..., \boldsymbol{\Theta}_d\}$ the sets of points from class ω which overlap with hull-class ψ in each examined latent space. This section outlines the procedure(s) applied to the algorithm's output, allowing us to uncover valuable characteristics hidden within the given dataset - characteristics such as those lending themselves to increased difficulty in distinguishing an ω-labelled image from the images of class ψ.

While the analysis will produce identical results when applied to output as is, we have found it to be conceptually advantageous to apply a simple transformation such that we instead examine the change in sets of overlapping points with respect to the change in dimensionality of the projective space. For a set of overlapping points at dimensionality $d-1$ and set of overlapping points at dimensionality d the change is simply the difference between the two sets:

$$\begin{aligned} \boldsymbol{\Xi}_d = \quad & \boldsymbol{\Theta}_d - \boldsymbol{\Theta}_{d-1} \\ \text{where} \quad & \boldsymbol{\Theta}_{d-1} = \{\emptyset\} \quad \text{for } d = 0 \end{aligned} \tag{3.1}$$

We refer to subset $\boldsymbol{\Xi}_d$ as the points which were *separated* at dimensionality-d. All visualized statistics, described in the sections 3.1.1 through 3.1.3, are obtained from the transformed algorithm output given by (3.1).

3.1 Analytical Procedure

We begin by using the output to create empirical probability distributions, later analyzed to identify regions of potential class characteristics. Next, we attempt to explain behaviors occurring within these regions by examining the latent variables and transformed image values. Where applicable, we visualize regions' constituent images, allowing for qualitative validation of extracted characteristics. In summary, we identify the dimensionality ranges *where* potential characteristics occur, we then determine *why* they occurred and *what* characteristic they relate to. Finally, we verify the resulting observations produced to describe general, class-pair difficulty.

3.1.1 Identifying Characteristics' Locations

Here we define the distributions representing the empirical probability of a point being separated at a given dimension. Following this definition, we explain the types of phenomena that we seek to identify within these distributions.

Now, let us consider a situation in which we are given $\mathbf{\Xi}_d$ the subset of test class ω's points which were separated from the convex hull formed by class ψ using a d-dimensional space. This subset $\mathbf{\Xi}_d$ corresponds to a fraction of class ω's total points and as such, this value is the empirical probability that a random point belonging to class ω, which overlaps with the convex hull formed by class ψ in a $d-1$ dimensional space, will be separated by incrementing to a d-dimensional space:

$$P\left(q \notin \text{Conv}(\mathbf{Z}^{\psi}) \mid d,\ q \subset \mathbf{Z}^{\omega}\right) \;=\; \frac{|\,\mathbf{\Xi}_d\,|}{|\,\mathbf{X}^{\omega}\,|} \qquad \text{where } \bigcup_{d=1}^{d=D} \mathbf{\Xi}_d = \mathbf{X}^{\omega} \tag{3.2}$$

By repeating this over each dimensional space for which overlap exists, we obtain an empirical separation probability distribution for a given class pair such as those in figure 3.1. It is important to note that this probability distribution gives a measure of dataset complexity. Distributions with long tails extending to high values of d are capturing complexity that requires many dimensions to separate out.

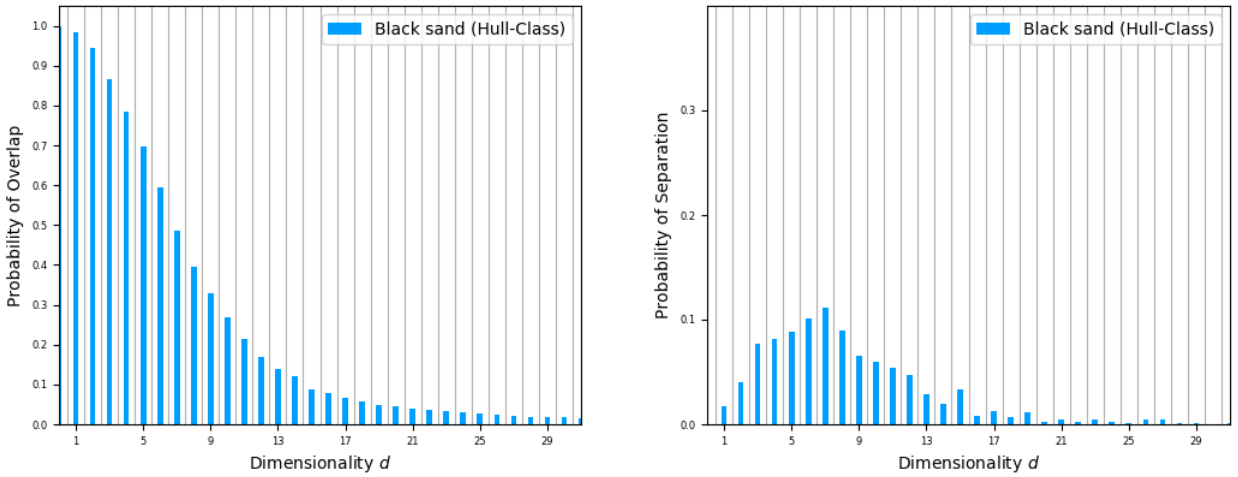
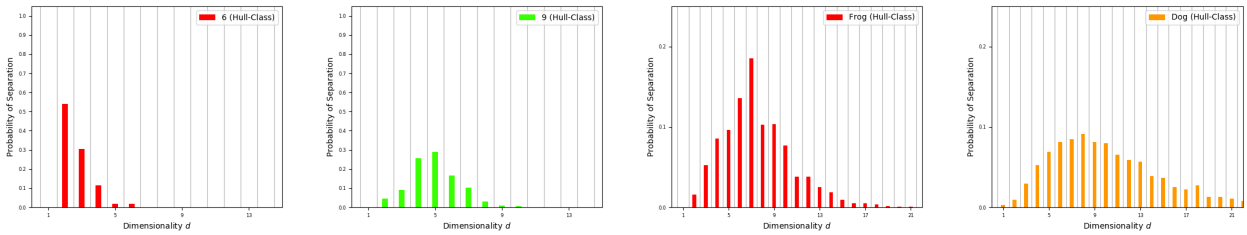


Figure 3.1: Example distributions describing (Left) the probability that a sample from a reference-class overlaps with the convex hull of a given hull-class at a given dimension; (Right) the probability that a sample from a reference-class separates from the convex hull of a given hull-class at a given minimum number of dimensions.

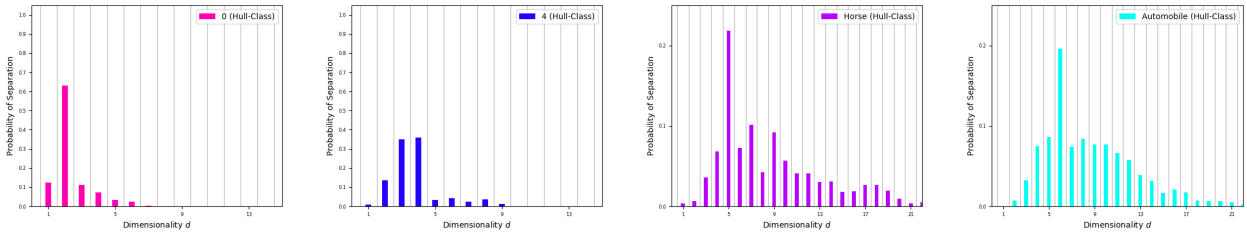
Continuing in the context of figure 3.1, we note an underlying primary distribution which is uni-modal and spans the full range of dimensions shown. From our experiments, we have observed that this type of uni-modal shape occurs for the majority of image classes, albeit with slight variations.

In order to ascertain regions related to meaningful characteristics, we identify abnormal phenomena within the obtained distributions. Abnormality is determined relative to the primary distribution - a primary distribution being that which one would expect had the abnormality not occurred. As reference, such distributions are presented in figure 3.2. To contrast the distributions from figure 3.2, distributions containing abnormal behaviours are shown within figure 3.3. In the context of figure 3.3(c), we note that at the fifth dimension an acute spike in separability relative to the primary distribution is observed. We have found that these types of spikes are particularly useful in determining *separating characteristics*; naturally-occurring characteristics which invoke a high degree of separability between respective image classes.



(a) **(b)** **(c)** **(d)**

Figure 3.2: Examples of distributions without abnormal behaviours.



(a) **(b)** **(c)** **(d)**

Figure 3.3: Examples of distributions which contain abnormal spikes, indicating the possibility of valuable separating features.

3.1.2 Determining Characteristic-Related Image Attributes

In the previous step, we defined a separation probability distribution, then summarized trends and phenomena occurring within these distributions that may relate to potentially valuable dataset characteristics. The following step allows us to discover the attributes tied to the identified regions of interest. In order to achieve this goal, we construct projected value histograms from which we examine the latent model applied to samples forming the probability distribution.

For any determined probability distribution, there is a set of separated samples associated with each dimension, $\boldsymbol{\Xi}_d$. Now we analyze the transformed values $f(\boldsymbol{\Xi}_d)$ at each dimension. As an example, studying the transformed values might allow us to learn that a class' behavior at a certain dimensionality d is due to a strong, positive correlation with the kth latent variable[1], relative to the other classes. While this allows us to understand why the abnormality occurred, the supplementary information provided by visualizing the latent

[1]In the case of PCA, the kth latent variable is the kth principal component.

variables allow us to describe the image characteristic it relates to.

Description of the Projected Value Histograms

In order to enable visual analysis of the distributions of projected values as a function of projected-space dimensionality, we were required to develop a somewhat unique visualization. As earlier discussed, studying the changes in PCA projected values with regard to changes in overlap may be of considerable value. Visualization of 1-D projected values for a single set of points is easily achieved using histograms or scatter plots. Higher dimensional data and/or comparison between multiple sets of samples is also possible using similar techniques, e.g. plotting the 2-D histograms adjacent to each other within a 3-D space, or using a 3-D mesh approximation. In attempting to apply these to our own task, we found the depth as limiting to our analyses due to the information it obscured. Dividing the results into multiple figures was somewhat of an improvement, however, it introduced the need to study hundreds to thousands of additional figures.

Responding to these issues, we developed visualizations which are conceptually tantamount to plotting the histograms adjacent to one another along a shared, perpendicular axis, producing a 3-dimensional "histogram stack," then rotating the stack about this shared axis such that the histograms' vertical axis is now perpendicular to the page. An example of the resulting visualization is presented in figure 3.4.

Now, specific details concerning these plots are discussed, using figure 3.4 as a contextual reference. Each plot corresponds to projections onto a single latent variable at index (d) indicated on the far-left. Within the referred figure, we are shown projections onto the latent variable at index (7). For this single latent feature, the maximum and minimum values, $\max\left(\mathbf{Z}_{(d)}\right)$ and $\min\left(\mathbf{Z}_{(d)}\right)$, are determined from the training set, then used to create representations of the positive and negative extremal observations, indicated by the tip of the upwards and downwards facing arrows respectively. The values may be seen by examining the chart's vertical axis as the maximum and minimum values are also used to define the histogram.

We partition the linear range formed by these extremal values into 40 bins of equal width. Application of a histogram allows for the 3D topology generated by the distribution

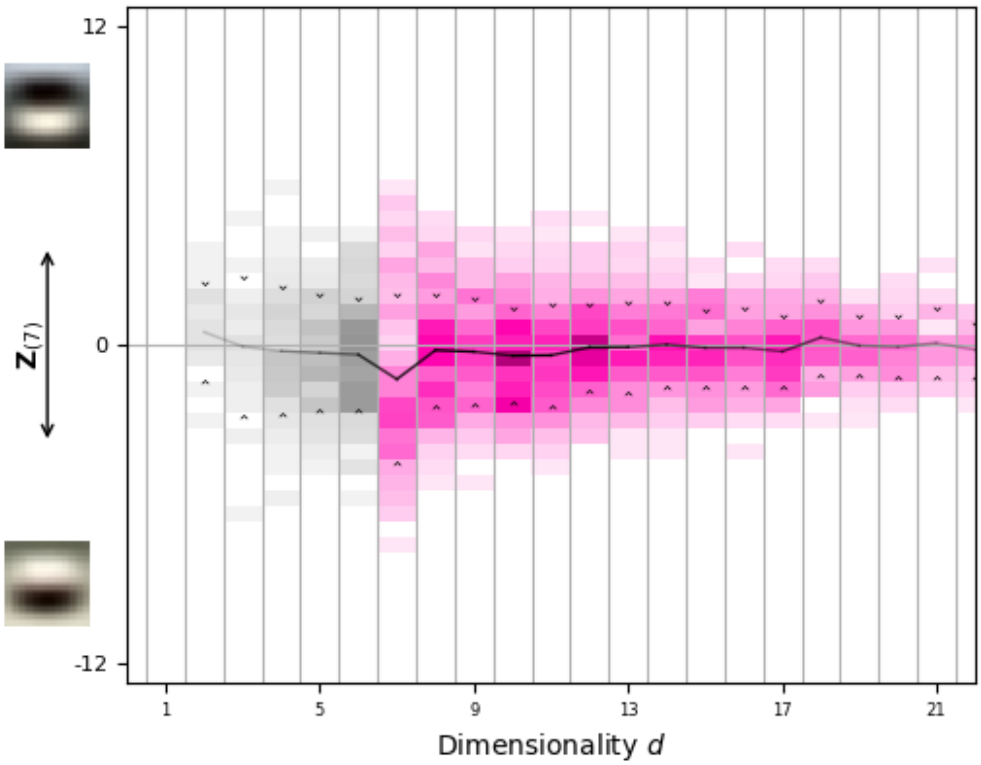


Figure 3.4: Example of visualizations developed, allowing for analysis of sequential histograms within a compact figure.

of projected values to be visualized within a 2D space. Densely-distributed regions are shown using darker-colored bins whereas bins containing no elements are white. While each plot corresponds to projections onto a single latent feature, a plot's horizontal axis may extend to dimensionalities for which projected values $\mathbf{Z}_{(d)}$ have no contribution to separability. To distinguish between dimensionalities less than d and greater than or equal to d, histogram data is shown in grayscale for the former-defined range. Within the context of figure 3.4, although we are shown projections relevant to the 7th component, we are shown projections shown for samples separated prior to the inclusion of this component, as well as those which precede it. The black piecewise-linear trendline(s) show the change in mean of projected values between $d-1$ and d for the projection of separated images Ξ_d. The opacity of a curve segment is proportional to the number of points separated at d, with a higher opacity indicating more points separated. Trends that may be present and thus captured are also considered class characteristics.

The standard deviations of projected values at each dimension are displayed as well, with downwards pointing chevrons denoting one standard deviation above the mean.

3.1.3 Validating / Visualizing Determined Characteristics-Related Image Attributes

When possible, we have found it beneficial to produce a visualization of the determined characteristic(s) using the images which form the abnormality or trend which was used to identify them. The qualitative information gained through these visualizations allows us to describe characteristic behaviors with greater certainty. Often, the combination of number of samples and dimensionality range for which the behavior is defined impedes our ability to extract useful information when all images are viewed simultaneously. To overcome this challenge, we substitute each dimension-specific image subset with a corresponding generalization (e.g. mean).

Description of the Image Difficulty Spectra

(a) **(b)**

Figure 3.5: Examples of difficulty spectra which summarize difficulties in discerning (a) *sand* samples from *sand* bedrock samples; (b) digit Nine samples from digit Seven samples, from the CSA MET and MNIST datasets respectively.

Once again, the data forming these figures extends from the obtained class pair sets of separated samples associated with each dimension. To obtain a generalization $\bar{x}$ of the images separated at each dimension, we simply compute the mean of the $n = |\boldsymbol{\Xi}_d|$ number of images within each set $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. An example of this these types of visualizations is shown in figure 3.5.

3.2 Example Application of Analytical Procedure

Here we summarize the analytical procedure through description of a case study. In this example, we detail the analysis of results obtained from the MNIST digits dataset. More specifically, we utilize the results obtained from selecting digit *Zero* as the reference class and digit *Eight* as the hull class.

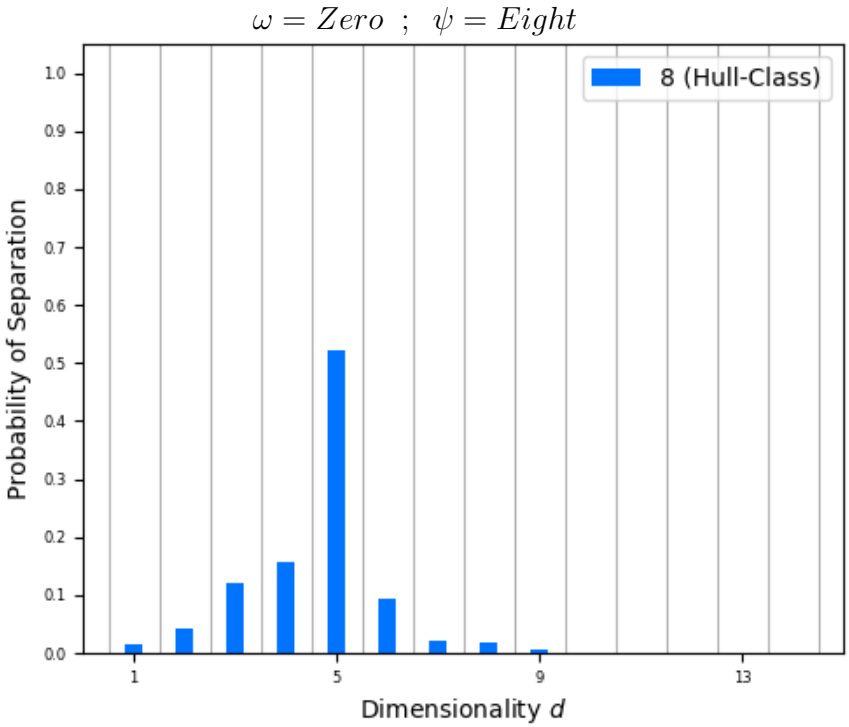


Figure 3.6: Probability distribution that a digit Zero sample will separate from the convex hull of digit Eight, given a dimensionality d and the knowledge that the sample was overlapping in a $d-1$ dimensional space

As previously discussed, we begin our analysis by examining the empirical distribution $P\left(q \notin \text{Conv}\left(\mathbf{Z}^{\psi}\right) \mid d,\ q \subset \mathbf{Z}^{\omega}\right)$ from (3.2) which describes the probability that some random image q from class ω will require d dimensions to separate from the convex hull formed by ψ, given that samples from both classes are represented in terms of the top d number of latent variables. The probability distribution obtained for separating *Zeros* from the convex hull of digit *Eight* is shown in figure 3.6.

We search figure 3.6 for trends or anomalous behavior. From these results, it would seem

that there is a main, uni-modal curve over dimensions 1 through 9. Relative to this uni-modal curve, there is a significant rise in separability at the fifth dimension. This observed spike constitutes an anomaly, which possibly indicates some information relevant to the task of distinguishing digit *Zero* from *Eight*.

In summary, we would like to produce explanations relevant to (i) the uni-modal trend between dimensions 1 through 9, and (ii) the spike at dimension 5. To achieve this will require use of the histograms of projected values shown in figures 3.7 and 3.9.

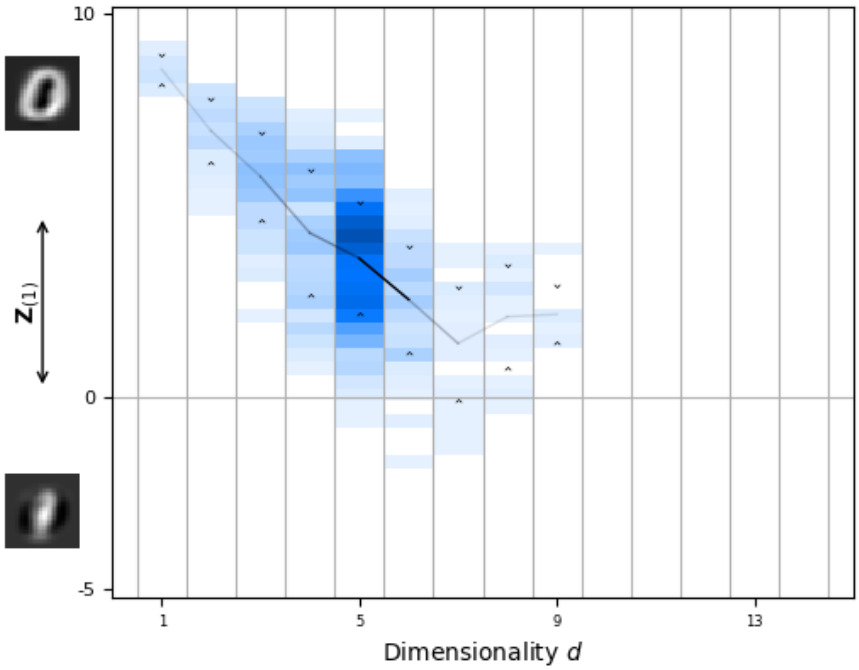


Figure 3.7: Projected value histograms for the 1st component using the MNIST digit Zero as the reference-class and digit Eight as the hull-class. As noted, the presence of a trend helps to predict the change in samples' visual characteristics with increasing sample difficulty.

Starting with 3.7, we note the negative, linear trend. This trend indicates that as images of digit *Zero* become more difficult to distinguish from digit *Eight*, they generally have a lower projected value with respect to the latent variable shown to the left. This observation provides partial explanation for the main curve identified in figure 3.6. Also of interest is that the trend seems to dissipate for $d \geq 6$. Interpreting the latent variable as an attribute suggests that as the difficulty of this task increases, *Zeros* will become progressively thinner, as $\mathbf{Z}^{\omega}_{(1)}$ has a large, positive value at $d = 1$ which decreases as d increases. Moreover, we

expect that for $d \geq 6$ this trend will no longer apply.

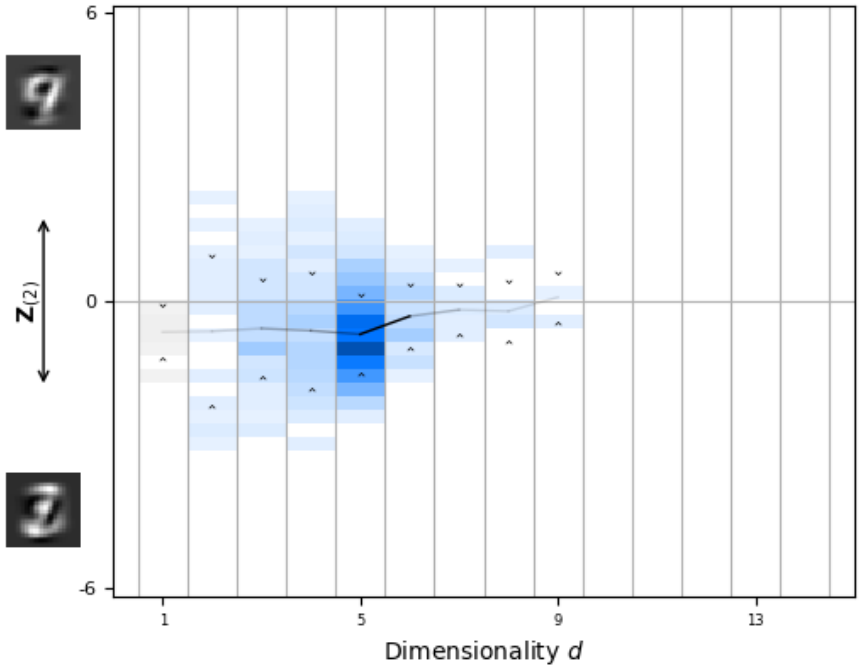


Figure 3.8: Projected value histograms for the 2nd component using the MNIST digit Zero as the reference-class and digit Eight as the hull-class.

A contrasting example is shown in figure 3.8. It can be seen that between $d \geq 2$ and $d \leq 4$ there is no significant difference in projected values of separated samples, with respect to this latent variable; this latent variable is at index $(d) = 2$ thus we expect any related behavior to become apparent for $d \geq 2$ however a trend only appears when $d \geq 5$.

To explain the anomalous spike identified at $d = 5$ we refer to figure 3.9 as it displays the histograms of separated samples' projected values for $(d) = 5$. It appears that the sharp rise in separability occurs as a result of this latent variable. We draw this conclusion by noting the clear difference of $\mathbf{Z}^{\omega}_{(5)}$ when comparing between samples separated at $d \leq 5$ and $d \geq 6$. Interpreting the latent variable, it would seem that more positive values correspond to vertical *Zeroes* whereas more negative values indicate examples written at a more horizontal angle. In this context, a "vertical" *Zero* describes an ellipse whose longer radius aligns parallel with the vertical axis, while its shorter radius aligns with the horizontal axis.

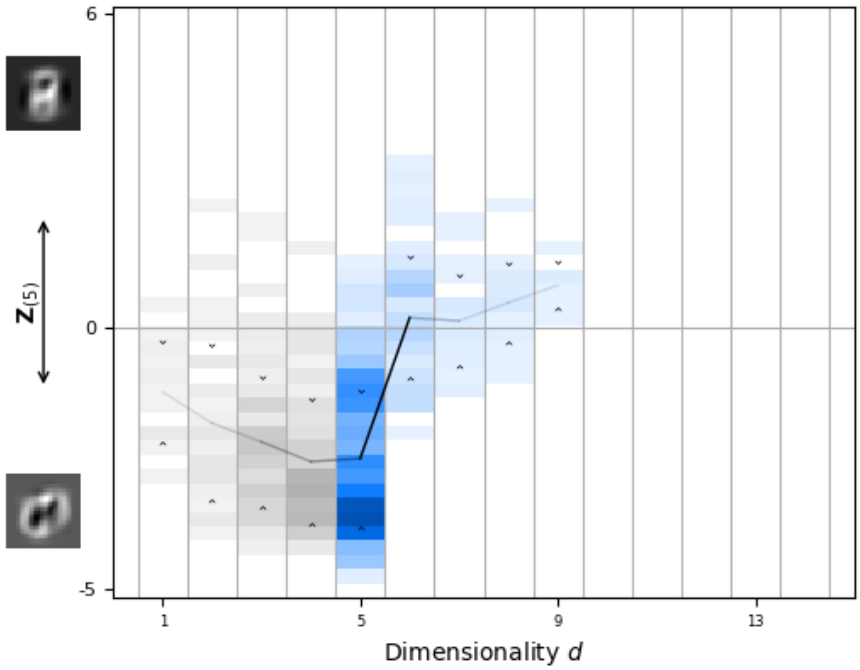


Figure 3.9: Projected value histograms for the 5th component using the MNIST digit Zero as the reference-class and digit Eight as the hull-class. The behaviour illustrated suggests the 5th component is as separating feature.

Summarizing all gathered information, **we expect that as task difficulty increases**, the *Zeros* depicted within images will:

- Appear increasingly **thinner** until $d \geq 6$ when this attribute is no longer relevant.
- Appear to be drawn more **angled or horizontal** for $d \leq 5$.
- Appear to be drawn more **vertical** for $d \geq 6$.

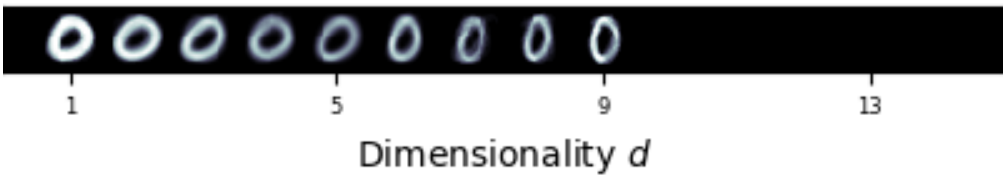


Figure 3.10: Difficulty spectra; these figures summarize modes of difficulty between classes. This visualization depicts increasing difficulty of digit Zero samples (true label / refernce class) w.r.t digit Eight samples (predicted label / hull class) computed by ReDiHull

Now as a final step, we attempt to verify the validity of these extracted characteristics using the difficulty spectrum shown in figure 3.10. Using this visualization, it appears that our assessment is acceptable: the **less difficult** *Zeroes* are thicker, more circular, and horizontally oblong, whereas **more difficult** examples are thinner, less circular, and vertically oblong.

3.3 Quantifying Difficulty Between Classes

Thus far, discussion of our technique has focused on extracting qualitative characteristics related to difficulty. To enable comparison with classifier performance, we propose quantifying dataset difficulty with the Kullback-Leibler divergence between separation probability distributions.

Our analysis procedure seeks image characteristics that promote between-class separability (i.e. separating features) in the distributions outlined above, corresponding to spikes in separability. The procedure described distinguishes a single reference class from a single hull-class, thus observations resulting from this procedure are specific to particular reference-hull-class pairs. It is worth noting that due to asymmetry between class convex hulls, a characteristic shown to help distinguish class "A" from class "B" may not be as helpful to distinguish B from A. Further, a characteristic that aids in distinguishing class A from class B may also reduce our ability to associate class B samples with itself, due to the characteristic's effect on class B's convex hull. The resulting practical implications are that a measure which predicts the ability to distinguish a reference class from a hull class should simultaneously consider (i) the probability distribution of separating a reference class from a hull class and (ii) the probability distribution of separating the hull class' samples from its own convex hull.

We explore the use of Kullback–Leibler (KL) divergence to achieve a measure instilled with these considerations. KL divergence is described as a measure of relative entropy between two distributions; given two distributions P and Q, their KL divergence $D_{\mathrm{KL}}\left(P \,||\, Q\right)$ provides a summary of the dissimilarity between them. It is computed as:

$$D_{\mathrm{KL}}\left(P \,||\, Q\right) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \tag{3.3}$$

Redefining (3.3) in the context of our work yields:

$$D_{\mathrm{KL}}\left(P \,||\, Q\right) = \sum_{d} P\left(x \notin \mathrm{Conv}(\mathbf{Z}^{\psi})\right) \log\left(\frac{P\left(x \notin \mathrm{Conv}(\mathbf{Z}^{\psi})\right)}{P\left(x \notin \mathrm{Conv}(\mathbf{Z}^{\omega})\right)}\right) \tag{3.4}$$

In which $x \in \mathrm{Conv}(\mathbf{Z})$ within a $d-1$ dimensional space, and x shares the same label as ω but is not a member of the samples used to define the ω-class convex hull, i.e. $x \notin \mathbf{Z}^{\omega}$. It

should be noted that the distributions applied to (3.4) are those defined by (3.2).

If either distribution contains zero-valued entries, computation will produce $D_{\mathrm{KL}} = \infty$. To overcome this challenge, we apply additive smoothing to all probability distributions, ensuring no points have zero probability. We added a pseudo-sample to each entry, emulating a dataset in which at least one sample was separated at each dimension, for any class pair. The change in overlapping points at each dimension (3.1) is updated such that $|\boldsymbol{\Xi}^*| = |\boldsymbol{\Xi}| + 1$ where $|\cdot|$ denotes the cardinality of the set.

Another common approach to circumventing this issue is to skip any entries that would produce a value of $D_{\mathrm{KL}}(P \,||\, Q) = \infty$ however we found this solution to produce inaccurate results. In the context of our work, zero-valued entries may correspond to features responsible for separating a reference-class from a hull-class without sacrificing the integrity of the reference-class' convex hull.

Using the smoothed probability distributions as inputs, we compute the KL divergences for class pairs, and these results are compared to a corresponding confusion matrix of a baseline classifier's performance; results gained using CIFAR-10 are presented in section 4.3, while others may be found in appendices A.2, B.2, C.1, and D.2.

Pairwise class difficulty can be extended to an N-class datasets through the construction of a matrix of pairwise class-difficulties, D, $d_{i,j} = D_{KL}(Class_i||Class_j)$. Given this matrix, we could then apply a matrix norm to $||M|| = ||D + D^T||$, a symmetric matrix, which should get smaller as the separation difficulty increases. We could also take the determinant of M, which should likewise decrease in value as the dataset gets more difficult to separate.

Chapter 4

RESULTS

To demonstrate the utility of ReDiHull, we examine results generated for two terrain classification datasets: the Martian Terrain dataset [1] and a new CSA MET dataset [35], as well as two general image classification datasets: MNIST handwritten digits [36], and the small-scale, object image dataset CIFAR-10 [37]. For each dataset, we present results from two source categories: results obtained using our proposed technique and results obtained through training and evaluating several deep learning architectures.

For results obtained using the proposed technique, we highlight specific class analyses on these datasets in order to illustrate the outlined technique for improved clarity of its application and demonstrate concise examples of the types of extracted characteristics and their potential utility. Most importantly, these results demonstrate the ability to discover image-characteristics of recognition difficulty which may not generally be presumed to cause challenges.

Each dataset was partitioned into three subsets - training, validation, and testing. The specific quantities applied to each dataset are detailed within their respective sections. Regardless of the dataset or selected class, convex hull points are retrieved from the training subset.

The same procedure is applied to all datasets studied: We begin by computing the PCA loading vectors against all training samples in the selected dataset. This results in a single PCA model, used to express samples by a varying amount of latent variables. As a single PCA model is used per dataset, the learned transformations are agnostic of the samples'

	#	type	in shape	out shape	units	size	stride	activation	param.
cnn-2l-16u	0	input	(1x32x32x3)	(1x32x32x3)	-	-	-	-	-
	1	conv2d	(1x32x32x3)	(1x32x32x16)	16	(3x3)	1	relu	448
	2	maxpool	(1x32x32x16)	(1x16x16x16)	-	(2x2)	2	-	-
	3	conv2d	(1x16x16x16)	(1x16x16x16)	16	(3x3)	1	relu	2320
	4	maxpool	(1x16x16x16)	(1x8x8x16)	-	(2x2)	2	-	-
	5	dense	(1x1024)	(1x10)	10	-	-	softmax	10250
	-	out	(1x10)	-	-	-	-	-	-
cnn-1l-16u	0	input	(1x32x32x3)	(1x32x32x3)	-	-	-	-	-
	1	conv2d	(1x32x32x3)	(1x32x32x16)	16	(3x3)	1	relu	448
	2	maxpool	(1x32x32x16)	(1x16x16x16)	-	(2x2)	2	-	-
	3	dense	(1x4096)	(1x10)	10	-	-	softmax	40970
	-	out	(1x10)	-	-	-	-	-	-

Table 4.1: Convolutional Neural Networks used in experiments; parameter configurations for CIFAR-10 shown.

class labels.

Once the PCA model is obtained, we apply ReDiHull to both the training and validation subsets. This produces two individual sets of statistics, enabling comparative validation of resulting observations. As discussed in chapter 2, the samples used to define convex hulls are determined from the training subset and maintained for training and validation trials. Maintaining the same convex hull between training and validation is necessary to produce reliable analyses, the convex hull serves as a non-parametric model of class extent. All results presented were obtained from the validation subsets.

The ability to compare results our technique to those deep learning algorithms were achieved using the architectures shown in tables 4.1 and 4.2. As noted, all convolutional layers comprised of a relu-activated, 2D spatial convolution, followed by a 2D maxpooling operation. Convolutions used 16 (3×3) kernels, 1×1 strides, and padding such that input dimensions are conserved. Maxpooling operations used 2×2 windows with a 1×1

	#	type	in shape	out shape	units	activation	param.
dnn-2l-8u	0	input	(1x32x32x3)	(1x3072)	-	-	-
	1	dense	(1x3072)	(1x8)	8	relu	24584
	2	dense	(1x8)	(1x8)	8	relu	72
	3	dense	(1x8)	(1x10)	10	softmax	90
	-	out	(1x10)	-	-	-	-
dnn-1l-8u	0	input	(1x32x32x3)	(1x3072)	-	-	-
	1	dense	(1x3072)	(1x8)	8	relu	24584
	2	dense	(1x8)	(1x10)	10	softmax	90
	-	out	(1x10)	-	-	-	-
dnn-bl	0	input	(1x32x32x3)	(1x3072)	-	-	-
	1	dense	(1x3072)	(1x10)	10	softmax	24584
	-	out	(1x10)	-	-	-	-

Table 4.2: Deep Neural Networks used in experiments; parameter configurations for CIFAR-10 shown.

stride. Each intermediate dense operations (table 4.2) contain 8 hidden units, to which relu activations are applied.

All datasets use identical architecture configurations, each trained from scratch using stochastic gradient descent, where training and optimization hyperparameters were determined for each dataset independently. To account for the likelihood of variation between identical networks, each architecture was trained 100 times, resulting in 500 models per dataset. From these, at most twenty networks are selected, using those that achieved the highest overall validation accuracy, so long as it met a certain threshold (network accuracy $\geq$ one standard deviation above the mean accuracy between all models).

With the exception of *sand* terrain classes, both the Martian terrain and CSA MET datasets are afflicted by sample sets of significantly limited size. We have found these properties - class imbalances and limited number of samples - to introduce a high-degree of uncertainty for networks trained using these datasets. As such, we use CIFAR-10 [37] to compare the results of the KL divergence described in 3.3 to performance results of networks trained and evaluated using the same dataset. With that being said, although both absolute and relative prediction errors of were found too unstable to use for meaningful comparison, the ordering of difficult examples remained mostly consistent, as can be seen by comparing various network results in appendices A.1 and B.1.

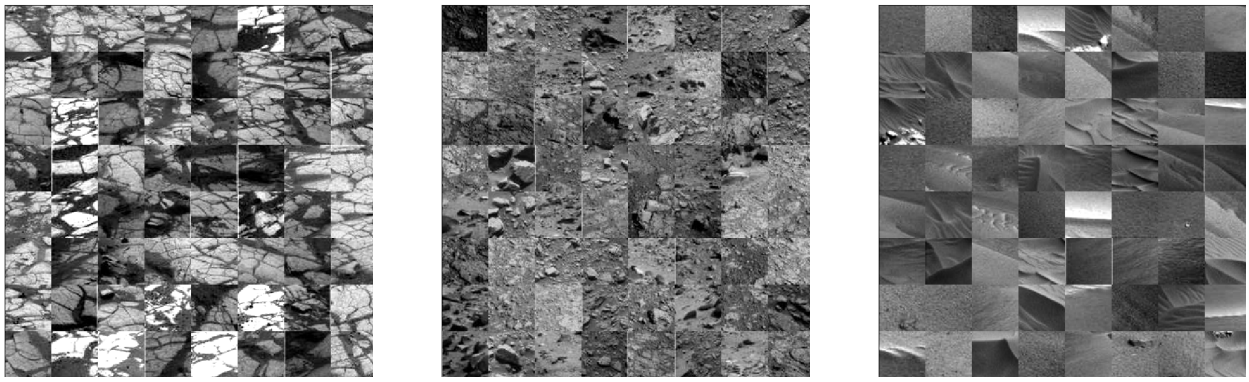


Figure 4.1: Examples of terrain image samples within the Martian terrain dataset [1]. From left to right bedrock, rock-strewn, and sand.

4.1 Martian Terrain

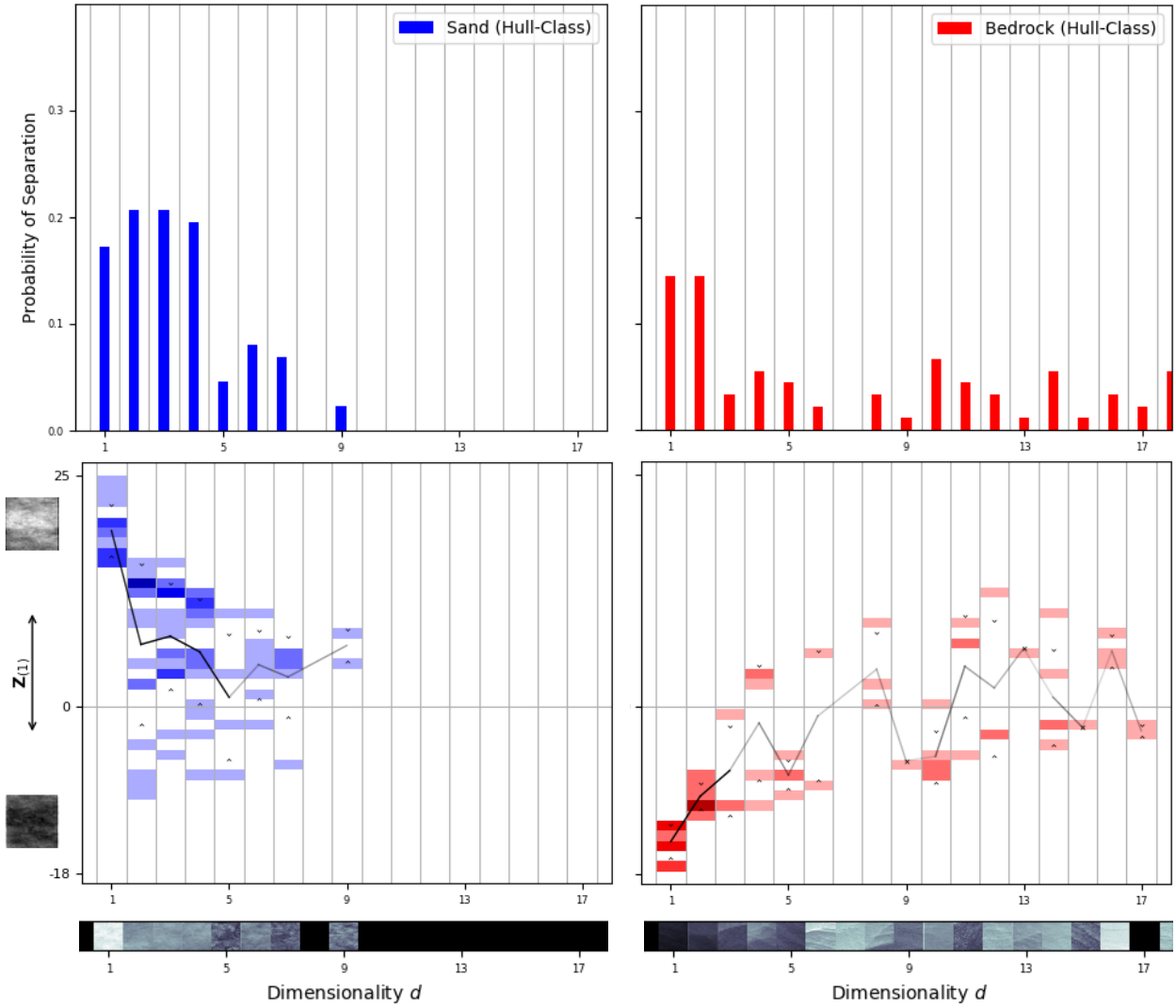
The Martian terrain dataset, labelled by Shukla and Skonieczny in [1], serves as an example of a real-world scenario. Samples depict bedrock, sand, or rock-strewn terrains, selected from a subset of images captured by the Opportunity Mars Exploration Rover during its deployment. Due to the terrains' lack of structured-features, high visual similarity, and multi-scale variations, the dataset contains many unique classification challenges not typically seen within object image datasets.

4.1.1 Observation(s) using ReDiHull

Figure 4.2 compares the statistics extracted when attempting to separate *Bedrock* from *Sand*'s convex hull, to the opposite scenario of trying to separate *Sand* from *Bedrock*'s convex hull. In either corresponding separability probability distribution, there are no notable spikes of potential interest. Additionally, while *Bedrock* displays a clear, single modal probability distribution, no general behavior is apparent for the case of separating *Sand* from *Bedrock*. Moving on to the distributions of projected values onto the first loading vector, it is evident that *Bedrock* images requiring more dimensions to separate from sand have a projected value near zero, whereas those requiring few dimensions have a large, positive projected value. Conversely, *Sand* images which separate in few dimensions from *Bedrock*'s convex hull have a large negative projection onto this loading vector. As *Sand* images require more

dimensions to separate, there is a clear trend that negative-valued projections tend towards zero.

By applying these explanations to the corresponding sets of difficulty spectra, our interpretations have a clear qualitative meaning - *Bedrock* images which are more difficult to distinguish from *Sand* are in general darker, which corresponds to difficulty arising during circumstances of low light or low exposure. In contrast, *Sand* images which are more difficult to distinguish from *Bedrock* are lighter overall, alluding to classification difficulty when *Sand* images are captured with high exposure. Extending upon this, we note that similar trends are observable in figures 4.3 4.4 which demonstrate *sand* and *bedrock* samples causing difficulty to neural networks.



(a) **(b)**

Figure 4.2: Select results obtained for the Martian terrain dataset introduced by [1]. From the first to last row - separation probability distributions, projected value histograms, and difficulty spectra for (a) *Bedrock* images separated from *Sand* and (b) *Sand* images separated from *Bedrock*

4.1.2 Comparing ReDiHull Sample Difficulty to Neural Network Sample Difficulty

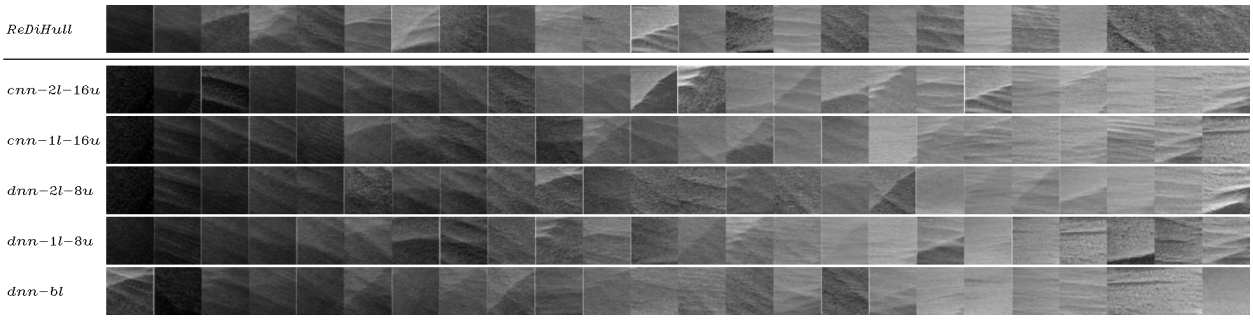


Figure 4.3: Increasing difficulty of *sand* terrain samples (true label) w.r.t *bedrock* terrain samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures

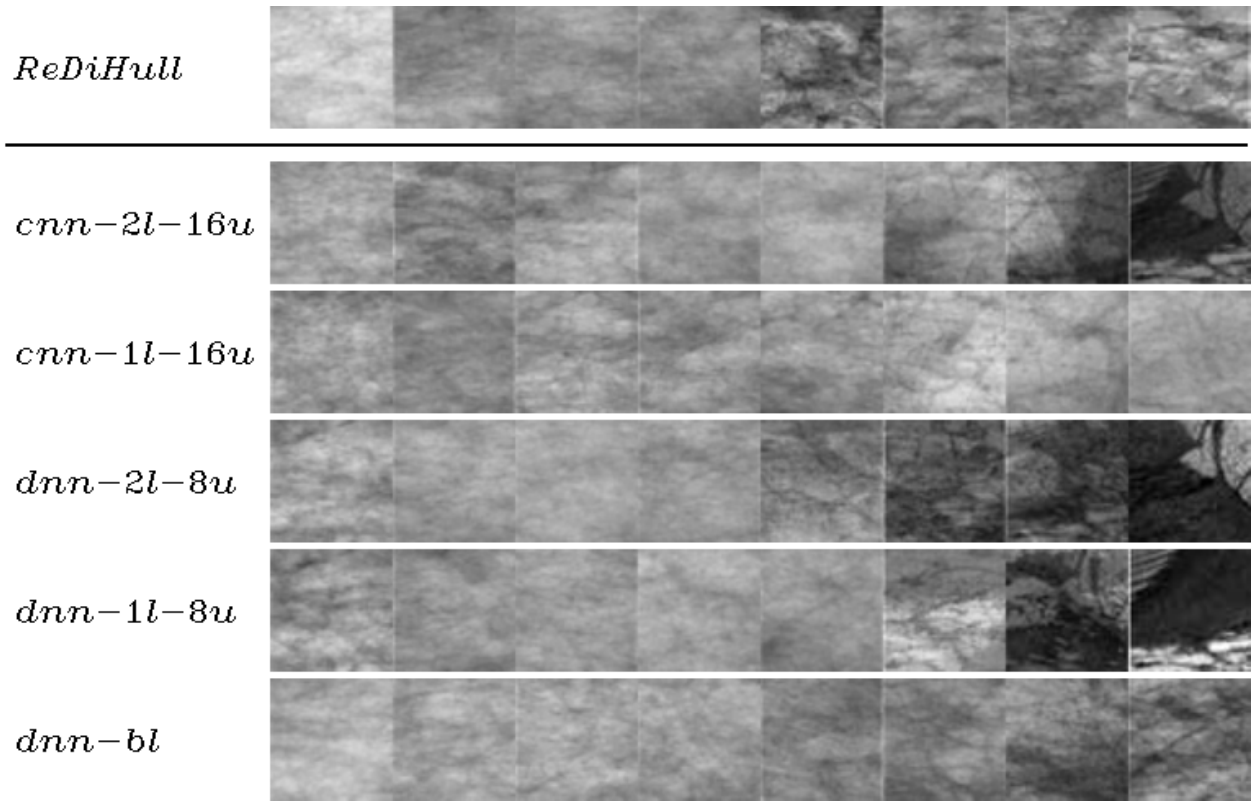


Figure 4.4: Increasing difficulty of *bedrock* terrain samples (true label) w.r.t *sand* terrain samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

4.2 Canadian Space Agency Mars Emulation Terrain

Developed by Mission Control Space Services, the CSA MET dataset [35] is composed of terrain images captured at the Canadian Space Agency Mars Emulation Terrain site in Quebec, Canada. This dataset shares a similar purpose to the Martian terrain set - for use in training and evaluating an autonomous soil assessment system used for autonomous path planning and risk assessment. The datasets differ in that the CSA MET samples are 3-channel (RGB) color images as opposed to the Martian terrain datasets' grayscale NavCam images. Further, the task in the CSA MET dataset is semantic segmentation rather than image classification.

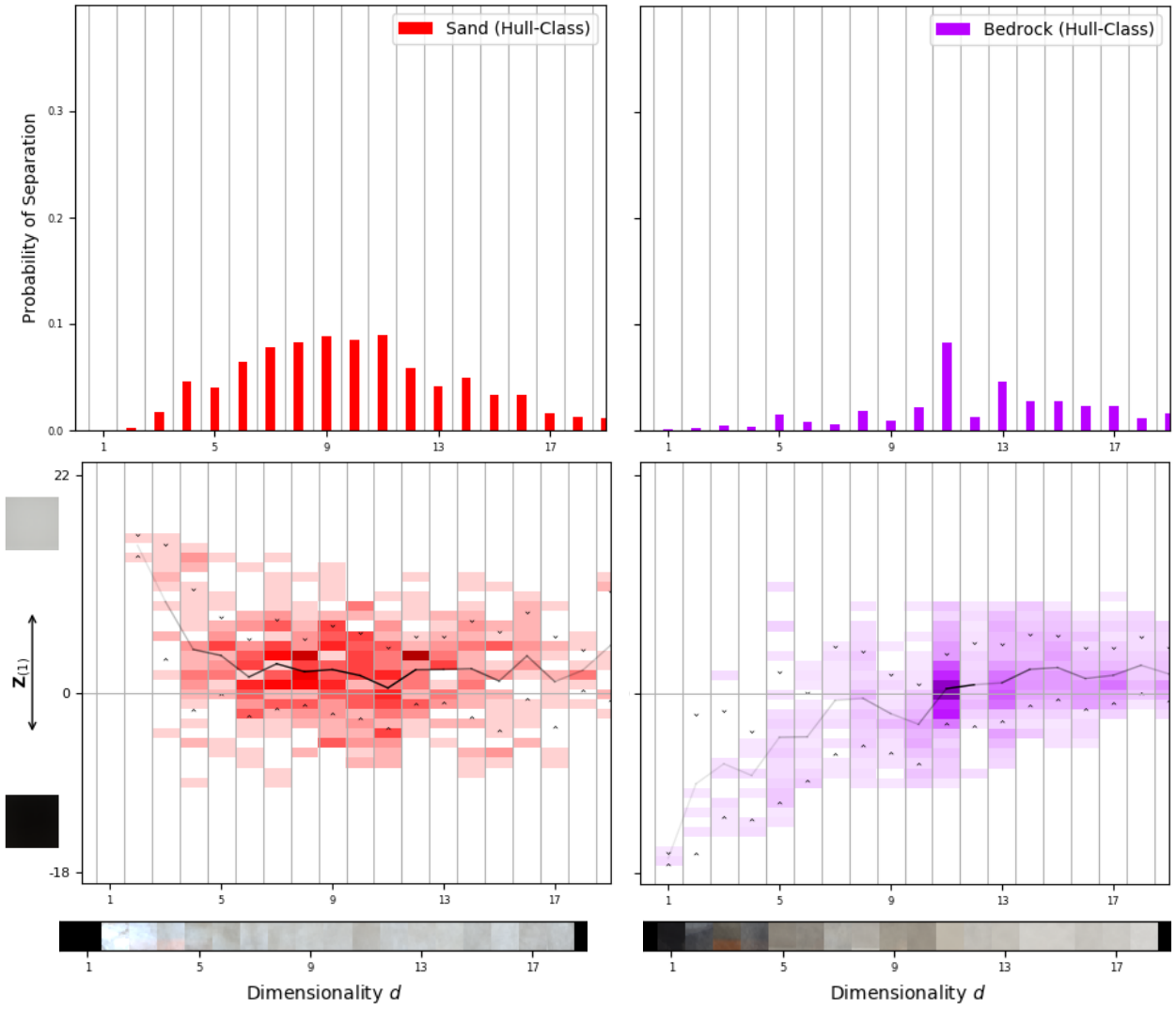
The goal of segmentation is to partition images such that each pixel is assigned a class label. To analyze segmentation datasets using our technique, we partition the the annotated images into smaller, square patches, providing sample images of unmixed terrain classes.

The images were captured at a resolution of 1080p - i.e. a width of 1920 pixels, height of 1080 pixels. They were then resized, without aspect-ratio preservation, to a size of 512x512 pixels. Annotations were produced for the resized images.

At the time of writing, the autonomous soil assessment system uses a neural network to segment terrain images. More specifically, we use the MobileNetV2 architecture [38]. To determine our image patch size, we solve for the value of k_s as $k_s = \frac{w_{in}}{w_{out}}$. Using the values specified by the authors, we set $w_{in} = 224$ as the input size of the network and $w_{out} = 7$ as the output size following the tenth network operation. This produces a value of 32×32 for the patch width and height. This procedure approximates the first ten operations of MobileNetV2 as a single convolution operation. We exclude from analysis the final, eleventh operation that produces the output predictions.

4.2.1 Observation(s) using ReDiHull

The first observation to discuss is the presence of a large spike in separability near the eleventh dimension when either *black sand*, *sand*, or *gravel* is applied as a reference-class and *bedrock* or *gravel* is applied as a hull-class. Figure 4.5b (top-right) shows the separation probability distribution for separating sand from bedrock.



(a) **(b)**

Figure 4.5: Select results obtained for the CSA MET dataset. From the first to last row - probability of separation distributions, projected values histograms, and difficulty spectra for (a) *Bedrock* images separated from *Sand* (b) *Sand* images separated from *Bedrock*

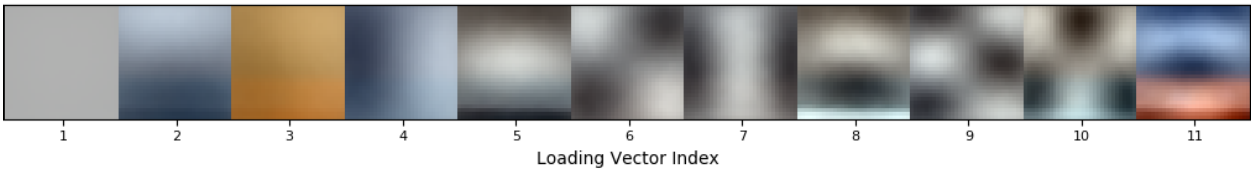


Figure 4.6: The top-11 PCA loading vectors computed for the CSA MET dataset.

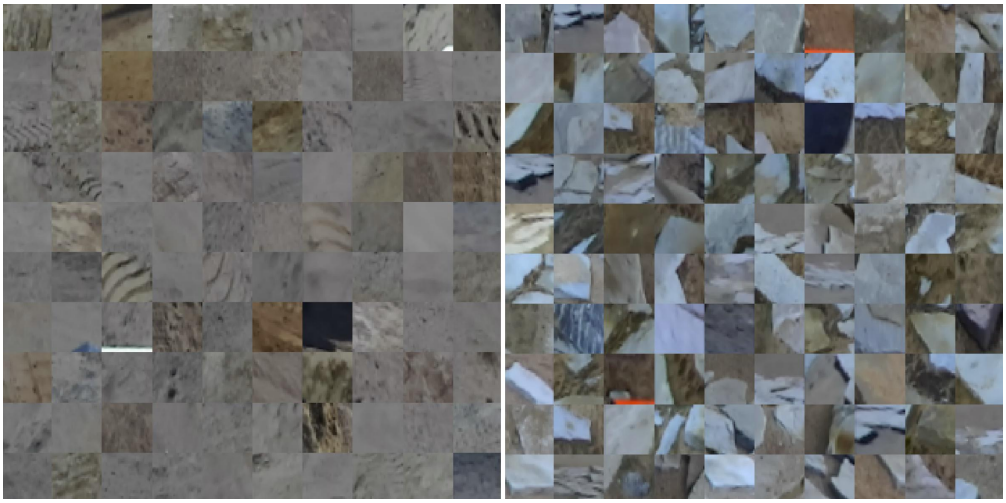


Figure 4.7: 100 samples each of color sand (left) and bedrock (right) images from the CSA MET dataset. Bedrock images include sub-regions of bluish stone and non-blue soil.

Recalling the discussion from section 3, we had noted that, in most situations, a spike in separability corresponds to the projection onto a distinguishing or separating feature.

Figure 4.6 shows the first 11 PCA loading vectors computed for this dataset. Note that the 11th loading vector is the first to encode a color gradient, where part of the filter is blue and the other part is non-blue.

Figure 4.7 shows 100 samples each of sand and bedrock images from the CSA MET dataset. It is clear that the bedrock class contains many images with distinct sub-regions (of bluish stone and non-blue soil), aligning with the main information that component 11 encodes. Sand images do not contain such color-distinct sub-regions.

To explicitly quantify the gradient in blueness between stone and soil sub-regions of bedrock images, as opposed to simply a gradient in brightness that could have been captured by loading vectors 2 and 4 for example, grayscale images of the bedrock are compared

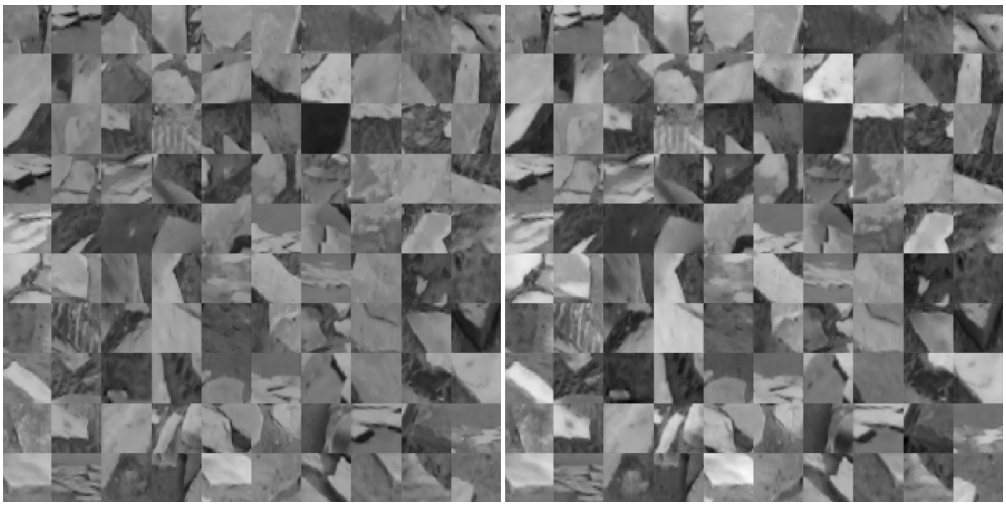


Figure 4.8: Grayscale (left) and blue channel (right) of CSA MET bedrock images. Blue channel shows higher contrast.

to the bedrock images' blue channel in figure 4.8. The blue channel data clearly shows higher contrast than the grayscale data; this higher contrast is further quantified by a higher standard deviation in pixel values for the blue channel ($\sigma = 45$) vs. grayscale ($\sigma = 35$).

On the other hand, there is very little difference between grayscale and blue channel for sand (see figure 4.9; $\sigma = 17$ vs. $\sigma = 19$).

Now looking at the projected value histogram for the 11th loading vector in figure 4.10, the large spike in separations of sand from bedrock at component 11 has a projected value centered very near 0. Combining all the information discussed above suggests that sand images become distinguishable from bedrock, in the color CSA MET dataset, when it becomes clear the sand images *do not* include much of a color gradient.

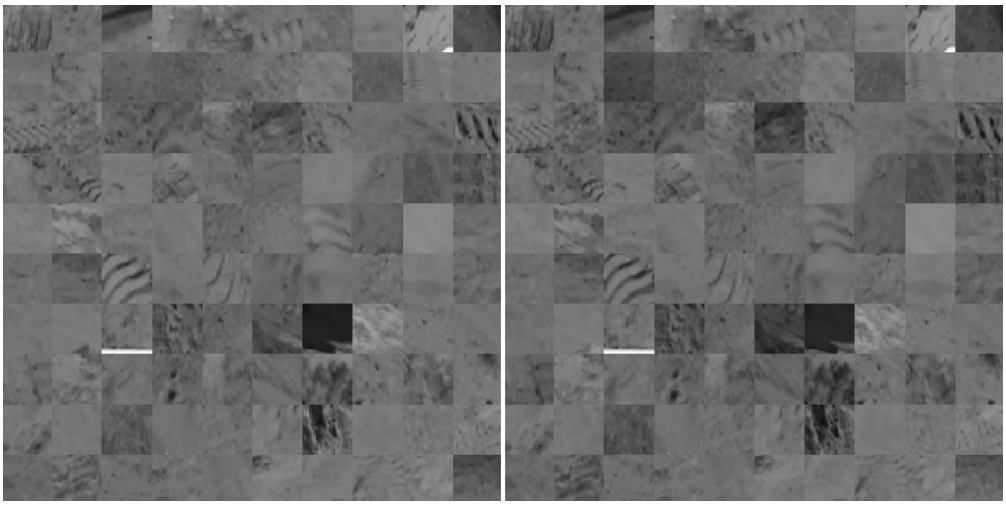


Figure 4.9: Grayscale (left) and blue channel (right) of CSA MET sand images, with little discernible difference in contrast.

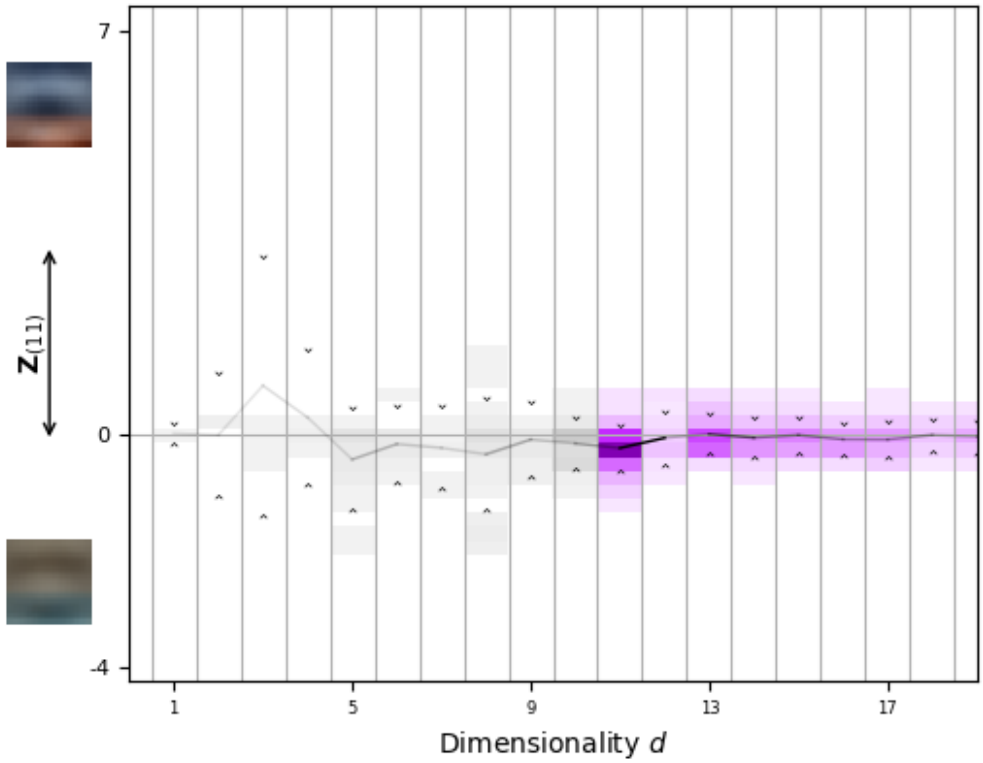


Figure 4.10: Resulting PCA projected value histograms for the 11th component using the CSA MET terrains *Sand* as the reference-class and *Bedrock* as the hull-class.

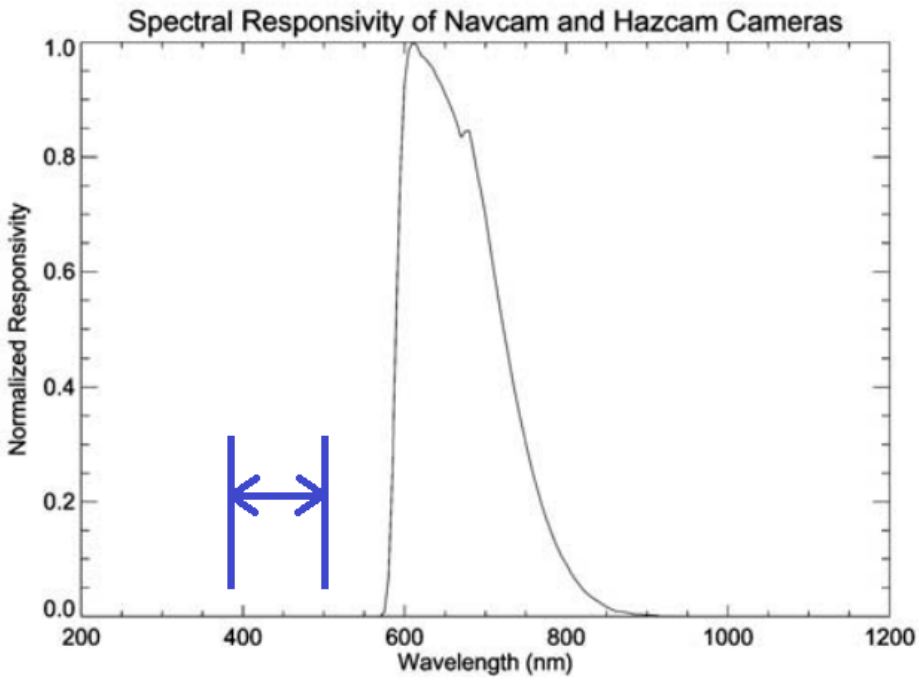


Figure 4.11: The spectral response of the NavCam cameras [4], annotated to show the range of blue-light wavelengths. As the band-pass filter rejects this range of signals, classifiers applied to NavCam images cannot leverage the separating feature discovered using the CSA MET dataset.

4.2.2 Curiosity MastCam Images

To investigate the potential of the separating feature noted for the CSA MET dataset, we produce a similar color analysis using Martian terrain images captured by the NASA Mars Science Laboratory (MSL) "Curiosity" rover. As images within the CSA MET dataset depicted emulated terrain, it was uncertain whether the discovered feature could be transferred and/or leveraged by systems designed to classify Martian terrain.

Within the same context, much of the literature consulted focused on Martian terrain classification enabled by Navigational camera (NavCam) images [1][6][7]. To the best of our knowledge, all NavCam modules deployed, both current and previous, have applied a red band-pass filter, abating signals within the blue light range [4]; the spectral response of the NavCam camera is included in figure 4.11, annotated to indicate the range of wavelengths corresponding to blue-light.

(a) **(b)**

Figure 4.12: (a) *sand* and (b) *bedrock* terrain patches generated for the MastCam image analysis.

Following the pre and post-processing procedures outlined by Shukla and Skonieczny [1]: 44 source scenes were selected, 23 sand-depicting scenes and 21 bedrock-depicting scenes. The PDS Imagine Node generally provides multiple products for a given data record corresponding to different types and/or levels of processing. To ensure coherency of the data, only "DRCX" type data records were selected, i.e. those which have been decompressed, radiometrically corrected, and color corrected [39].

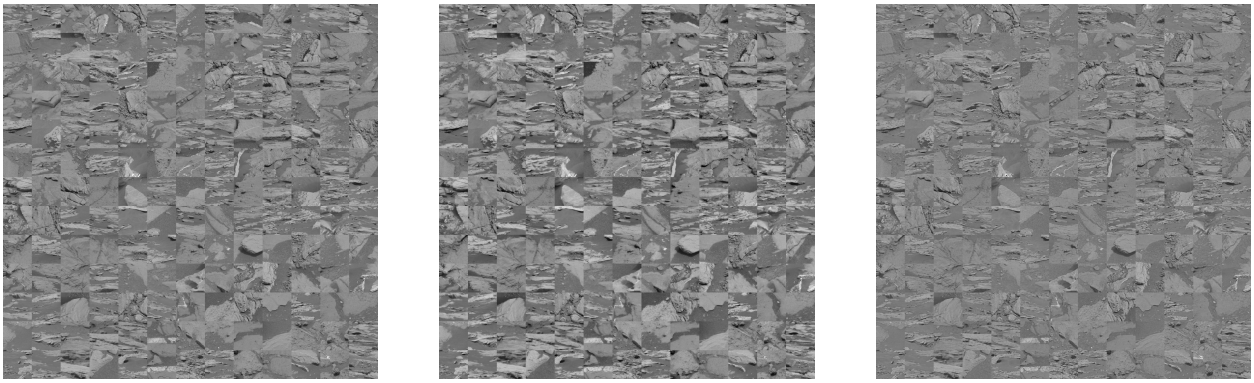
Next, we extract non-overlapping, 128×128 sized patch samples from the source scenes. Sand and bedrock terrain depicted within the resulting images should be easily identifiable as well as mostly homogeneous - that is absent of visible rocks, rover tracks, or other circumstances introducing label uncertainty. In response, we manually validate the resulting image (i.e. patch) samples, discarding those which do not fit the criteria for suitable terrain image content, as described in the works done by Rothrock [6] and Shukla [1]. For visual reference, a subset of both terrains' resulting image samples can be found within figure 4.12.

To assess the potential existence of a similar color-related separating discovered within CSA MET, we apply the same statistical color analysis used. As presented in table 4.3, we find a similar statistical relation to occur for the Curiosity MastCam *sand* and *bedrock* images, as well as when comparing images' red and blue channel data. The corresponding

	CSA MET		MastCam	
	Sand	*Bedrock*	*Sand*	*Bedrock*
Blue	19	45	24	32
Gray	17	35	24	25
Red			26	22

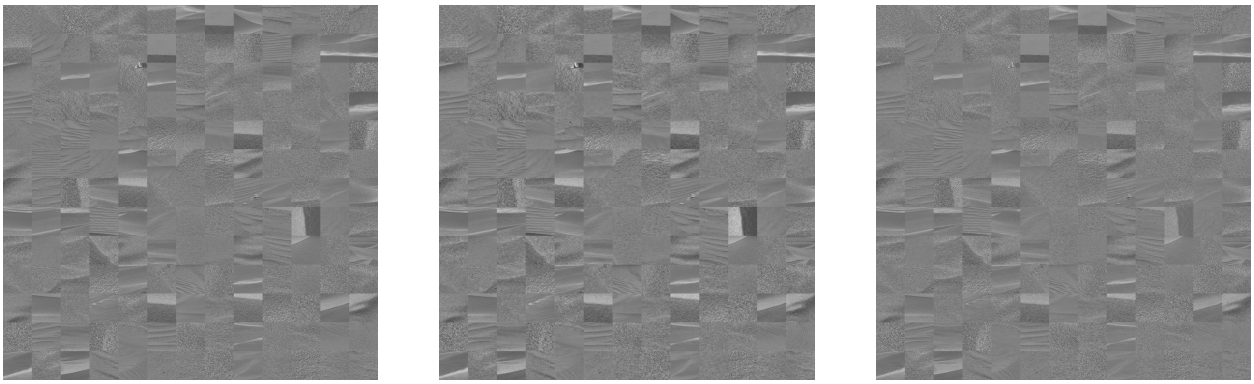
Table 4.3: Standard deviations of pixel values for terrains indicated in each columns. We note that for both sets of images: *sand* demonstrates minor differences in standard deviation between grayscale and blue-channel variants whereas *bedrock* demonstrates considerable differences between grayscale and blue-channel standard deviations. Additionally for Curiosity MastCam images, the same behaviour is noted when comparing red and blue channel statistics.

qualitative differences can be noted within figures 4.13 and 4.14.



(a) **(b)** **(c)**

Figure 4.13: (a) Grayscale (b) blue channel (c) red channel data for bedrock images captured by the Curiosity rover MastCam. Blue channel shows higher contrast than gray, whereas red channel data shows reduced contrast.



(a) **(b)** **(c)**

Figure 4.14: (a) Grayscale (b) blue channel (c) red channel data for sand images captured by the Curiosity rover MastCam, noting little discernible difference in contrast between the three.

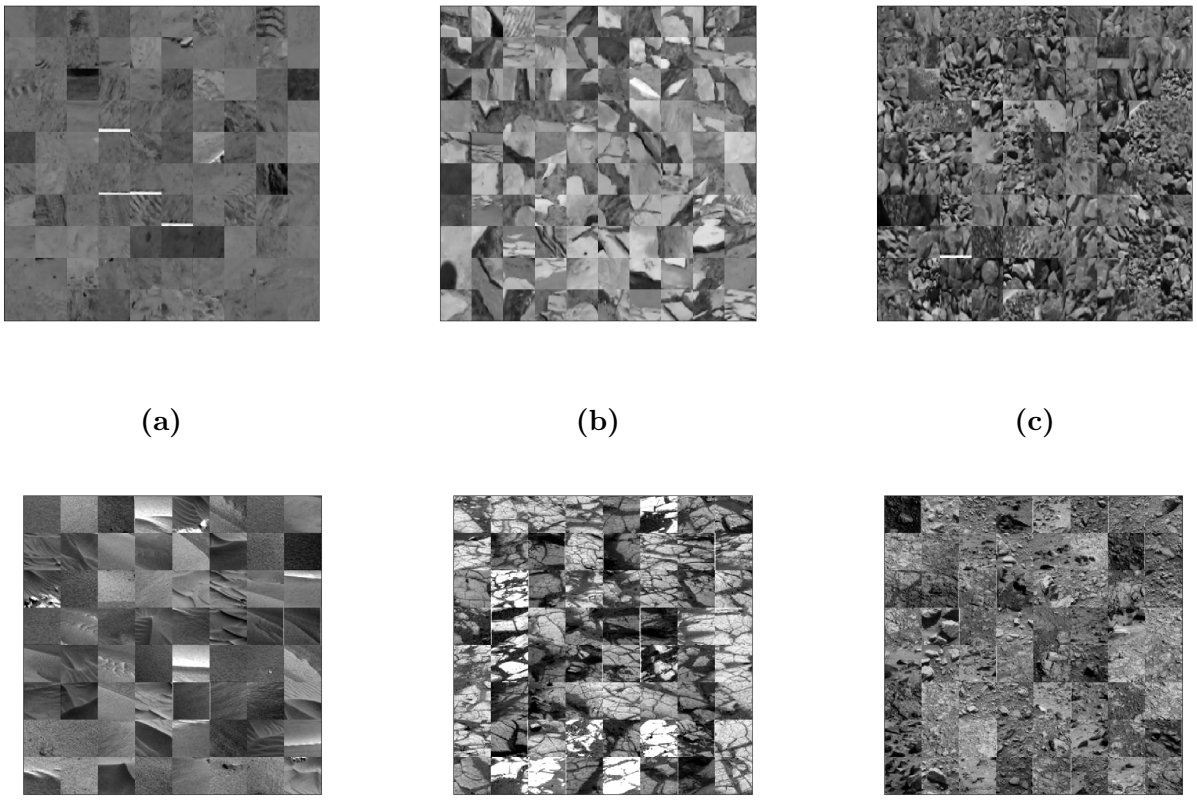
4.2.3 Shared Characteristics: Martian terrain and CSA MET

Our results from analyzing the CSA MET dataset revealed similarities between the CSA MET and Martian terrain datasets. Through our analysis we found we can identify terrestrial terrain images that resemble Martian terrain images, and that could act as surrogate terrain data for training neural networks. Martian data is limited, so assembling large, visual analog training sets from terrestrial data enables the use of deep learning for planetary terrain classification.

The CSA MET dataset contained terrain classes that were not represented in the Martian terrain dataset. In order to compare samples, we established class-correspondences. The Martian terrain dataset has three terrain types - *Bedrock*, *Rocks*, and *Sand*. The CSA MET dataset has seven terrain types - *Bedrock*, *Black Sand*, *Gravel*, *Outcrop*, as well as *Sky*, and *Vegetation* classes, which we excluded from our experiments. Because the Martian terrain dataset does not contain samples resembling CSA MET's *Black Sand* or *Outcrop* terrains they were also excluded from comparison. The remaining CSA MET classes of *Gravel*, *Bedrock*, and *Sand*, are qualitatively similar to the Martian terrain terrain classes of *Rocks*, *Bedrock*, and *Sand*, respectively. Figure 4.15 is provided to illustrate the samples' visual similarities.

The PCA projection histograms in figure 4.16, noting the similarity between the first-computed loading vector of each dataset - for either dataset, the direction of maximal variance appears to coincide with overall image exposure. Now, examining each row individually, it seems that these loading vectors also share in their relation to general trends of inter-class difficulty. Within the first row, corresponding to *Sand* as a reference-class and *Bedrock* as a hull-class, we can see that *Sand* images which require fewer dimensions to separate have large, negative projections. As the number of dimensions to separate increases, images generally tend towards more positive projected values. In the next row, the reference-class is maintained while the hull-class is set to *Gravel* for CSA MET and *Rocks* for Martian terrain. Under this scenario, the absolute projected values diminish as the number of dimensions to separate increases, creating a cone-like distribution, observable within both columns.

We continue to the third row of figure 4.16, which compares the datasets' results for use



(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 4.15: Top row presents 100 patch samples from the CSA MET terrain classes (a) *Sand*, (b) *Bedrock*, and (c) *Gravel*. *Sand* terrain class converted to grayscale. Bottom row presents 64 samples from the Martian terrain image classes (d) *Sand*, (e) *Bedrock*, and (f) *Rocks*.

of *Gravel* or *Rock* as a reference-class and *Bedrock* as a hull-class. For either dataset, this formulation demonstrates a less pronounced version of the trend discussed for first row of figure 4.16. On the fourth row, *Gravel* and *Rock* are maintained as reference-classes while the hull-class is set to *Sand*. Under these conditions, it appears that the datasets share an absence of any discernible relation to the number of dimensions required for separation in regards to the first PCA loading vector.

The fifth row of figure 4.16 compares the projected value histograms when *Bedrock* is used as the reference-class and *Gravel* or *Rocks* as the hull-class. In either column, a negative trend is identifiable, indicating that *Bedrock* samples tend towards more negative projections with increasing number of dimensions to separate. This trend is similar to that discussed within the Martian terrain results section, wherein *Bedrock* was used as the reference-class and *Sand* as the hull-class. Now moving to the final row of figure 4.16, we are incapable of identifying any parallels with certainty; apart from differences within the terrain images themselves, this may possibly be explained by the relatively low number of *Bedrock* images within the Martian terrain dataset. In other words, while the Martian terrain results seem to indicate a negative trend, the observation may subside if more *Bedrock* were available.

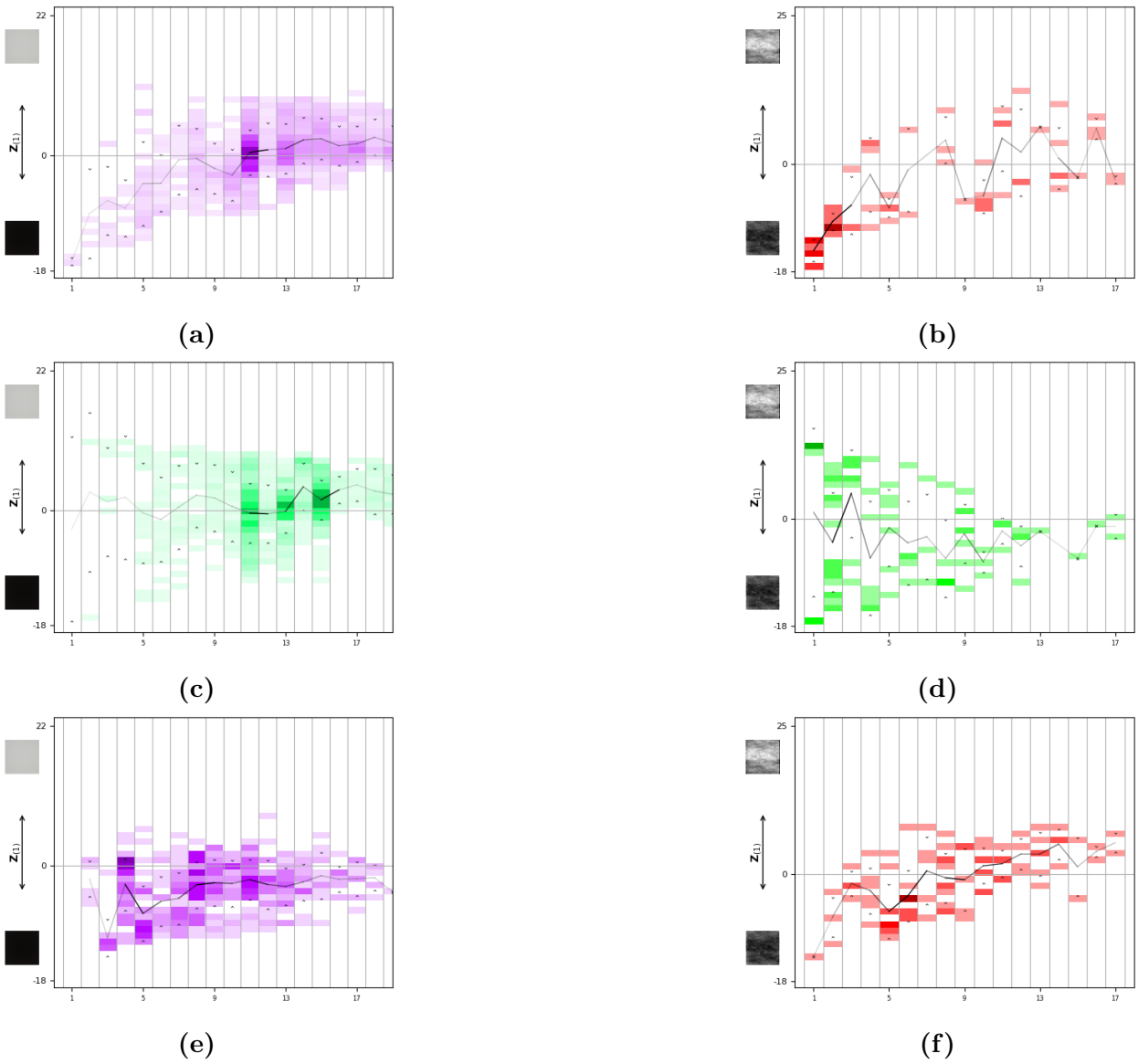


Figure 4.16: For the left-hand column, starting from the top row, we are shown CSA MET PCA projected value histograms for the first component using (a) hull-class as *Bedrock* and reference-class as *Sand*, (c) hull-class as *Bedrock* and reference-class as *Gravel*, (e) hull-class as *Gravel* and reference-class as *Bedrock*. For the right-hand column, starting from the top row, we are shown Martian terrain PCA projected value histograms for the first component using (b) hull-class as *Bedrock* and reference-class as *Sand*, (d) hull-class as *Bedrock* and reference-class as *Rocks*, (f) hull-class as *Rocks* and reference-class as *Bedrock*.

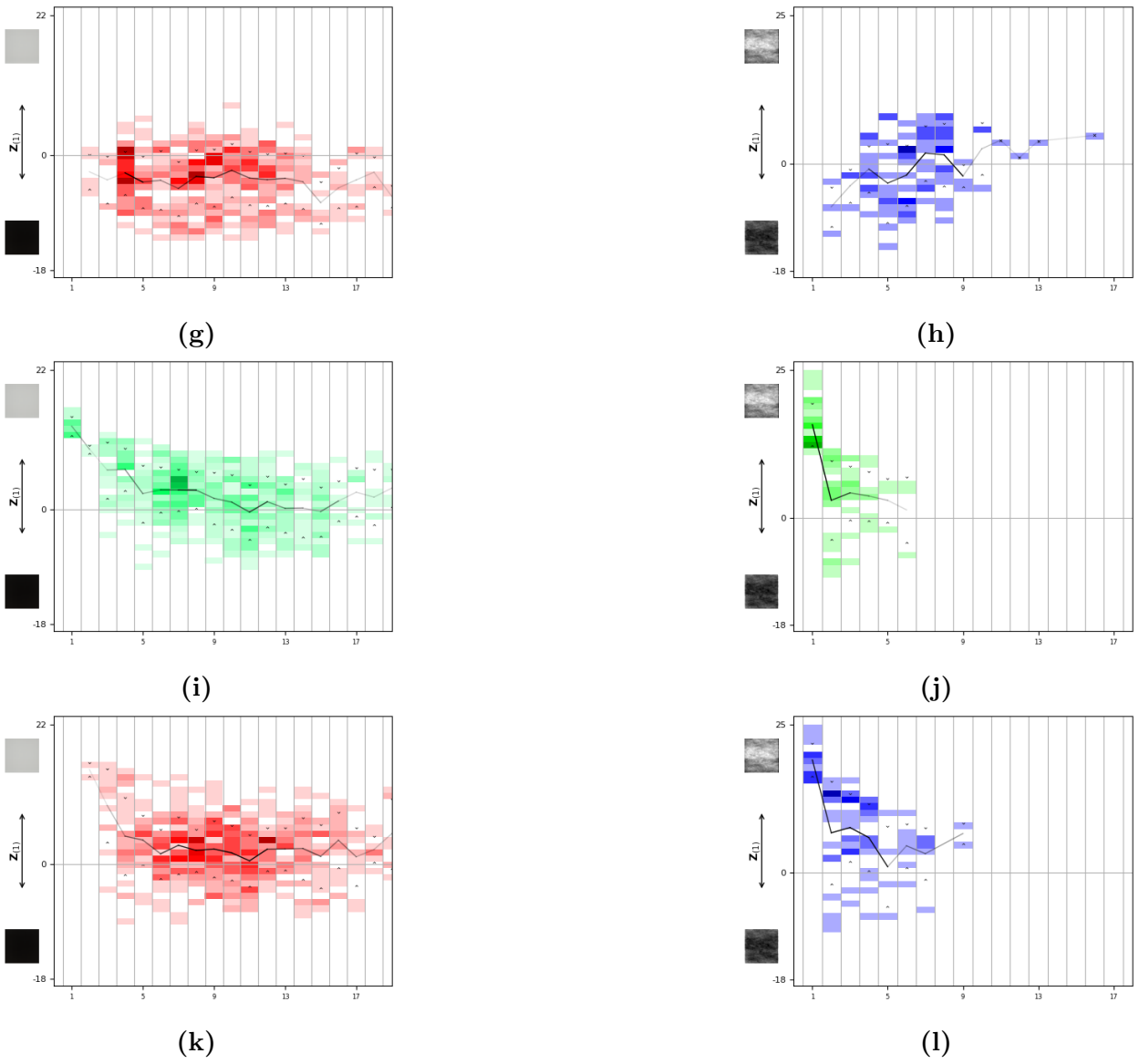


Figure 4.16: (Cont.) For the left-hand column, starting from the top row, we are shown CSA MET PCA projected value histograms for the first component using (g) hull-class as *Gravel* and reference-class as *Sand*, (i) hull-class as *Sand* and reference-class as *Gravel*, and (k) hull-class as *Sand* and reference-class as *Bedrock*. For the right-hand column, starting from the top row, we are shown Martian terrain PCA projected value histograms for the first component using (h) hull-class as *Rocks* and reference-class as *Sand*, (j) hull-class as *Sand* and reference-class as *Rocks*, and (l) hull-class as *Sand* and reference-class as *Bedrock*.

4.3 CIFAR-10

We apply our technique to CIFAR-10 - an image classification dataset depicting 10 classes of real-world objects and animals [37]. For results shown, the dataset was split into 50 000 training images and 10 000. From this evaluation, we choose to highlight observations made in regards to the Deer class, then validate their relevance to classification through qualitative comparison of first layer features computed by various neural networks discriminating Deer from other classes.

4.3.1 Observation(s) using ReDiHull

We begin our discussion by examining the probability distribution shown in figure 4.17. Doing so reveals a large spike in separability at seventeen dimensions, which, as mentioned in section 3.1.1, constitutes an anomalous behavior, with regard to the dominant trend. To determine the cause of the observed behavior, we refer to the corresponding projected value distribution (figure 4.17). Through analysis of the projected data, it is clear that the characteristic related to this behavior is that images belonging to the Deer class have a stronger, positive correlation with the seventeenth loading vector, relative to images belonging to the Horse class. To the left of the projected value distribution, we are shown that the seventeenth loading vector encodes green (positive) to magenta (negative) overall image content. Summarizing these observations - images containing a significant amount of green content are likely to depict a Deer. Moreover, the result shown in figure 4.18 allows us to confirm this observation's relevance, demonstrating the convolutional neural networks' affinity towards learning parameters that capture green color-information, and the deep fully-connected networks' association of the color green to the Deer class.

Noting that the separating feature is horizontally symmetric, there is potential relevance to applications seeking to build more efficient networks via weight sharing and/or directing networks to leverage compressible features. For example, this observation could be applied to enforce a simple feature detector, leveraging the separability observed for green-colored content. As mostly-green images usually correspond to Deer, a high-confidence prediction could be quickly obtained for preliminary decision making, while simultaneously testing for

additional features to refine the prediction. Moreover, the feature itself is mostly uniform thus it is likely that it may be compressed significantly. If we assume some prior knowledge of the problem, the observation for separating Deer from other classes may lead one to formulate a converse application for this characteristic. Green colored background does not form a causal relation with Deer, thus preventing a classifier from associating this feature with Deer images may result in a more robust system. The work by Dieleman, De Fauw, and Kavukcuoglu [40] and that by Cohen and Welling [41] have produced far more advanced interpretations and methods of applying feature symmetry. Further, motivating networks to learn weights expressing fewer unique values - similar to the non-uniqueness observed via the separating feature's color uniformity - has seen success by Raghavan et al. in achieving high performing, compressible networks [42].

4.3.2 Comparing Difficulty Measure with Network Performance

To obtain performance values used to compare with KL divergence, we trained and evaluated the architectures detailed in table 4.1 and 4.2. For comparative reference, the classification rates for the "cnn-2l-16u" model are presented in table 4.5 and the corresponding KL divergence values are shown in table 4.4.

It should be noted when interpreting KL divergence tables, larger values indicate better performance. From (3.3), we may deduce that D_{KL} will be large for values $Q << P$. Translating this to our method - larger values of D_{KL} are computed for separating features which (i) induce a high degree of separability between class ω and class ψ, while simultaneously (ii) having little to no effect on ω's convex hull.

Note that all 4 class pairings that are confused the most by the classifier (confused at least 10% of the time) in table 4.5 have very low (i.e. poor) KL divergence, 0.06-0.11 in table 4.4. For less extreme examples of classification difficulty, KL divergence has some further correlation with the confusion matrix, though with some false positives and false negatives.

Hull Class

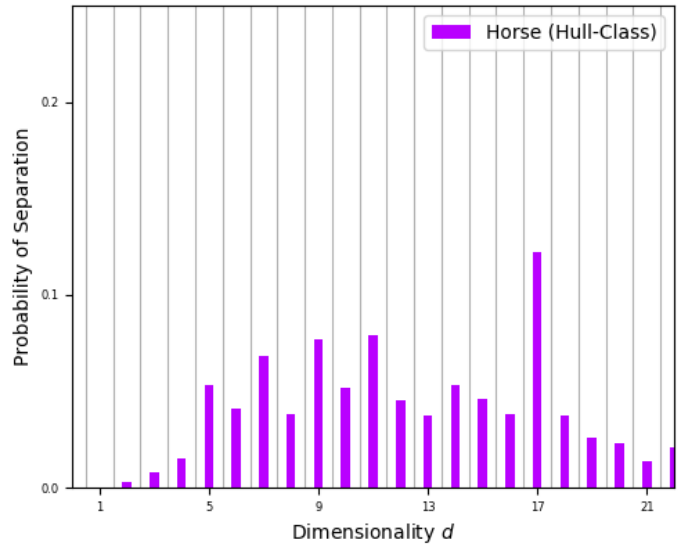
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane		0.27	0.36	0.30	0.53	0.31	1.24	0.37	0.13	0.31
Automobile	0.87		1.34	0.46	1.39	0.47	1.45	0.45	0.49	0.10
Bird	0.11	0.22		0.23	0.12	0.22	0.13	0.16	0.18	0.21
Cat	0.51	0.18	0.47		0.63	0.07	0.55	0.09	0.52	0.22
Deer	0.11	0.26	0.10	0.34		0.23	0.06	0.23	0.25	0.22
Dog	0.65	0.25	0.51	0.08	0.81		0.69	0.11	0.56	0.28
Frog	0.18	0.25	0.20	0.24	0.19	0.20		0.18	0.31	0.23
Horse	0.47	0.08	0.48	0.15	0.51	0.11	0.70		0.62	0.08
Ship	0.20	0.20	0.59	0.32	0.68	0.50	1.61	0.49		0.15
Truck	0.81	0.06	1.27	0.34	1.21	0.52	1.55	0.30	0.51	

Table 4.4: CIFAR-10 difficulty quantified using KL divergence.

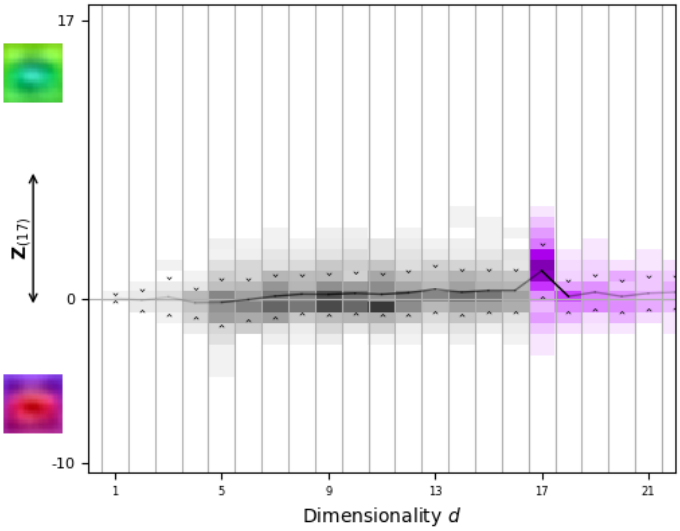
Predicted Class

	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane		2.2	5.2	2.2	2.5	1.1	0.9	1.4	7.7	3.3
Automobile	2.4		0.8	1.0	0.5	0.6	0.9	0.7	2.9	8.8
Bird	5.3	0.8		5.9	7.9	6.6	5.5	4.1	1.5	1.0
Cat	2.2	1.2	6.1		5.6	14.0	6.3	3.7	1.7	1.7
Deer	2.2	0.5	7.9	5.7		4.7	6.0	6.5	1.1	0.6
Dog	1.3	0.5	5.5	12.2	4.0		2.7	5.2	1.0	0.9
Frog	0.8	0.8	4.5	5.9	5.0	3.3		1.1	0.8	0.7
Horse	1.7	0.7	3.7	3.7	5.5	5.7	1.0		0.5	1.6
Ship	7.1	3.8	1.6	1.6	1.0	0.9	0.8	0.7		3.1
Truck	3.1	7.9	0.9	1.6	0.8	1.1	0.7	2.0	2.8	

Table 4.5: Average sample prediction error [%] for "cnn-2l-16u" trained and evaluated on the CIFAR-10 dataset

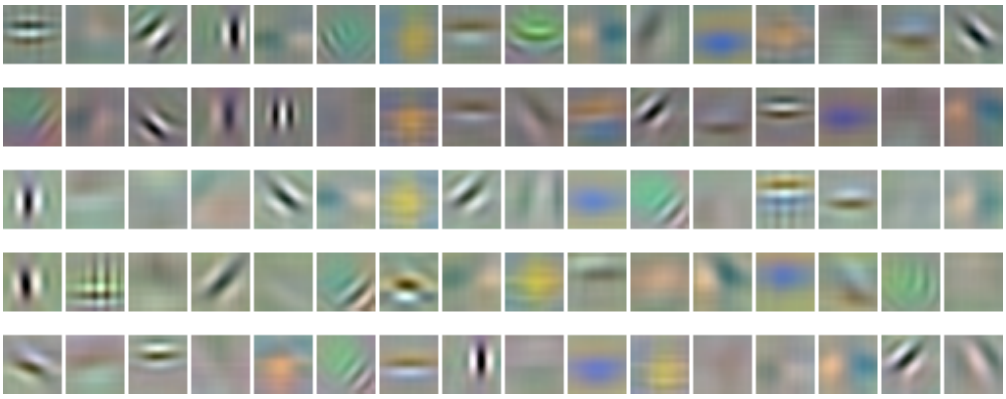


(a)



(b)

Figure 4.17: Resulting visualizations from applying our algorithm and analytical procedure to CIFAR-10. (a) shows the probability of separation for a given dimensional-space when separating Deer from the Horse class convex hull. (b) shows the histograms of projected values, allowing for interpretation of the behavior observed in the probability distribution.



(a)

(b)

Figure 4.18: (a) First layer weights learned by several convolutional neural networks, (b) the first layer weight most activated for Deer images, learned by a deep fully-connected network.

4.4 MNIST

Produced by [36], the MNIST dataset contains monochrome images of handwritten digits. For the context of this work, MNIST assumes the role of a less challenging image classification task, with even simple models capable of achieving below 10% error. To produce the results discussed herein, we designated 50 000 images to a training set, 10 000 to a validation set, and the remaining 10 000 to a test set.

The first set of results, shown in figure 4.19 demonstrate that relational difficulty is not necessarily a commutative property. In the left column digits appear more *Eight*-like with increasing dimensions required to separate them. The right column shows the result of attempting to separate digit *Eight* images from the hull digit of the respective row. Comparing between columns reveals that variational modes which lead a digit to appear more *Eight*-like are not the same as those which lead a *Eight* to appear more like the utilized hull digit. For example, in the row comparing ($\psi : Eight$, $\omega : Four$) to ($\psi : Four$, $\omega : Eight$) the within-class variation leading to increased difficulty in distinguishing *Four* from *Eight* are not the same as the those leading to increased difficulty in distinguishing *Eight* from *Four*. Additionally, separating digit *Eight* images from the hull of digit *Four* requires more dimensions than the reversed scenario. This is behavior is also noted for classification.

The validation set performance of two convolutional neural networks are shown in tables 4.6 and 4.7 respectively. Referring to these results, it can be seen that when either model is tasked with classifying images of digit *Four*, half of the erroneous classifications recorded are due to prediction of digit *Nine*. The reversed scenario, in which the classifiers attempt to predict images of digit *Nine*, only see a quarter of its error correspond to digit *Four*.

Another observation warranting discussion is phenomena wherein visually distinct subgroups invoke different levels of difficulty. For this, we refer to figure 4.20. Similar to the previously discussed results, examples which require more dimensions to achieve separability appear increasingly similar to the respective convex hull-defining digit. Applying a more thorough examination reveals that the similarity can be attributed to distinct variations of digit *One*.

The images of digit *Zero* which required 3 dimensions to separate from digit *One* are

Hull Class

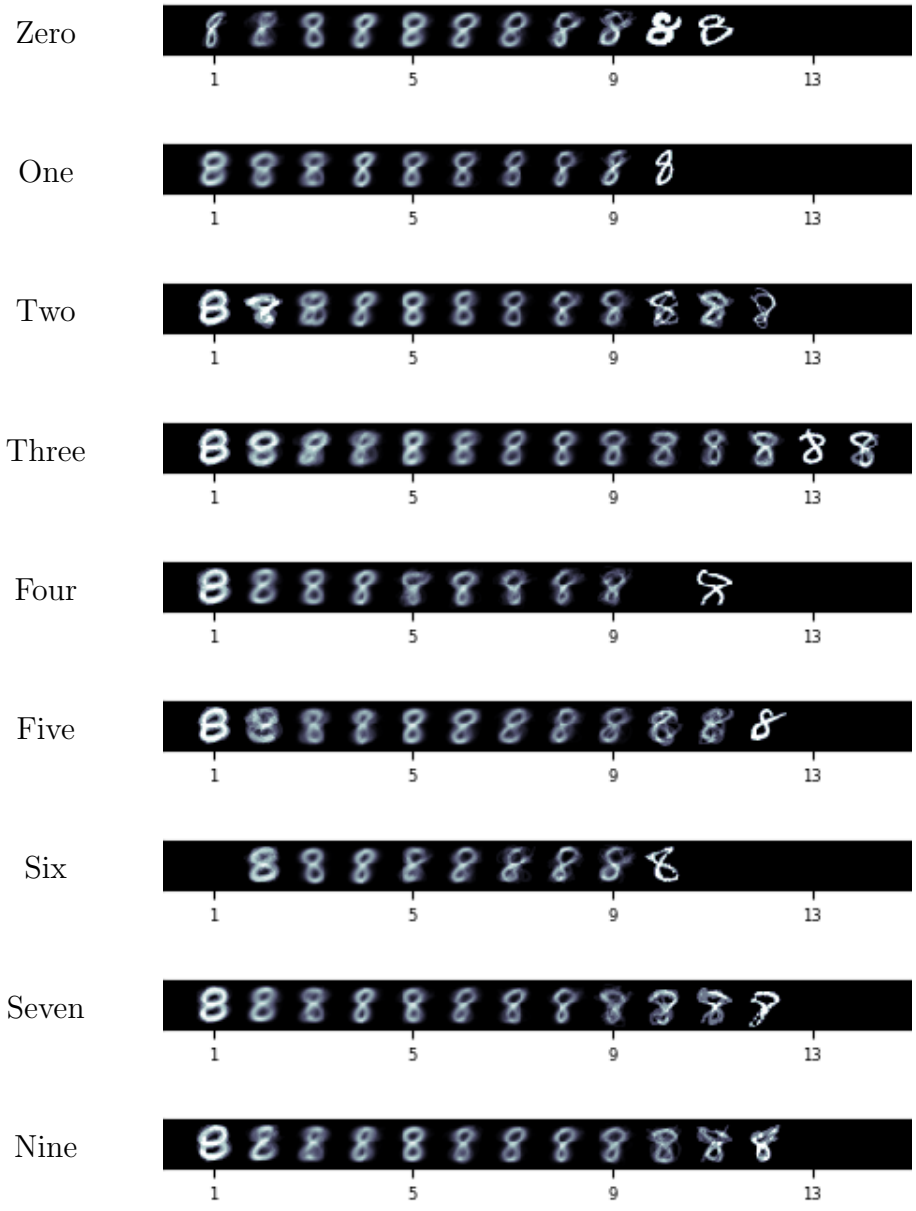


Figure 4.19: Difficulty spectra produced for MNIST digit Eight

those which are most similar to the variation of digit *One* drawn at 45°, sans-serif. Those which required 4 and 5 dimensions to separate appear more similar to *Ones* drawn at 45°and 0°(vertical) angles and with serif. Finally, the images which separated at dimensions 6 and 7 bear resemblance to vertical, sans-serif examples of digit *One*.

Figure 4.20: MNIST difficulty spectra produced for images from digit Zero which are increasingly difficult to separate from digit One.

Images in this dataset contain little to no variation in contrast, number of objects, or object's position(s). As such, using a metric which uses such attributes as a basis might provide indication that classification of MNIST images is not overtly difficult as a whole, however, such attributes would not sufficiently describe the specific class-pair difficulties observed using our method. With the exception of serif differences, the digit variations lending themselves to increased difficulty are mostly linear transformations (e.g. rotation, scale, skew).

4.4.1 Neural Networks' performance on MNIST

	Predicted Class									
	0	1	2	3	4	5	6	7	8	9
0		0.0	0.3	0.1	0.0	0.2	0.4	0.2	0.2	0.2
1	0.1		0.2	0.2	0.1	0.0	0.1	0.1	0.1	0.0
2	0.3	0.6		0.6	0.1	0.0	0.2	0.6	0.4	0.2
3	0.1	0.0	0.3		0.0	0.7	0.0	0.2	0.3	0.2
4	0.1	0.1	0.2	0.0		0.0	0.3	0.2	0.1	1.0
5	0.2	0.0	0.1	1.2	0.0		0.6	0.1	0.3	0.1
6	0.7	0.3	0.1	0.0	0.3	0.3		0.0	0.3	0.0
7	0.1	0.5	1.2	0.4	0.2	0.0	0.0		0.3	0.6
8	0.6	0.1	0.6	0.4	0.2	0.3	0.4	0.5		0.7
9	0.3	0.4	0.1	0.4	1.1	0.5	0.0	1.1	0.6	

Table 4.6: Confusion matrix for "cnn-1l-16u" trained and evaluated on the MNIST dataset [%]

	Predicted Class									
	0	1	2	3	4	5	6	7	8	9
0		0.0	0.2	0.0	0.0	0.1	0.3	0.1	0.2	0.1
1	0.0		0.3	0.1	0.2	0.0	0.2	0.1	0.1	0.0
2	0.3	0.4		0.5	0.1	0.0	0.1	0.5	0.4	0.0
3	0.1	0.0	0.2		0.0	0.5	0.0	0.2	0.3	0.2
4	0.1	0.1	0.1	0.0		0.0	0.2	0.2	0.2	1.1
5	0.2	0.0	0.0	0.9	0.0		0.4	0.1	0.2	0.3
6	0.5	0.2	0.0	0.1	0.2	0.4		0.0	0.2	0.0
7	0.0	0.3	1.0	0.3	0.1	0.0	0.0		0.2	0.7
8	0.7	0.1	0.6	0.4	0.2	0.3	0.4	0.3		0.6
9	0.2	0.3	0.1	0.4	0.7	0.4	0.0	0.8	0.3	

Table 4.7: Confusion matrix for "cnn-2l-16u" trained and evaluated on the MNIST dataset [%]

Chapter 5

CONCLUSIONS

In this paper we developed an analysis technique for quantifying the difficulty of specific image classification tasks, and investigated the human-interpetability of our results. By extracting human-interpretable features we hope to provide meaningful insight into what makes a dataset challenging to classify. Further, understanding data set complexity can be used to guide the design of neural networks, enabling autonomous vehicles to be better able to predict terrain trafficablility.

To this end we designed the ReDiHull algorithm for estimating dataset classification complexity. We used the ReDiHull to quantify the complexity of two terrain image datasets and conducted a preliminary analysis of terrain datasets collected on Mars by NASA's Jet Propulsion Laboratory and at the Canadian Space Agency Mars Emulation Terrain. Data derived from the statistical analysis was used to produce image difficulty visualisations, allowing us to identify anomalies and trends, and uncovering image characteristics that help distinguish between different classes of terrain.

The case studies presented in this work demonstrate that light-toned bedrock is easier to distinguish from sand, and darker sand is easier to distinguish from bedrock. In color images, distinguishing sand from bedrock gets an important boost from sand's lack of color gradient. Expanding off this observation, we conducted preliminary research assessing its possible application to directing classification systems design for Martian rovers.

Further, we investigated the algorithms relevance to deep learning algorithms, supporting its potential in detecting samples which are harder to correctly classify, as well as observing

demonstrable links between deep neural networks and the separating features discovered using the algorithm.

Our initial implementation of the ReDiHull algorithm uses Principle Component Analysis in order to project images into a reduced dimensionality space. PCA was selected because of its ease of use and efficient implementations. It remains an open question as to whether other projection methods would provide superior analysis of dataset complexity, and what is the interplay between projection method and the complexity of neural networks to learn a given classification task.

The results presented in this paper provide a jumping off point for the analysis of terrain classification difficulty, and can inform designers of autonomous vehicles how challenging it can be for autonomous vehicles to distinguish classes of terrain, which can aid in the design of autonomous systems, as well as provide insight into the risk associated with making traverse decisions. As AI becomes an increasing component of trafficability analysis, dataset complexity estimation becomes an important tool in the toolbox for deploying autonomous vehicles.

Bibliography

[1] D. K. Shukla and K. Skonieczny, "Simple texture descriptors for classifying monochrome planetary rover terrains," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5495–5502.

[2] A. F. R. Rahman and M. C. Fairhurst, "Measuring classification complexity of image databases: a novel approach," in *Proceedings 10th International Conference on Image Analysis and Processing*, 1999, pp. 893–897.

[3] Tin Kam Ho and H. S. Baird, "Estimating the intrinsic difficulty of a recognition problem," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5)*, vol. 2, Oct 1994, pp. 178–183 vol.2.

[4] J. N. Maki, J. F. Bell, K. E. Herkenhoff, S. W. Squyres, A. Kiely, M. Klimesh, M. Schwochert, T. Litwin, R. Willson, A. Johnson, M. Maimone, E. Baumgartner, A. Collins, M. Wadsworth, S. T. Elliot, A. Dingizian, D. Brown, E. C. Hagerott, L. Scherr, R. Deen, D. Alexander, and J. Lorre, "Mars exploration rover engineering cameras," *Journal of Geophysical Research E: Planets*, vol. 108, 12 2003. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2003JE002077 https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2003JE002077 https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2003JE002077

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] B. Rothrock, R. Kennedy, C. Cunningham, J. Papon, M. Heverly, and M. Ono, "Spoc: Deep learning-based terrain classification for mars rover missions," in *AIAA SPACE 2016*, 2016, p. 5539.

[7] R. Gonzalez and K. Iagnemma, "Deepterramechanics: Terrain classification and slip estimation for ground robots via deep learning," *arXiv preprint arXiv:1806.07379*, 2018.

[8] L. Wellhausen, A. Dosovitskiy, R. Ranftl, K. Walas, C. Cadena, and M. Hutter, "Where should i walk? predicting terrain properties from images via self-supervised learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1509–1516, 2019.

[9] Y. Iwashita, K. Nakashima, A. Stoica, and R. Kurazume, "Tu-net and tdeeplab: Deep learning-based terrain classification robust to illumination changes, combining visible and thermal imagery," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2019, pp. 280–285.

[10] R. Zhou, L. Ding, H. Gao, W. Feng, Z. Deng, and N. Li, "Mapping for planetary rovers from terramechanics perspective," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1869–1874.

[11] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. null, p. 281–305, Feb. 2012.

[12] A. Zela, A. Klein, S. Falkner, and F. Hutter, "Towards automated deep learning: Efficient joint neural architecture and hyperparameter search," *CoRR*, vol. abs/1807.06906, 2018. [Online]. Available: http://arxiv.org/abs/1807.06906

[13] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *CoRR*, vol. abs/1707.07012, 2017. [Online]. Available: http://arxiv.org/abs/1707.07012

[14] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[15] F. Scheidegger, R. Istrate, G. Mariani, L. Benini, C. Bekas, and A. C. I. Malossi, "Efficient image dataset classification difficulty estimation for predicting deep-learning accuracy," *CoRR*, vol. abs/1803.09588, 2018. [Online]. Available: http://arxiv.org/abs/1803.09588

[16] Y. Luo and X. Tang, "Photo and video quality evaluation: Focusing on the subject," in *Computer Vision – ECCV 2008*, D. Forsyth, P. Torr, and A. Zisserman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 386–399.

[17] O. Russakovsky, J. Deng, Z. Huang, A. C. Berg, and L. Fei-Fei, "Detecting avocados to zucchinis: What have we done, and where are we going?" in *2013 IEEE International Conference on Computer Vision*, Dec 2013, pp. 2064–2071.

[18] R. Tudor Ionescu, B. Alexe, M. Leordeanu, M. Popescu, D. P. Papadopoulos, and V. Ferrari, "How hard can it be? estimating the difficulty of visual search in an image," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[19] D. Liu, Y. Xiong, K. Pulli, and L. Shapiro, "Estimating image segmentation difficulty," vol. 6871, 08 2011, pp. 484–495.

[20] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," vol. 1, 11 2005, pp. 90 – 97 Vol. 1.

[21] L. Marchesotti, C. Cifarelli, and G. Csurka, "A framework for visual saliency detection with applications to image thumbnailing," 11 2009, pp. 2232 – 2239.

[22] S. Vijayanarasimhan and K. Grauman, "What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations." 01 2009, pp. 2262–2269.

[23] B. Schölkopf, J. Platt, and T. Hofmann, *Multi-Instance Multi-Label Learning with Application to Scene Classification*, 2007, pp. 1609–1616.

[24] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[25] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006. [Online]. Available: https://science.sciencemag.org/content/313/5786/504

[26] M. Vasconcelos and N. Vasconcelos, "Natural image statistics and low-complexity feature selection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, pp. 228–44, 03 2009.

[27] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[28] R. T. Rockafellar, *Convex analysis*, ser. Princeton Mathematical Series. Princeton, N. J.: Princeton University Press, 1970.

[29] E. Andersen and K. Andersen, "The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm," vol. 33, 01 1999.

[30] J. M.A and H. Fleyeh, "Convex hulls in image processing: A scoping review," *American Journal of Intelligent Systems*, vol. 2016, pp. 48–58, 05 2016.

[31] K. Zhao, A. Wiliem, S. Chen, and B. C. Lovell, "Manifold convex hull (mach): Satisfying a need for spd," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 251–255.

[32] X. Zhou and Y. Shi, "Nearest neighbor convex hull classification method for face recognition," in *Computational Science - ICCS 2009, 9th International Conference, Baton Rouge, LA, USA, May 25-27, 2009, Proceedings, Part II*, ser. Lecture Notes in Computer Science, G. Allen, J. Nabrzyski, E. Seidel, G. D. van Albada, J. J. Dongarra, and P. M. A. Sloot, Eds., vol. 5545. Springer, 2009, pp. 570–577. [Online]. Available: https://doi.org/10.1007/978-3-642-01973-9_64

[33] H. Cevikalp, D. Larlus, M. Neamtu, B. Triggs, and F. Jurie, "Manifold based local classifiers: Linear and nonlinear approaches," *Signal Processing Systems*, vol. 61, pp. 61–73, 10 2010.

[34] G. Nalbantov, P. Groenen, and J. Bioch, "Nearest convex hull classification," Erasmus University Rotterdam, Erasmus School of Economics (ESE), Econometric Institute, Econometric Institute Research Papers EI 2006-50, Dec. 2006. [Online]. Available: https://ideas.repec.org/p/ems/eureir/8217.html

[35] K. Raimalwala, M. Faragalli, E. Smal, M. Battler, E. Reid, B. Stefanuk, and K. Skonieczny, "Enabling autonomy in commercial-class lunar missions," 2020.

[36] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278 – 2324, 12 1998.

[37] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[38] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381

[39] M. Malin and K. Edgett, "Mast camera (mastcam), mars hand lens imager (mahli), and mars descent imager (mardi) experiment data record (edr) and reduced data record (rdr) pds data products. jpl d-75410, sis-sci035-msl," 9 2013.

[40] S. Dieleman, J. D. Fauw, and K. Kavukcuoglu, "Exploiting cyclic symmetry in convolutional neural networks," *CoRR*, vol. abs/1602.02660, 2016. [Online]. Available: http://arxiv.org/abs/1602.02660

[41] T. S. Cohen and M. Welling, "Group equivariant convolutional networks," 2016.

[42] A. Raghavan, M. R. Amer, and S. M. Chai, "Bitnet: Bit-regularized deep neural networks," *CoRR*, vol. abs/1708.04788, 2017. [Online]. Available: http://arxiv.org/abs/1708.04788

Appendix A

Martian Terrain - Supplementary Results

A.1 ReDiHull & Neural Network Sample Difficulty

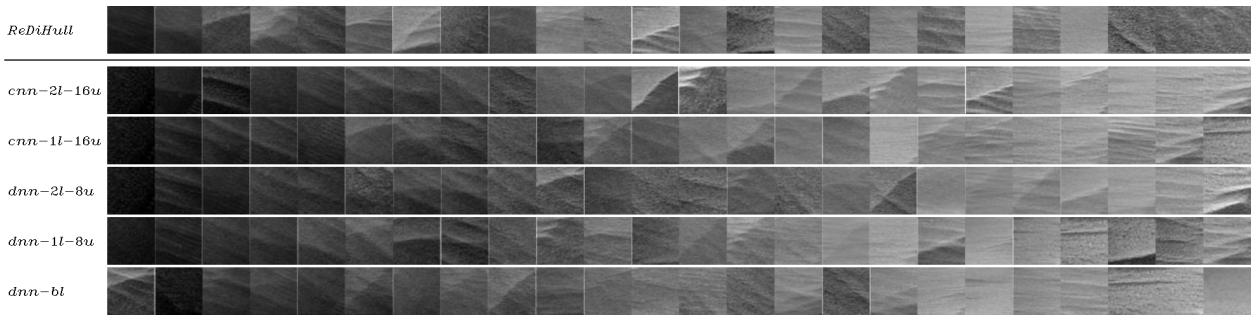


Figure A.1: Increasing difficulty of *sand* terrain samples (true label) w.r.t *bedrock* terrain samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

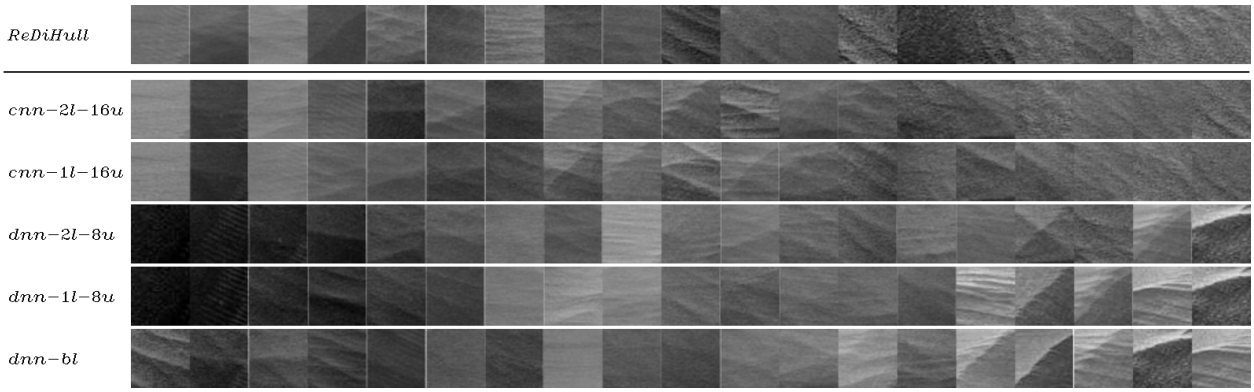


Figure A.2: Increasing difficulty of *sand* terrain samples (true label) w.r.t *rock-strewn* terrain samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

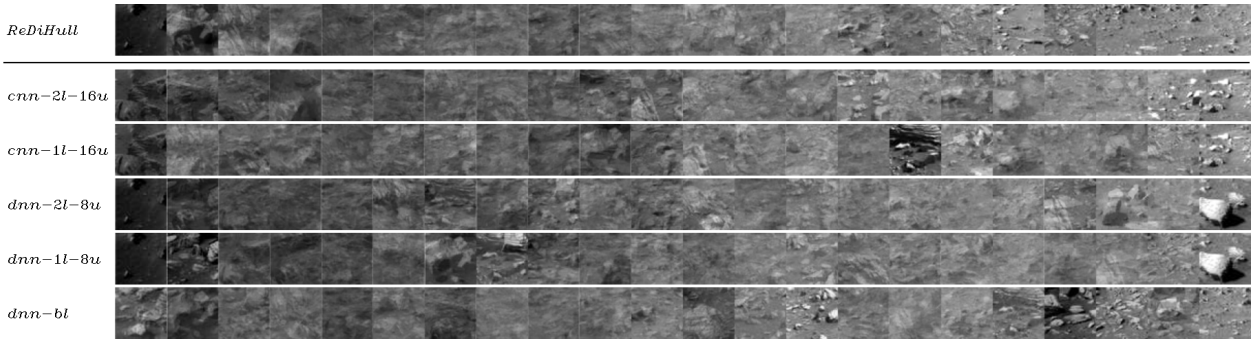


Figure A.3: Increasing difficulty of *rock-strewn* terrain samples (true label) w.r.t *bedrock* terrain samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

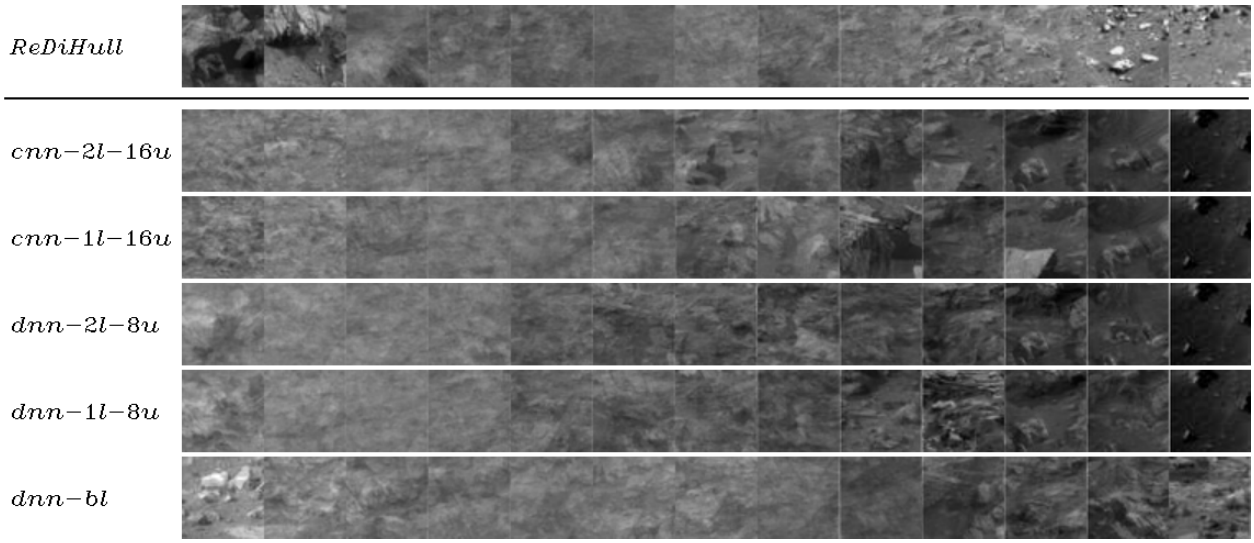


Figure A.4: Increasing difficulty of *rock-strewn* terrain samples (true label) w.r.t *sand* terrain samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

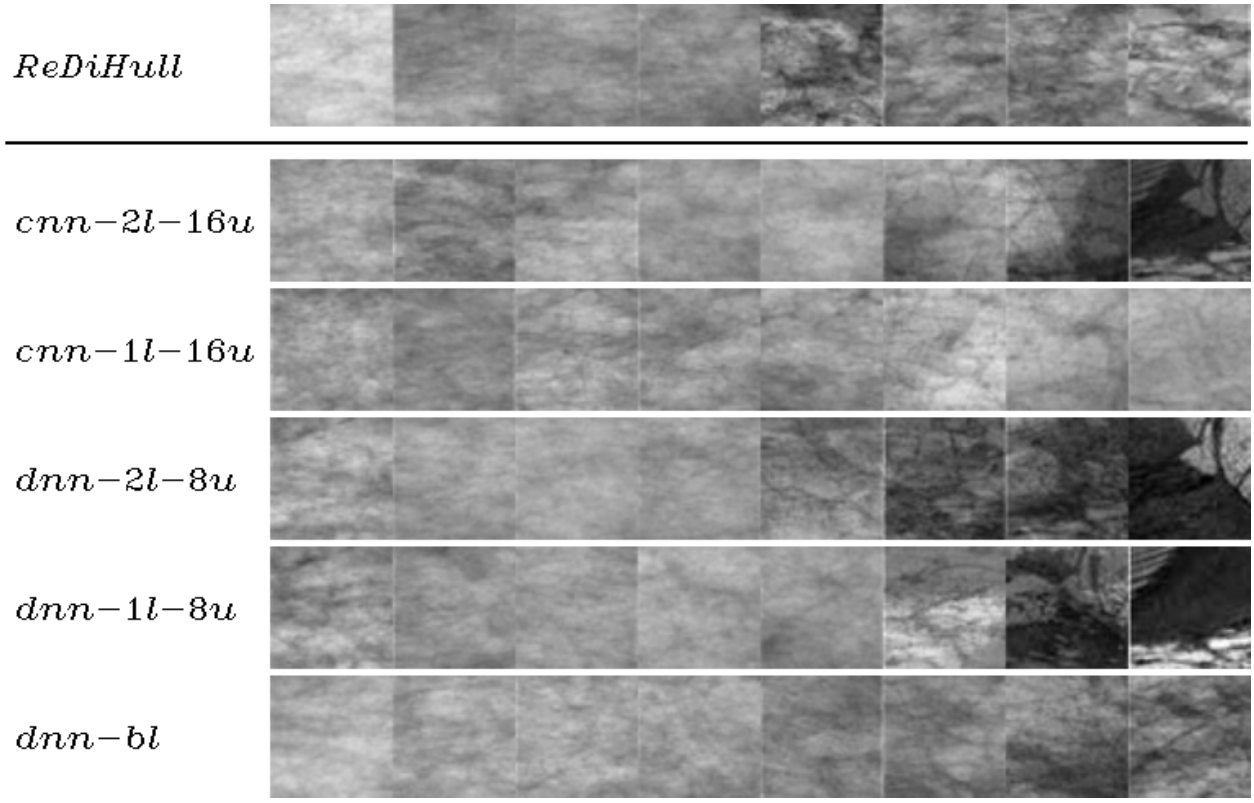


Figure A.5: Increasing difficulty of *bedrock* terrain samples (true label) w.r.t *sand* terrain samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

A.2 ReDiHull-Computed Dataset Difficulty

		Predicted Class	
	Bedrock	Rocks	Sand
Bedrock		1.54	1.65
Rocks	0.64		0.63
Sand	1.23	0.64	

Table A.1: Martian Terrain difficulty quantified using KL divergence

A.3 Neural Network Error Matrices

		Predicted Class	
	Bedrock	Rocks	Sand
Bedrock		16.3	10.4
Rocks	12.8		16.2
Sand	11.1	14.8	

Table A.2: Average sample prediction error matrix for "dnn-bl" trained and evaluated on the Martian Terrain (original Dhara JPL) dataset [%]

		Predicted Class	
	Bedrock	Rocks	Sand
Bedrock		13.9	14.1
Rocks	14.9		17.6
Sand	10.2	13.8	

Table A.3: Average sample prediction error matrix for "dnn-1l-8u" trained and evaluated on the Martian Terrain (original Dhara JPL) dataset [%]

	Predicted Class		
	Bedrock	Rocks	Sand
Bedrock		15.0	13.6
Rocks	15.0		16.3
Sand	12.0	14.6	

Table A.4: Average sample prediction error matrix for "dnn-2l-8u" trained and evaluated on the Martian Terrain (original Dhara JPL) dataset [%]

	Predicted Class		
	Bedrock	Rocks	Sand
Bedrock		14.7	17.0
Rocks	18.6		16.3
Sand	13.6	12.7	

Table A.5: Average sample prediction error matrix for "cnn-1l-16u" trained and evaluated on the Martian Terrain (original Dhara JPL) dataset [%]

	Predicted Class		
	Bedrock	Rocks	Sand
Bedrock		16.8	11.2
Rocks	21.4		16.0
Sand	17.4	16.3	

Table A.6: Average sample prediction error matrix for "cnn-2l-16u" trained and evaluated on the Martian Terrain (original Dhara JPL) dataset [%]

Appendix B

CSA MET - Supplementary Results

B.1 ReDiHull & Neural Network Sample Difficulty

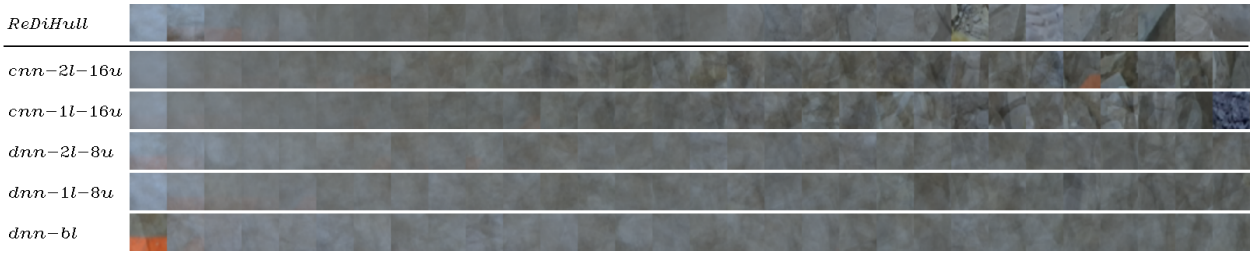


Figure B.1: Increasing difficulty of *bedrock* image samples (true label) w.r.t *gravel* image samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

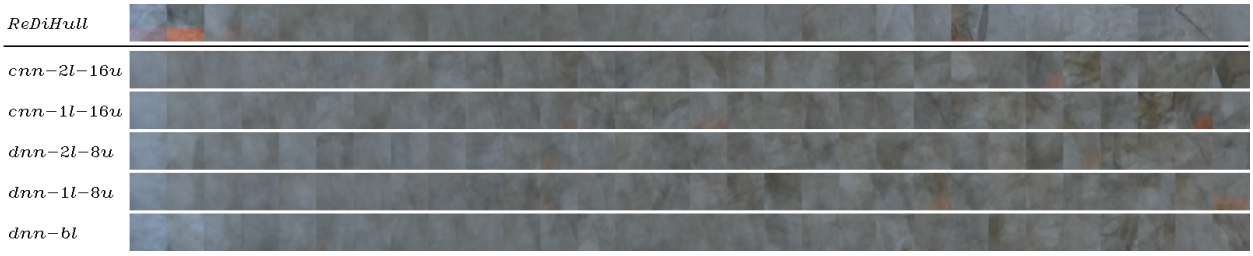


Figure B.2: Increasing difficulty of *bedrock* image samples (true label) w.r.t *sand* image samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

Figure B.3: Increasing difficulty of *bricks* image samples (true label) w.r.t *black sand* image samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

Figure B.4: Increasing difficulty of *bricks* image samples (true label) w.r.t *gravel* image samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

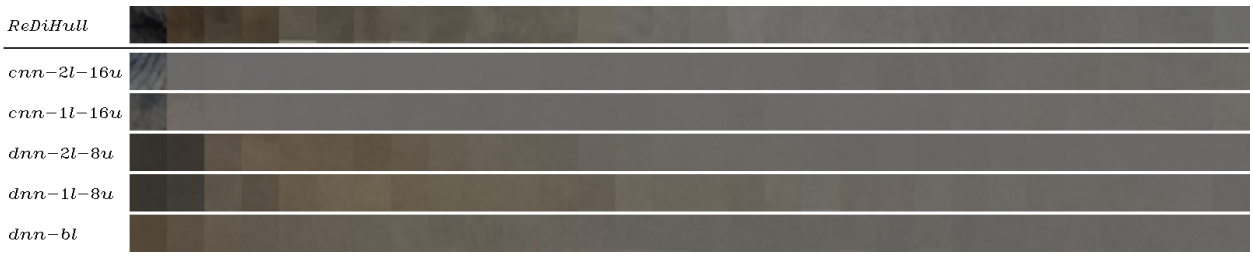


Figure B.5: Increasing difficulty of *sand* image samples (true label) w.r.t *bedrock* image samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

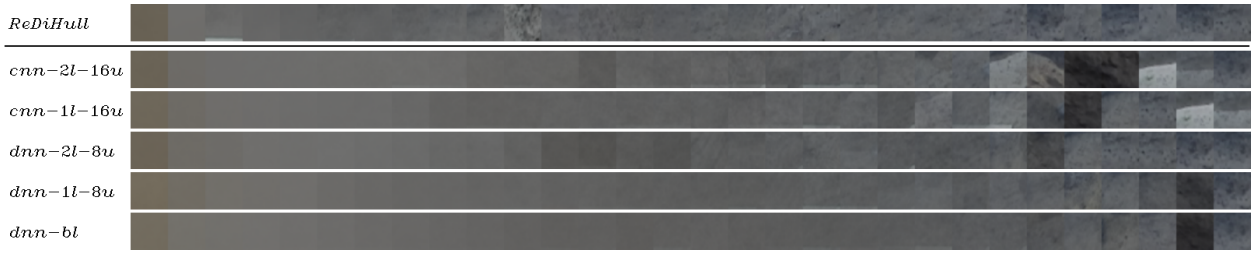


Figure B.6: Increasing difficulty of *sand* image samples (true label) w.r.t *bricks* image samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

B.2 ReDiHull-Computed Dataset Difficulty

	Predicted Class				
	Bedrock	Sand	Black Sand	Gravel	Bricks
Bedrock		1.18	0.54	1.05	1.68
Sand	2.71		2.50	4.53	4.68
Black sand	0.59	1.13		0.71	1.16
Gravel	0.32	1.51	0.27		0.31
Bricks	0.55	1.00	0.69	0.21	

Table B.1: CSA MET difficulty quantified using KL divergence.

B.3 Neural Network Error Matrices

		Predicted Class			
	Bedrock	Sand	Black Sand	Gravel	Bricks
Bedrock		10.7	5.6	15.7	2.6
Sand	14.9		9.5	5.6	1.1
Black sand	2.0	7.5		8.1	13.5
Gravel	7.0	4.0	11.0		18.3
Bricks	2.4	0.8	10.0	11.6	

Table B.2: Average sample prediction error matrix for "dnn-bl" trained and evaluated on the CSA MET dataset [%]

		Predicted Class			
	Bedrock	Sand	Black Sand	Gravel	Bricks
Bedrock		8.0	4.0	11.4	3.6
Sand	9.0		12.0	5.0	1.0
Black sand	2.2	8.5		8.4	12.2
Gravel	5.2	3.1	9.2		22.0
Bricks	2.2	0.5	11.1	11.0	

Table B.3: Average sample prediction error matrix for "dnn-1l-8u" trained and evaluated on the CSA MET dataset [%]

	Predicted Class				
	Bedrock	Sand	Black Sand	Gravel	Bricks
Bedrock		6.2	3.7	12.1	3.5
Sand	6.3		11.4	4.0	0.8
Black sand	2.1	9.3		6.6	12.5
Gravel	5.4	2.8	8.5		23.8
Bricks	2.1	0.7	12.2	10.4	

Table B.4: Average sample prediction error matrix for "dnn-2l-8u" trained and evaluated on the CSA MET dataset [%]

	Predicted Class				
	Bedrock	Sand	Black Sand	Gravel	Bricks
Bedrock		11.7	2.6	3.0	4.8
Sand	5.1		9.2	0.9	0.8
Black sand	2.7	9.1		6.4	10.2
Gravel	0.7	0.3	8.1		15.5
Bricks	2.3	0.6	8.8	4.4	

Table B.5: Average sample prediction error matrix for "cnn-2l-16u" trained and evaluated on the CSA MET dataset [%]

	Predicted Class				
	Bedrock	Sand	Black Sand	Gravel	Bricks
Bedrock		11.7	3.5	4.4	4.7
Sand	5.6		9.8	1.3	1.0
Black sand	2.8	8.5		6.2	10.8
Gravel	1.3	0.3	10.4		15.4
Bricks	2.5	0.7	12.6	5.7	

Table B.6: Average sample prediction error matrix for "cnn-1l-16u" trained and evaluated on the CSA MET dataset [%]

Appendix C

CIFAR-10 - Supplementary Results

C.1 ReDiHull-Computed Dataset Difficulty

	Hull Class									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane		0.27	0.36	0.30	0.53	0.31	1.24	0.37	0.13	0.31
Automobile	0.87		1.34	0.46	1.39	0.47	1.45	0.45	0.49	0.10
Bird	0.11	0.22		0.23	0.12	0.22	0.13	0.16	0.18	0.21
Cat	0.51	0.18	0.47		0.63	0.07	0.55	0.09	0.52	0.22
Deer	0.11	0.26	0.10	0.34		0.23	0.06	0.23	0.25	0.22
Dog	0.65	0.25	0.51	0.08	0.81		0.69	0.11	0.56	0.28
Frog	0.18	0.25	0.20	0.24	0.19	0.20		0.18	0.31	0.23
Horse	0.47	0.08	0.48	0.15	0.51	0.11	0.70		0.62	0.08
Ship	0.20	0.20	0.59	0.32	0.68	0.50	1.61	0.49		0.15
Truck	0.81	0.06	1.27	0.34	1.21	0.52	1.55	0.30	0.51	

Table C.1: CIFAR-10 difficulty quantified using KL divergence

C.2 Neural Network Error Matrices

					Predicted Class					
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane		2.7	3.2	2.3	2.1	2.0	1.4	2.9	7.2	3.3
Automobile	3.1		2.2	2.5	1.9	2.2	2.1	2.4	4.1	6.8
Bird	3.8	2.0		4.1	5.6	4.1	4.9	4.1	2.5	1.8
Cat	2.4	2.7	4.5		4.1	6.1	5.1	3.3	2.2	2.6
Deer	2.4	1.7	6.0	3.9		4.2	5.8	4.7	1.6	1.6
Dog	2.6	2.3	4.6	6.3	4.5		4.1	3.8	2.5	1.8
Frog	1.3	1.9	4.5	5.5	5.4	4.2		2.9	1.2	1.8
Horse	2.8	2.2	4.0	3.4	4.7	3.6	2.8		2.1	3.1
Ship	7.2	4.0	2.2	2.1	1.4	2.0	1.1	1.6		4.7
Truck	3.7	7.2	1.9	2.1	1.7	1.6	2.1	2.9	4.4	

Table C.2: Average sample prediction error matrix for "dnn-bl" trained and evaluated on the CIFAR-10 dataset [%]

	Predicted Class									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane		2.7	3.7	2.1	2.3	1.7	1.2	2.5	7.5	3.3
Automobile	3.1		1.7	2.2	1.5	1.9	1.7	2.0	4.1	7.9
Bird	4.4	1.8		4.0	6.3	3.9	5.4	4.1	2.3	1.7
Cat	2.6	2.5	4.3		3.9	6.7	5.6	3.4	2.4	2.5
Deer	2.8	1.4	6.4	3.7		3.7	6.6	4.9	1.5	1.5
Dog	2.3	2.0	4.2	6.9	4.0		4.3	4.0	2.5	1.8
Frog	1.4	1.6	5.0	5.5	6.1	4.3		2.5	1.2	1.5
Horse	3.1	2.1	3.7	3.4	4.7	3.8	2.4		1.8	3.5
Ship	6.6	4.0	1.9	2.0	1.5	2.0	1.0	1.3		4.6
Truck	3.6	8.9	1.3	2.0	1.3	1.5	1.6	2.9	4.6	

Table C.3: Average sample prediction error matrix for "dnn-1l-8u" trained and evaluated on the CIFAR-10 dataset [%]

	Predicted Class									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane		2.6	3.7	2.2	2.7	1.9	1.3	2.5	8.3	2.9
Automobile	3.0		1.5	2.1	1.3	1.8	1.5	1.8	4.3	7.9
Bird	4.0	1.6		4.4	6.7	4.4	5.3	3.9	2.1	1.5
Cat	2.1	2.2	4.1		3.9	7.5	5.6	3.2	2.1	2.3
Deer	2.7	1.2	6.4	4.1		4.1	6.4	4.3	1.4	1.3
Dog	1.9	1.6	4.1	7.3	4.0		4.4	3.7	2.1	1.5
Frog	1.2	1.4	4.7	5.9	5.9	4.8		2.3	1.0	1.4
Horse	2.8	2.0	3.9	3.7	4.8	4.3	2.4		1.6	3.2
Ship	7.1	4.0	1.9	2.3	1.4	2.1	1.0	1.2		4.0
Truck	3.4	8.3	1.3	2.2	1.3	1.7	1.5	2.7	4.9	

Table C.4: Average sample prediction error matrix for "dnn-2l-8u" trained and evaluated on the CIFAR-10 dataset [%]

	Predicted Class									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane		2.2	4.6	2.0	2.4	1.0	1.0	1.2	8.2	3.5
Automobile	3.1		0.9	1.3	0.7	0.7	0.9	0.8	4.0	9.3
Bird	5.3	0.8		6.1	8.1	6.5	5.1	4.2	1.7	1.1
Cat	2.3	1.2	6.1		6.1	11.7	5.6	3.7	1.8	1.6
Deer	2.4	0.5	7.9	6.0		4.2	5.4	6.0	1.2	0.7
Dog	1.4	0.6	6.6	12.0	4.4		2.7	5.4	1.2	0.9
Frog	1.0	0.8	4.3	6.2	5.3	3.0		1.3	0.7	0.7
Horse	1.7	0.6	3.6	4.1	5.4	5.4	1.0		0.6	1.7
Ship	7.9	3.9	1.5	1.4	1.0	0.9	0.7	0.6		3.8
Truck	3.6	7.8	1.0	1.5	0.9	1.0	0.7	2.2	3.6	

Table C.5: Average sample prediction error matrix for "cnn-1l-16u" trained and evaluated on the CIFAR-10 dataset [%]

	Predicted Class									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane		2.2	5.2	2.2	2.5	1.1	0.9	1.4	7.7	3.3
Automobile	2.4		0.8	1.0	0.5	0.6	0.9	0.7	2.9	8.8
Bird	5.3	0.8		5.9	7.9	6.6	5.5	4.1	1.5	1.0
Cat	2.2	1.2	6.1		5.6	14.0	6.3	3.7	1.7	1.7
Deer	2.2	0.5	7.9	5.7		4.7	6.0	6.5	1.1	0.6
Dog	1.3	0.5	5.5	12.2	4.0		2.7	5.2	1.0	0.9
Frog	0.8	0.8	4.5	5.9	5.0	3.3		1.1	0.8	0.7
Horse	1.7	0.7	3.7	3.7	5.5	5.7	1.0		0.5	1.6
Ship	7.1	3.8	1.6	1.6	1.0	0.9	0.8	0.7		3.1
Truck	3.1	7.9	0.9	1.6	0.8	1.1	0.7	2.0	2.8	

Table C.6: Average sample prediction error [%] for "cnn-2l-16u" trained and evaluated on the CIFAR-10 dataset

Appendix D

MNIST - Supplementary Results

D.1 ReDiHull & Neural Network Sample Difficulty

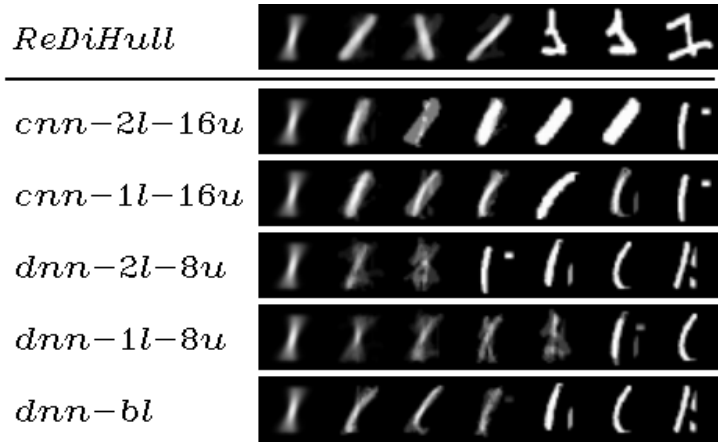


Figure D.1: Increasing difficulty of digit One samples (true label) w.r.t digit Zero samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

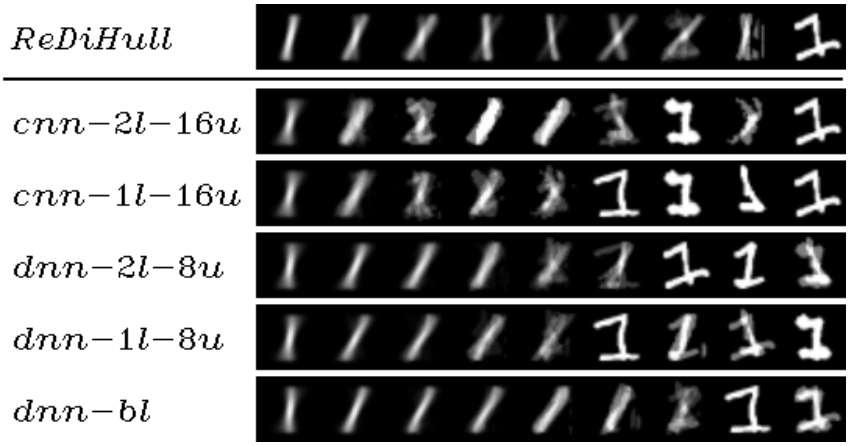


Figure D.2: Increasing difficulty of digit One samples (true label) w.r.t digit Two samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

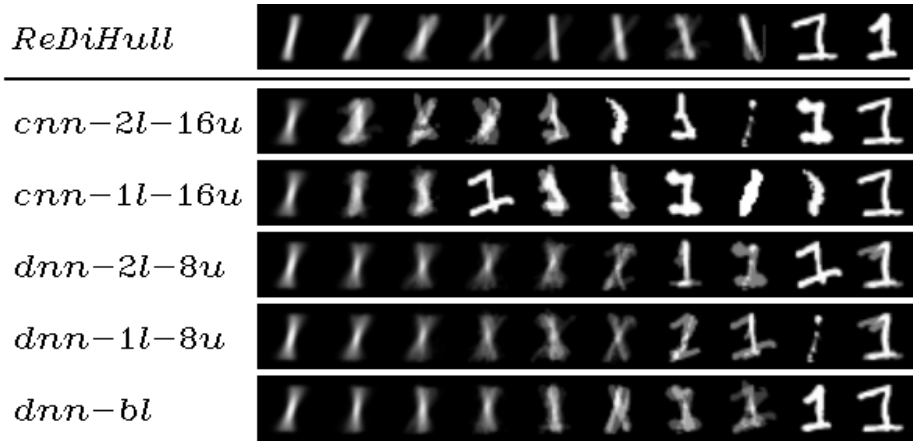


Figure D.3: Increasing difficulty of digit One samples (true label) w.r.t digit Three samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

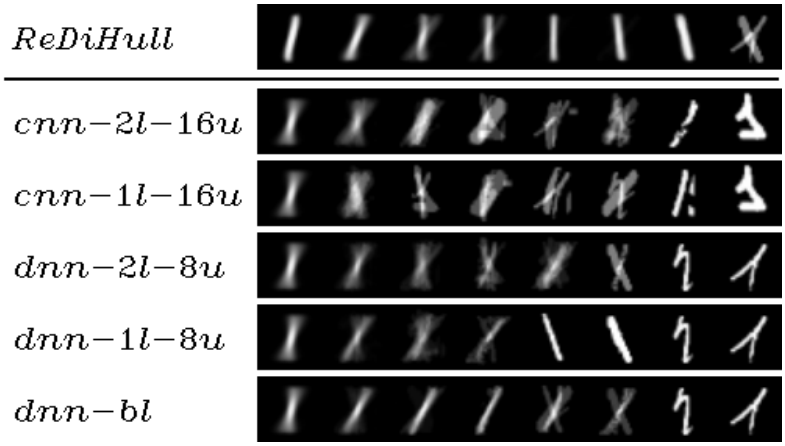


Figure D.4: Increasing difficulty of digit One samples (true label) w.r.t digit Four samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

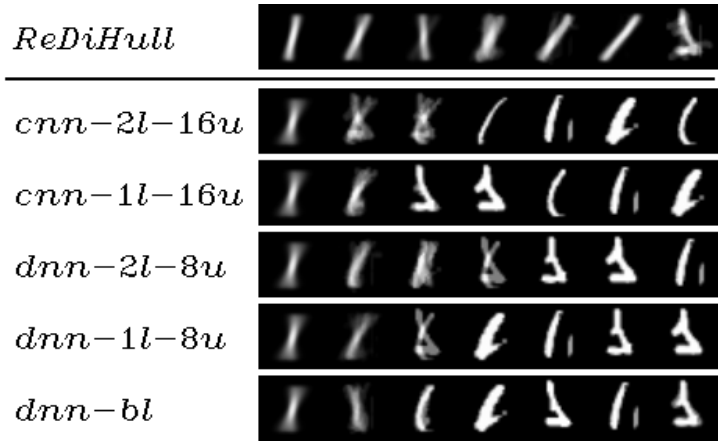


Figure D.5: Increasing difficulty of digit One samples (true label) w.r.t digit Six samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

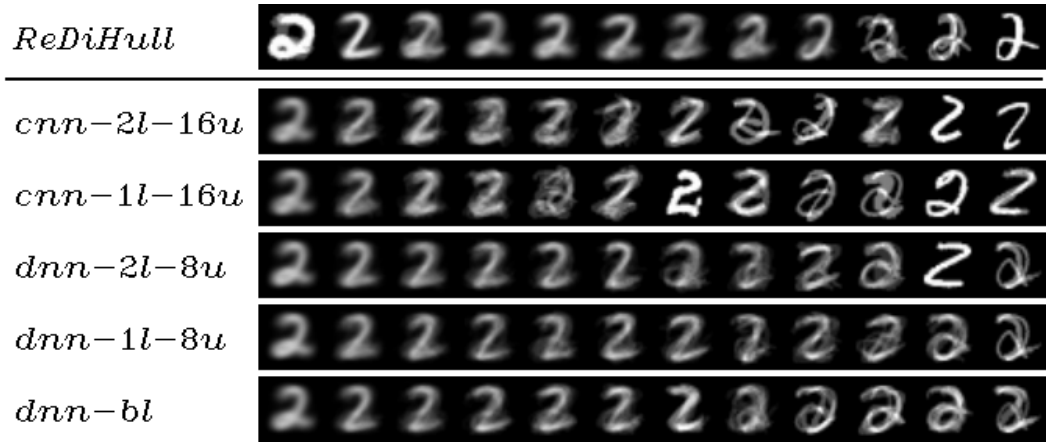


Figure D.6: Increasing difficulty of digit Two samples (true label) w.r.t digit Three samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

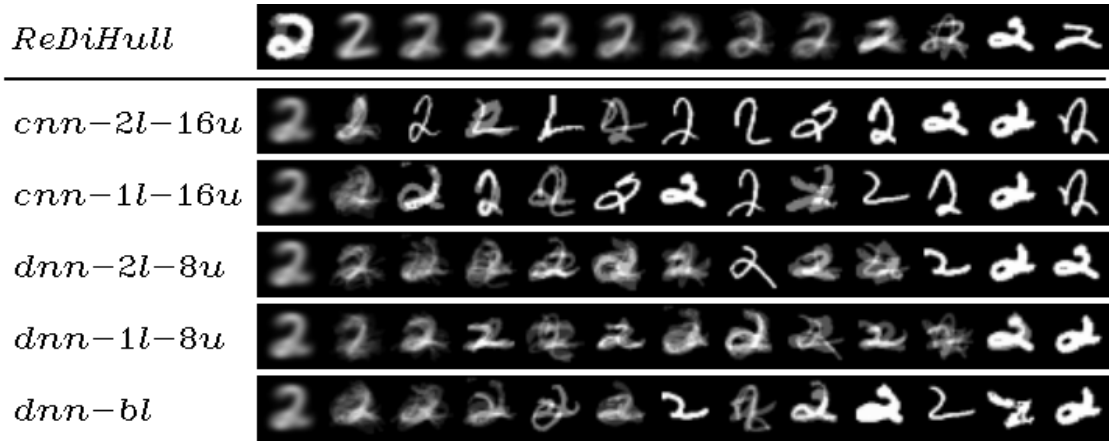


Figure D.7: Increasing difficulty of digit Two samples (true label) w.r.t digit Four samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

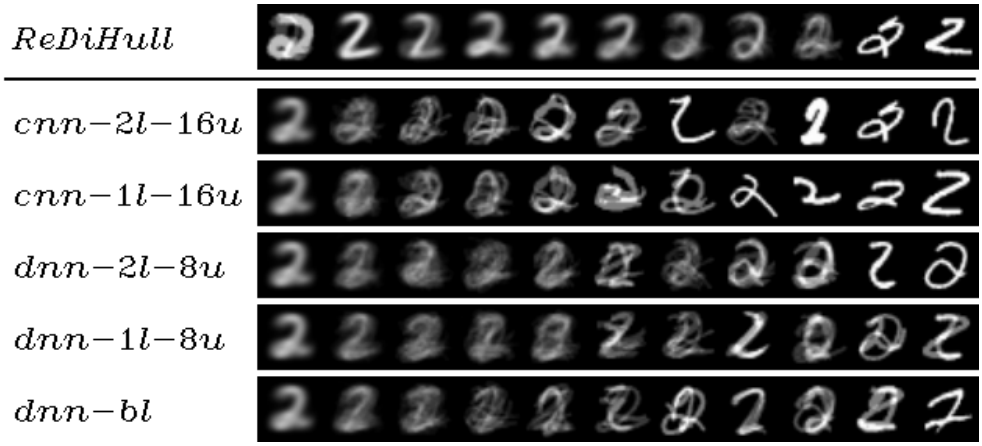


Figure D.8: Increasing difficulty of digit Two samples (true label) w.r.t digit Five samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

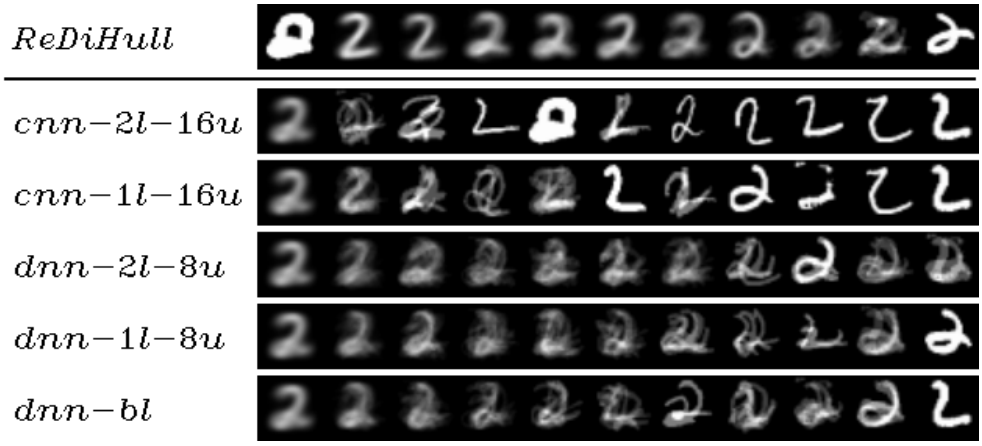


Figure D.9: Increasing difficulty of digit Two samples (true label) w.r.t digit Six samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

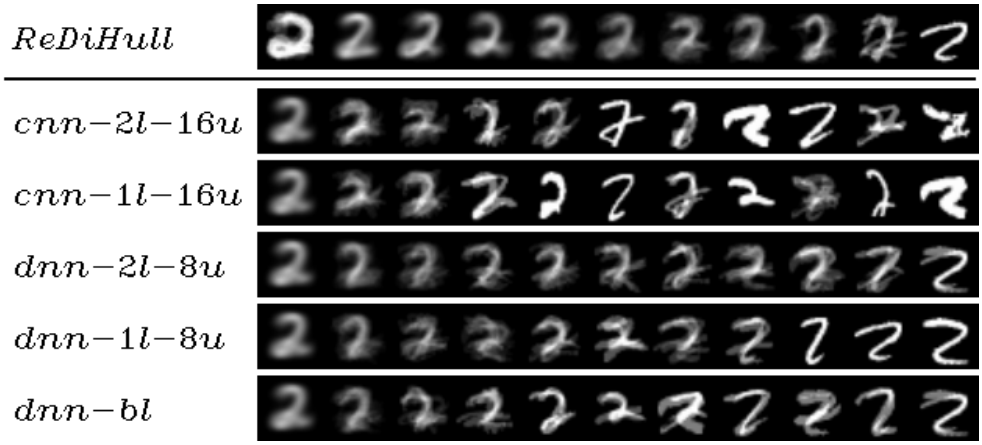


Figure D.10: Increasing difficulty of digit Two samples (true label) w.r.t digit Seven samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

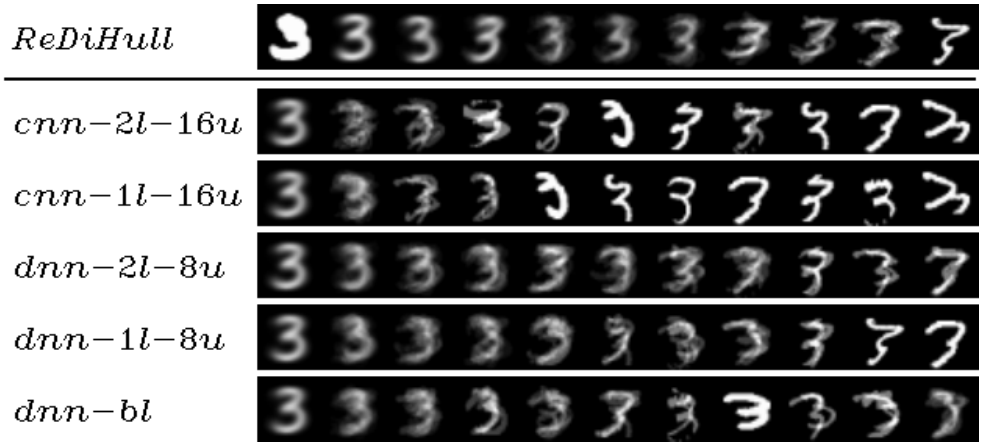


Figure D.11: Increasing difficulty of digit Three samples (true label) w.r.t digit Seven samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

Figure D.12: Increasing difficulty of digit Four samples (true label) w.r.t digit Zero samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

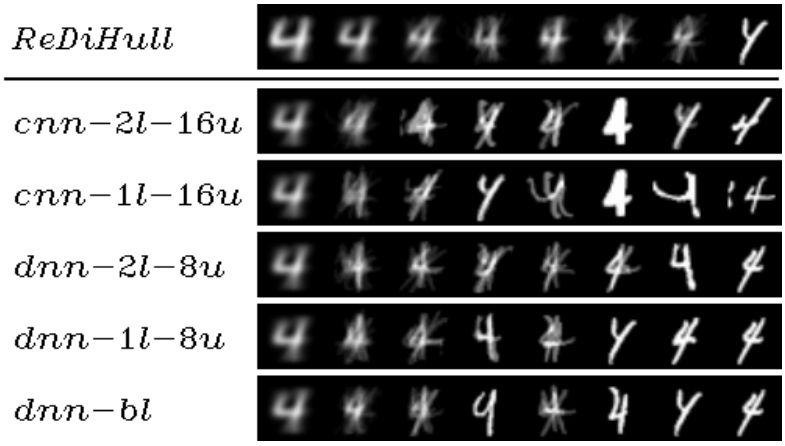


Figure D.13: Increasing difficulty of digit Four samples (true label) w.r.t digit One samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

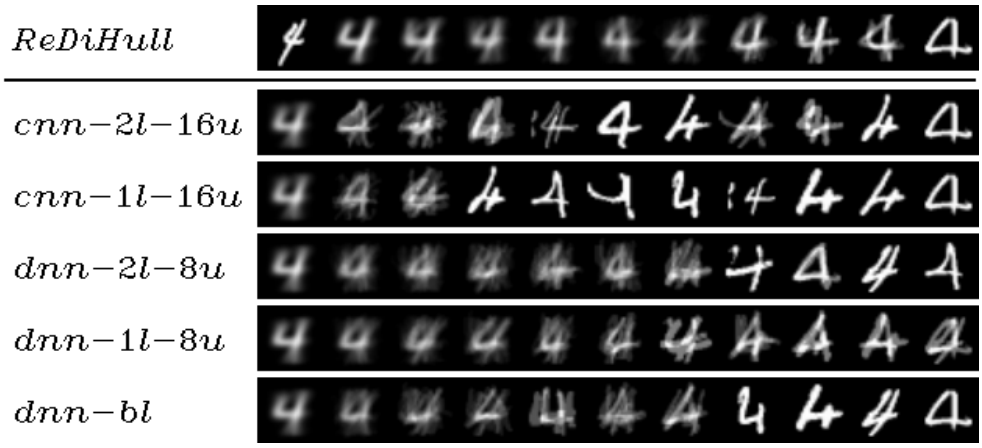


Figure D.14: Increasing difficulty of digit Four samples (true label) w.r.t digit Two samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

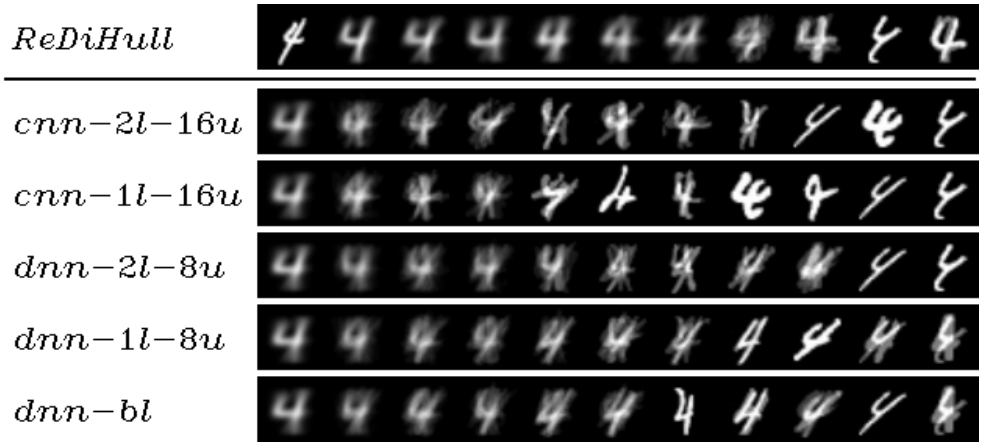


Figure D.15: Increasing difficulty of digit Four samples (true label) w.r.t digit Eight samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

Figure D.16: Increasing difficulty of digit Five samples (true label) w.r.t digit Two samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

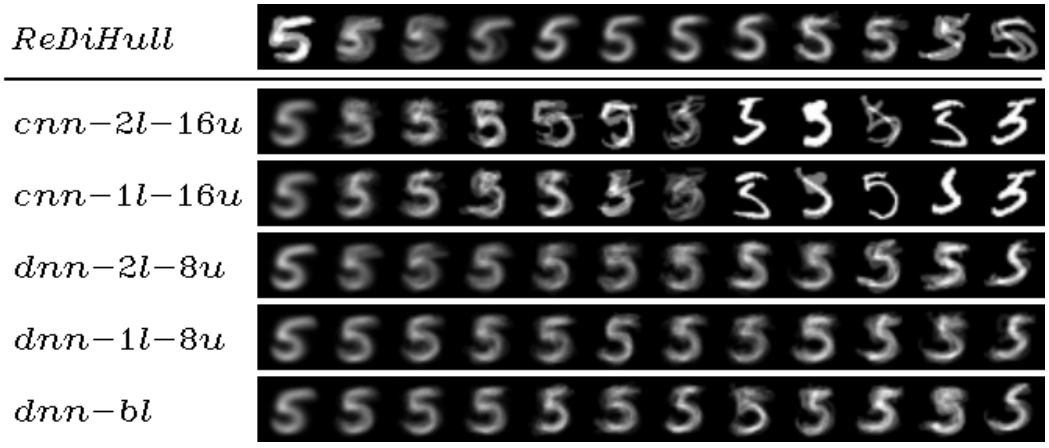


Figure D.17: Increasing difficulty of digit Five samples (true label) w.r.t digit Three samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

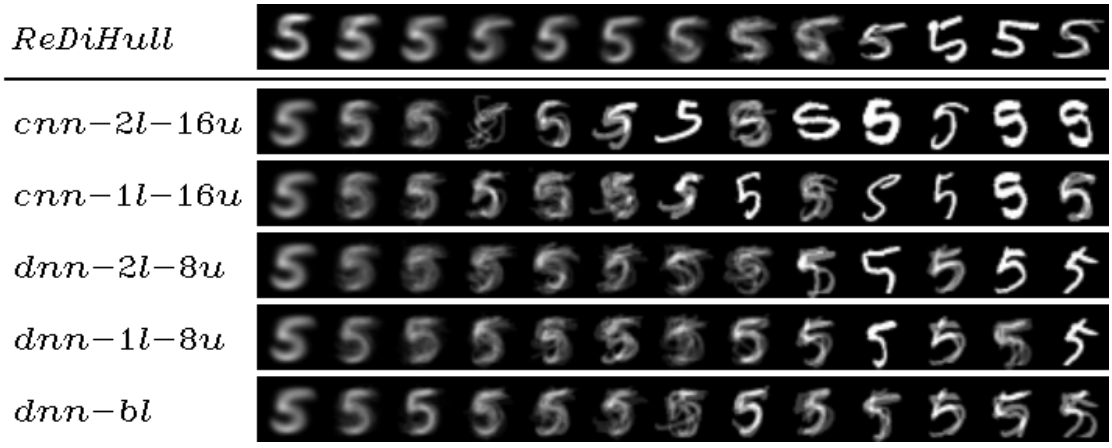


Figure D.18: Increasing difficulty of digit Five samples (true label) w.r.t digit Nine samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

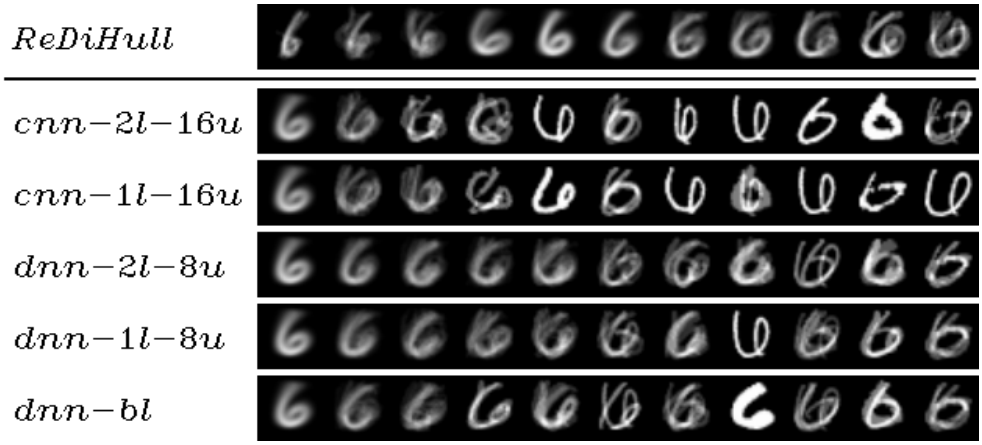


Figure D.19: Increasing difficulty of digit Six samples (true label) w.r.t digit Zero samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

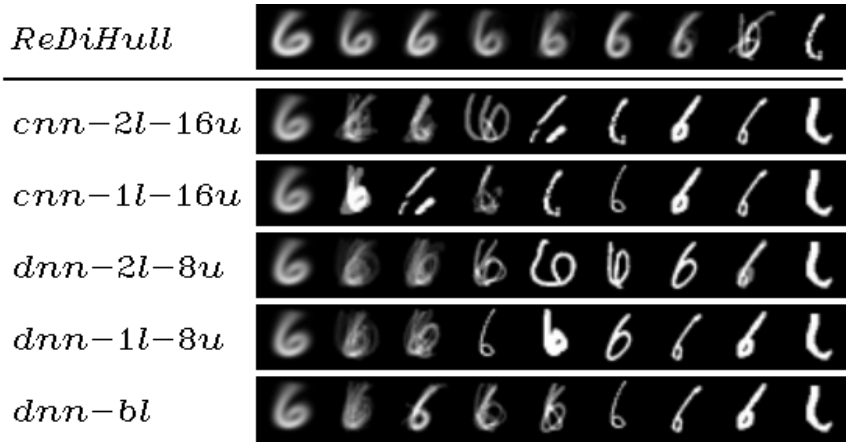


Figure D.20: Increasing difficulty of digit Six samples (true label) w.r.t digit One samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

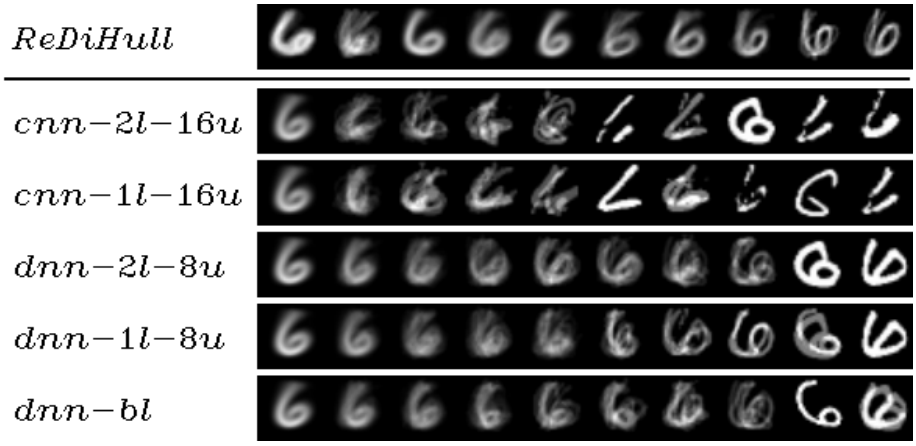


Figure D.21: Increasing difficulty of digit Six samples (true label) w.r.t digit Two samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

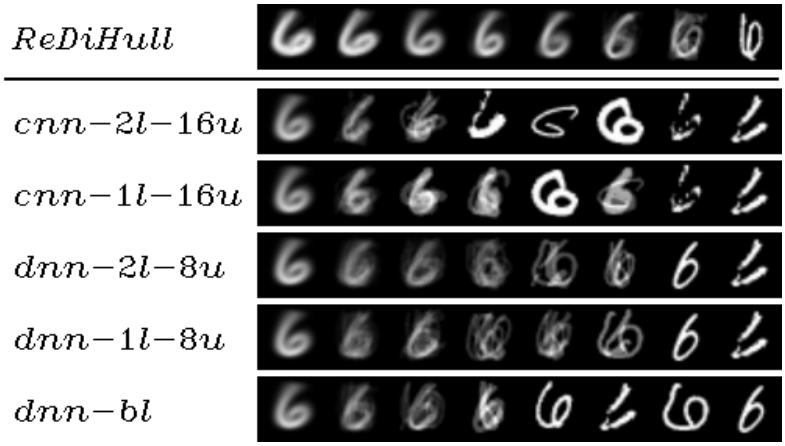


Figure D.22: Increasing difficulty of digit Six samples (true label) w.r.t digit Three samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

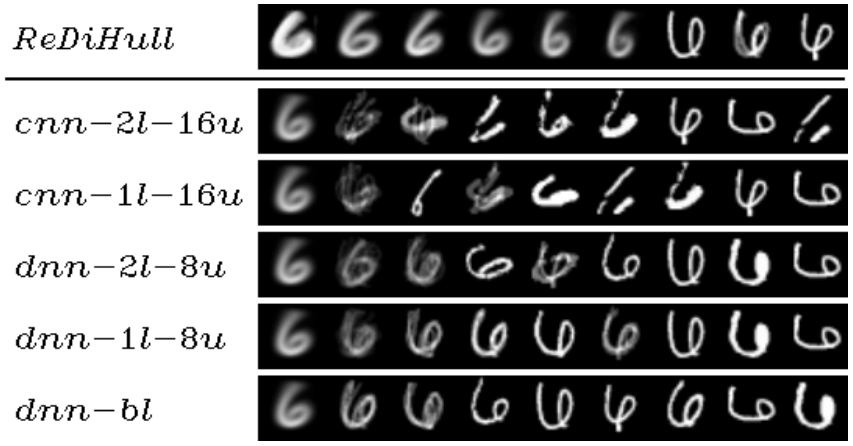


Figure D.23: Increasing difficulty of digit Six samples (true label) w.r.t digit Seven samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

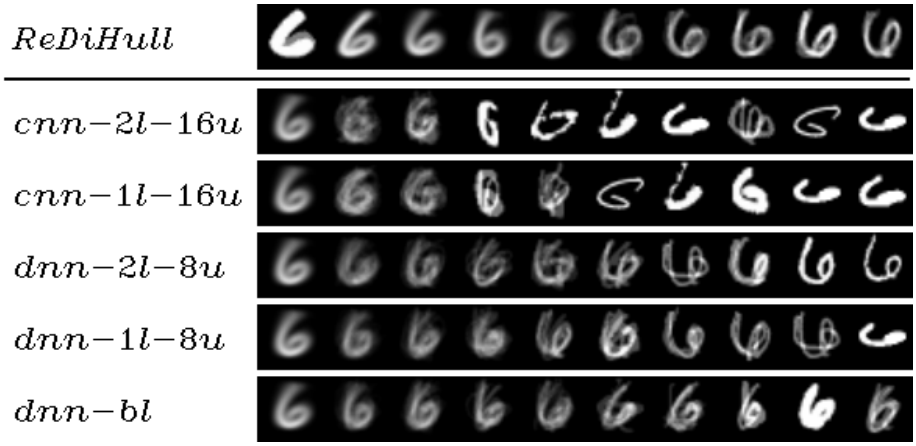


Figure D.24: Increasing difficulty of digit Six samples (true label) w.r.t digit Nine samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

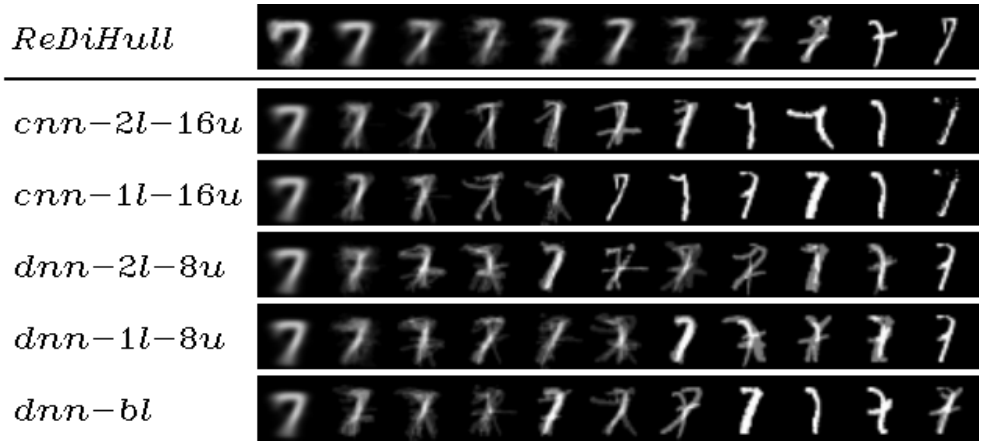


Figure D.25: Increasing difficulty of digit Seven samples (true label) w.r.t digit One samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

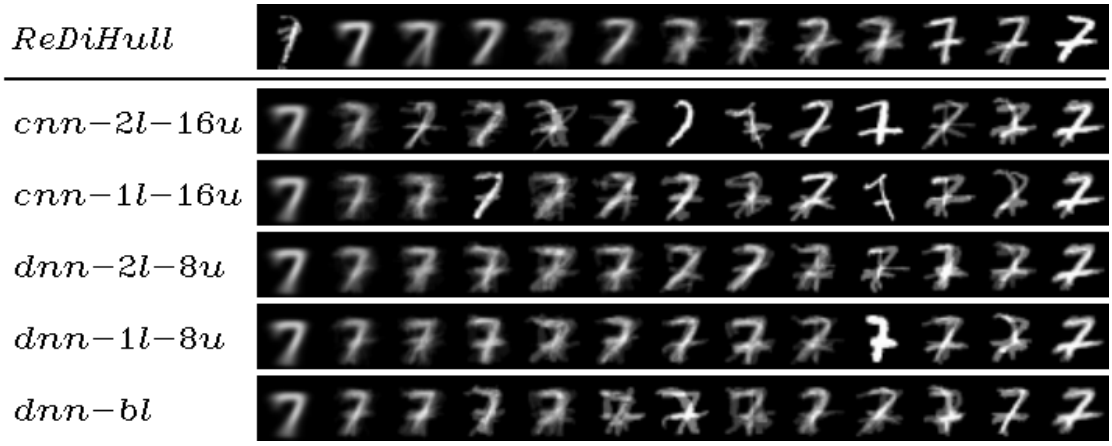


Figure D.26: Increasing difficulty of digit Seven samples (true label) w.r.t digit Two samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

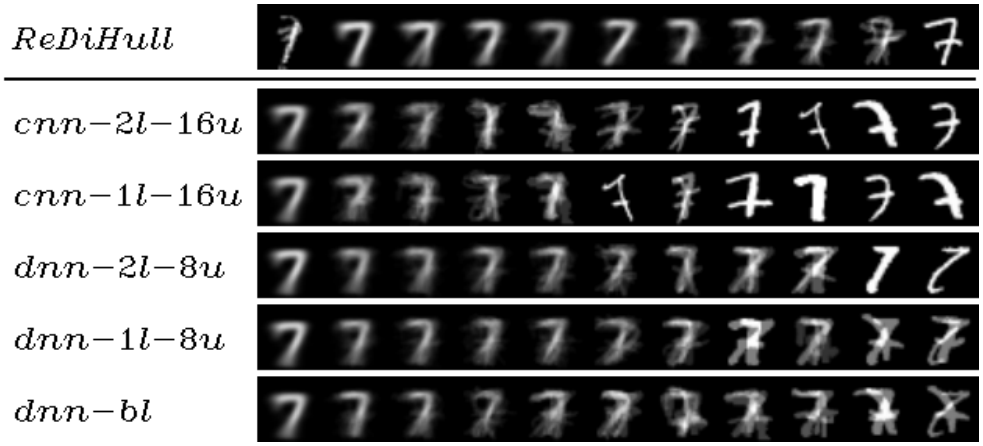


Figure D.27: Increasing difficulty of digit Seven samples (true label) w.r.t digit Three samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

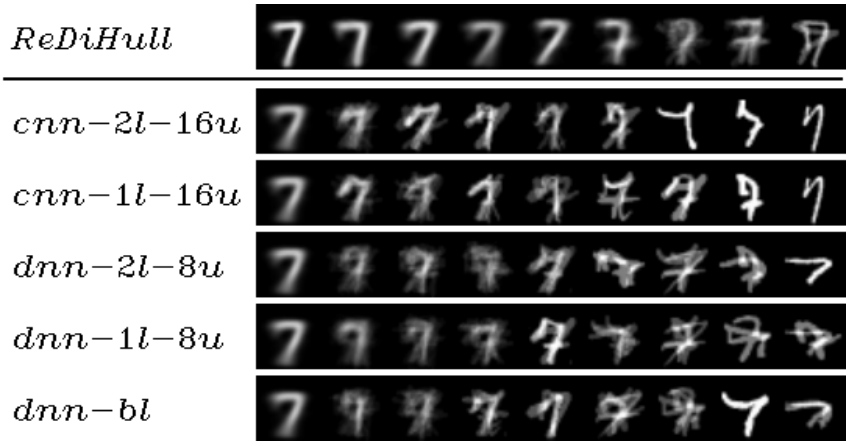


Figure D.28: Increasing difficulty of digit Seven samples (true label) w.r.t digit Four samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

Figure D.29: Increasing difficulty of digit Seven samples (true label) w.r.t digit Nine samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

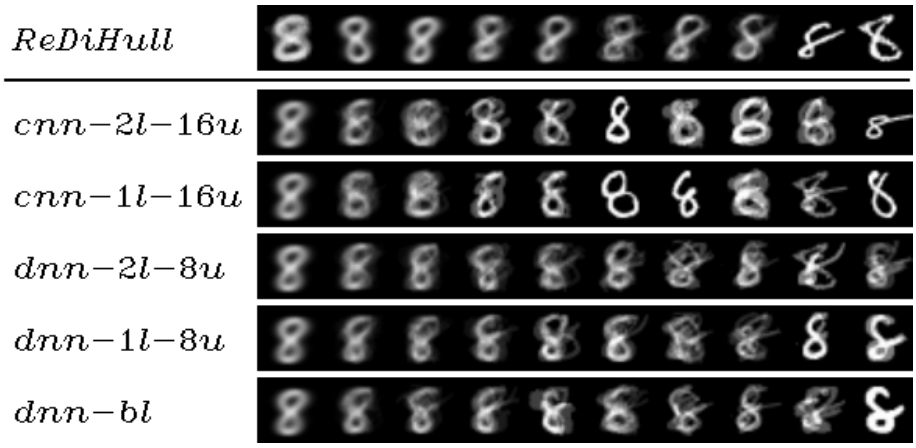


Figure D.30: Increasing difficulty of digit Eight samples (true label) w.r.t digit Six samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

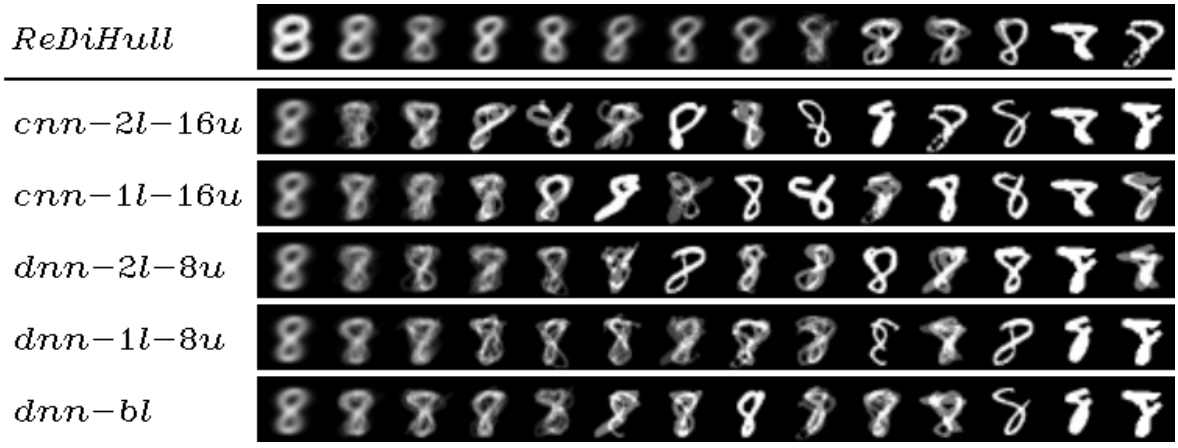


Figure D.31: Increasing difficulty of digit Eight samples (true label) w.r.t digit Seven samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

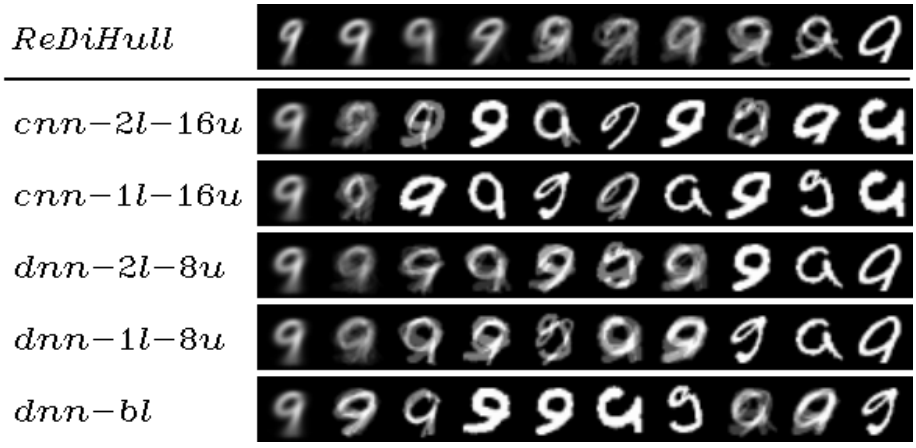


Figure D.32: Increasing difficulty of digit Nine samples (true label) w.r.t digit Zero samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

Figure D.33: Increasing difficulty of digit Nine samples (true label) w.r.t digit One samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

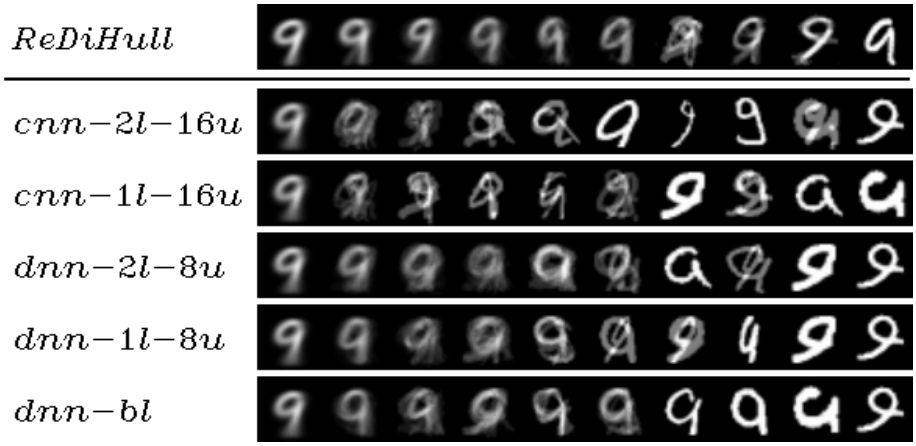


Figure D.34: Increasing difficulty of digit Six samples (true label) w.r.t digit Two samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

Figure D.35: Increasing difficulty of digit Six samples (true label) w.r.t digit Three samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

Figure D.36: Increasing difficulty of digit Nine samples (true label) w.r.t digit Six samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

Figure D.37: Increasing difficulty of digit Nine samples (true label) w.r.t digit Seven samples (predicted label) computed by ReDiHull (top-row) and several neural network architectures.

D.2 ReDiHull-Computed Dataset Difficulty

Predicted Class

	0	1	2	3	4	5	6	7	8	9
0		7.4	3.9	3.1	5.9	2.0	4.8	5.7	3.4	3.4
1	8.9		6.4	5.3	6.4	7.1	7.4	7.0	6.9	7.3
2	2.6	3.7		2.1	2.8	3.1	2.7	3.1	2.5	3.6
3	3.9	4.2	3.0		6.0	2.1	5.4	4.1	2.2	2.9
4	4.6	6.5	3.9	3.8		4.3	4.1	2.6	3.9	0.5
5	2.6	6.1	3.6	1.2	4.2		4.0	3.7	2.5	2.0
6	2.8	5.3	3.0	5.3	2.4	2.7		5.7	5.0	3.5
7	6.8	4.4	3.5	3.3	3.6	5.9	7.6		4.7	2.2
8	3.5	3.0	1.5	0.3	3.3	1.4	3.8	2.1		1.0
9	5.6	5.4	4.6	2.8	0.7	3.7	6.2	1.3	3.3	

Table D.1: MNIST difficulty quantified using KL divergence.

D.3 Neural Network Error Matrices

	Predicted Class									
	0	1	2	3	4	5	6	7	8	9
0		0.0	0.4	0.4	0.1	1.5	0.9	0.3	0.3	0.1
1	0.0		0.9	0.5	0.0	0.3	0.3	0.3	1.9	0.1
2	0.7	1.2		2.7	0.8	0.7	1.3	1.0	3.7	0.6
3	0.3	0.2	2.4		0.1	4.1	0.3	1.2	2.8	1.2
4	0.2	0.3	0.8	0.3		0.3	1.2	0.9	1.3	5.7
5	1.2	0.4	0.9	4.7	1.3		1.7	0.9	5.1	1.1
6	1.1	0.3	1.5	0.2	1.3	2.0		0.2	0.6	0.1
7	0.1	0.9	1.9	1.0	0.8	0.2	0.0		0.5	4.9
8	0.8	1.2	1.7	3.8	1.3	4.6	1.2	0.9		2.0
9	0.7	0.6	0.2	1.2	5.0	1.0	0.1	4.3	1.7	

Table D.2: Average sample prediction error matrix for "dnn-bl" trained and evaluated on the MNIST dataset [%]

	Predicted Class									
	0	1	2	3	4	5	6	7	8	9
0		0.0	0.6	0.6	0.4	1.4	1.2	0.5	0.7	0.2
1	0.0		1.0	0.9	0.1	0.2	0.2	0.2	1.8	0.1
2	1.3	1.3		3.2	1.2	0.5	1.5	1.2	3.0	0.4
3	0.4	0.4	2.9		0.2	3.8	0.2	1.3	2.5	0.8
4	0.3	0.2	0.7	0.2		0.3	1.3	0.7	1.0	6.3
5	1.5	0.3	0.7	6.2	1.1		1.9	0.5	4.1	1.1
6	1.4	0.2	1.4	0.1	1.6	1.9		0.2	0.8	0.1
7	0.3	0.9	2.1	1.4	0.9	0.1	0.0		0.5	5.0
8	1.1	1.3	1.4	3.5	1.2	3.8	1.6	0.7		2.3
9	0.8	0.4	0.2	1.6	5.3	0.7	0.1	3.8	1.6	

Table D.3: Average sample prediction error matrix for "dnn-1l-8u" trained and evaluated on the MNIST dataset [%]

	Predicted Class									
	0	1	2	3	4	5	6	7	8	9
0		0.0	0.8	0.3	0.2	1.3	1.4	0.3	0.4	0.2
1	0.0		0.9	0.8	0.0	0.4	0.3	0.4	1.8	0.1
2	1.4	1.1		2.8	0.9	0.4	1.4	1.4	2.2	0.2
3	0.4	0.4	3.0		0.1	4.9	0.2	1.6	2.5	1.0
4	0.4	0.1	0.8	0.2		0.4	1.5	0.6	0.8	4.9
5	1.9	0.5	0.6	5.1	1.0		2.3	0.4	4.1	1.2
6	1.9	0.3	1.4	0.1	1.6	2.0		0.1	0.7	0.0
7	0.4	0.8	2.1	1.5	0.6	0.1	0.1		0.3	3.6
8	0.6	1.3	1.6	2.6	1.5	3.5	1.5	0.7		2.1
9	0.9	0.4	0.2	1.1	5.7	1.1	0.1	3.1	1.8	

Table D.4: Average sample prediction error matrix for "dnn-2l-8u" trained and evaluated on the MNIST dataset [%]

	Predicted Class									
	0	1	2	3	4	5	6	7	8	9
0		0.0	0.3	0.1	0.0	0.2	0.4	0.2	0.2	0.2
1	0.1		0.2	0.2	0.1	0.0	0.1	0.1	0.1	0.0
2	0.3	0.6		0.6	0.1	0.0	0.2	0.6	0.4	0.2
3	0.1	0.0	0.3		0.0	0.7	0.0	0.2	0.3	0.2
4	0.1	0.1	0.2	0.0		0.0	0.3	0.2	0.1	1.0
5	0.2	0.0	0.1	1.2	0.0		0.6	0.1	0.3	0.1
6	0.7	0.3	0.1	0.0	0.3	0.3		0.0	0.3	0.0
7	0.1	0.5	1.2	0.4	0.2	0.0	0.0		0.3	0.6
8	0.6	0.1	0.6	0.4	0.2	0.3	0.4	0.5		0.7
9	0.3	0.4	0.1	0.4	1.1	0.5	0.0	1.1	0.6	

Table D.5: Average sample prediction error matrix for "cnn-1l-16u" trained and evaluated on the MNIST dataset [%]

	Predicted Class									
	0	1	2	3	4	5	6	7	8	9
0		0.0	0.2	0.0	0.0	0.1	0.3	0.1	0.2	0.1
1	0.0		0.3	0.1	0.2	0.0	0.2	0.1	0.1	0.0
2	0.3	0.4		0.5	0.1	0.0	0.1	0.5	0.4	0.0
3	0.1	0.0	0.2		0.0	0.5	0.0	0.2	0.3	0.2
4	0.1	0.1	0.1	0.0		0.0	0.2	0.2	0.2	1.1
5	0.2	0.0	0.0	0.9	0.0		0.4	0.1	0.2	0.3
6	0.5	0.2	0.0	0.1	0.2	0.4		0.0	0.2	0.0
7	0.0	0.3	1.0	0.3	0.1	0.0	0.0		0.2	0.7
8	0.7	0.1	0.6	0.4	0.2	0.3	0.4	0.3		0.6
9	0.2	0.3	0.1	0.4	0.7	0.4	0.0	0.8	0.3	

Table D.6: Average sample prediction error matrix for "cnn-2l-16u" trained and evaluated on the MNIST dataset [%]