INFORMATION LIES IN THE EYE OF THE BEHOLDER: THE EFFECT OF REPRESENTATIONS ON OBSERVED MUTUAL INFORMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning can be framed as trying to encode the mutual information between input and output while discarding other information in the input. Since the distribution between input and output is unknown, also the true mutual information is. To quantify how difficult it is to learn a task, we calculate a *observed mutual information* score by dividing the estimated mutual information by the entropy of the input. We substantiate this score analytically by showing that the estimated mutual information has an error that increases with the entropy of the data. We show that the observed entropy and mutual information can vary wildly depending on how the data is represented. This suggests that there needs to be a match between how data is represented and how a model encodes it. Experimentally, we analyze image-based input data representations and demonstrate that performance outcomes of extensive network architectures searches are well aligned to the calculated score. Therefore, to ensure better learning outcomes, representations may need to be tailored to both task and model to align with the implicit distribution of the model.

1 INTRODUCTION

Sometimes perspective is everything. While the information content of encoded data may not change when the way it is represented changes, its usefulness can vary dramatically (see Fig. 1). A "useful" representation then is one that makes it easy to extract information of interest. This in turn very much depends on who or which algorithm is extracting the information. Evidently the way data is encoded and how a model "decodes" the information needs to match.

Historically, people have invented a large variety of "data representations" to convey information. An instance of this theme is the heliocentric vs. geocentric view of the solar system. Before the heliocentric viewpoint was widely accepted, scholars had already worked out the movements of the planets (Theodossiou et al., 2002). The main contribution of the new perspective was that now the planetary trajectories were simple ellipses instead of more complicated movements involving loops[1].

In a machine learning context, many have experimented with finding good data representations for specific tasks such as speech recognition (Logan et al., 2000), different color spaces for face recognition (Hsu et al., 2002), for increased robustness in face detection (Podilchuk & Zhang, 1998), and many others. Yet no clear understanding has emerged of why a given representation is more suited to one task but less for another. We cast the problem of choosing the data representation for learning as one of determining the ease of encoding the relationship between input and output which depends both on how the data is represented and which model is supposed to encode it.

Contribution: In this work, we argue that learning a task is about encoding the relationship between input and output. Each model implicitly has a way of encoding information, where some variations in the data are easier to encode than others. Armed with this insight, we empirically evaluate different data representations and record what impact data representations have on learning outcomes and types of networks found by automated network optimization. Most interestingly, we are able

[1]For a clear illustration, see for example
`http://astronomy.nmsu.edu/geas/lectures/lecture11/slide01.html`

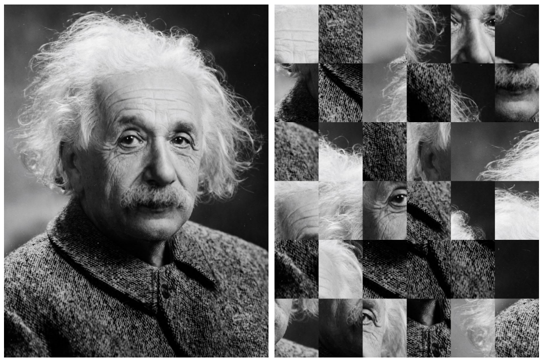

Figure 1: These images contain different representations of the same information. However, one of the two is much easier for us to understand. We posit that, for our nervous system, one of the two images has a higher *observed mutual information* for the task of recognizing the person.

to show that relative learning outcomes can be predicted by an empirical mutual information score, which we coin *Observed Mutual Information* (*OMI*) score.

1.1 Related Work

This work aims to bring us a bit closer to understanding what makes a given learning task easier or harder. While there appears to be little work on this question, a fresh-eyed excursion into what makes an optimization easier or harder (Alpcan et al., 2014) has been ventured before.

Data representations: Data representations have been optimized for a long time. In fact there is a rich theory of linear invertible representations for both finite and infinite dimensional spaces called *Frame Theory* (Christensen et al., 2016). Specific popular examples of frames are Wavelets (Mallat, 1999) and Curvelets (Candes & Donoho, 2000). Empirically tested only on Imagenet, Uber research (Gueguen et al., 2018) showed that using a data representation closer to how JPEG encodes information, may help to create faster residual network architecture with slightly better performance. In a similar spirit in a robotics context, Grassmann and Kahrs (Grassmann et al., 2018) evaluated learning performance on approximating robot dynamics using various common robot dynamics data representations such as Euler angles. What is more common in deep learning is to adapt the network architecture to the task at hand. An intriguing recent example taking this idea a step further are Weight Agnostic Neural Networks (Gaier & Ha, 2019) which have been designed to already "function" on a task even when starting from randomly initialized weights.

Measuring learning difficulty: There appears to be little newer literature on the question, yet Thornton (1995) already posed the question of how to measure how easy or difficult a learning task is in the nineties. Similar to our own findings, they related the difficulty to information theoretic measures called the information gain (mutual information) and the information gain ratio (very similar to our proposed *OMI* value) introduced in the context of decision trees by Quinlan (1986; 2014). Sadly, this interesting line of inquiry does not appear to have received much attention since. Tin Kam Ho & Basu (2002) take a different road by comparing several possible scores to assess the difficulty of classification learning problems such as linear separability and feature efficiency. More commonly, instead of judging task difficulty, there is a vast literature on feature selection (Guyon & Elisseeff, 2003), e.g. judging how suitable a feature is for a given learning problem. Desirable features are reliably selected for a learning task (Meinshausen & Bühlmann, 2010) and ideally are highly predictive of the output variable. Relating the overall difficulty of selecting good features to how difficult a learning task is, has not been established to our understanding.

2 Data Representations and Observed Mutual Information

The objective of learning can be phrased as finding a function that minimizes the uncertainty of the output given the input while discarding as much task irrelevant information as possible. In information theoretic language, this viewpoint was introduced by Tishby et al. (2000) and extended by Achille & Soatto (2018) in the form of the objective of the Information Bottleneck (IB) Lagrangian. Given an input x, a model encoding z, an output y and mutual information $I(\cdot;\cdot)$, the IB-Lagrangian Tishby et al. (2000) aims to minimize the following cost function:

$$\mathcal{L}(p(z|x)) = I(x;z) - \beta I(y;z)$$

The model is supposed to find an encoding z of data x that maximizes the mutual information to output y while also minimizing the mutual information with x. It becomes apparent that the "ideal" learning task is one where all of the information in x is highly relevant to output y. In this case minimizing $I(z;x)$ becomes unnecessary and the learning algorithm can place all of its focus on simply associating x with y. A highly difficult task would be one where $I(x;y)$ is very small in absolute terms or where estimating $I(x;y)$ is difficult for the chosen model.

An attractive option to evaluate how difficult a learning task is thus is measuring its mutual information $I(x;y)$. Empirically estimating the mutual information however leads to errors that have a bias with a bound proportional to the entropy $H(x)$. To adjust, we divide by the entropy term, leading to a score of mutual information which takes into account the uncertainty of the estimate. In the following we bound the deviation of the true mutual information from the estimated mutual information. This estimated mutual information we coin *Observed Mutual Information* since it is the information, dependent on representation and model, that we may ultimately be able to extract from the data. We begin by stating a result of Paninski (2003) bounding the error in entropy estimation.

Lemma 1 (Paninski (2003)). *For a discrete random variable $x \in \mathcal{X}$, with the plug-in estimate $\hat{H}(\cdot)$ on its entropy, based on an i.i.d sample of size m, we have that*

$$|\mathbb{E}[\hat{H}(x) - H(x)]| \leq \log\left(1 + \frac{|\mathcal{X}| - 1}{m}\right) \tag{1}$$

Using the above bound we are able to bound the error in mutual information estimation, in a particular regime which is relevant for several applications of interest, such as object detection.

Definition 1 (Distillation regime). *In the* distillation regime *we assume that:*

i The samples x have very high entropy $H(x)$.

ii The entropy of y is small with respect to the entropy of x.

iii The number of samples m of x that we have is small compared to $H(x)$.

Example 1. *Typical object detection tasks are in the distillation regime. The entropy of images is high (property 1), while labels are compactly represented (property 2). Furthermore, the number of image samples is small compared to all possible samples (property 3).*

Lemma 2 (Mutual information bias). *For discrete random variables $x \in \mathcal{X}$, $y \in \mathcal{Y}$, with the plug-in estimates $\hat{H}(\cdot)$ on their entropy, based on an i.i.d sample of size m, we bound the deviation of the estimated mutual information to the true mutual information in the distillation regime and conclude*

$$|\mathbb{E}[\hat{I}(x;y) - I(x;y)]| \leq 2(H(x) + \epsilon - \log(m)))$$

Proof. We start with a similar left-hand side as Ineq. 1 and expand the mutual information with entropy terms:

$$\begin{aligned}
|\mathbb{E}[\hat{I}(x;y) - I(x;y)]| &= |\mathbb{E}[\hat{H}(x) + \hat{H}(y) - \hat{H}(x,y) - H(x) - H(y) + H(x,y)]| \\
&= |\mathbb{E}[\hat{H}(x) - H(x) + \hat{H}(y) - H(y) + H(x,y) - \hat{H}(x,y)]| \\
&\leq |\mathbb{E}[\hat{H}(x) - H(x)]| + |\mathbb{E}[\hat{H}(y) - H(y)]| + |\mathbb{E}[H(x,y) - \hat{H}(x,y)]|
\end{aligned}$$

Using the stated assumptions and applying Ineq. 1 we arrive at the following bound using the asymptotic equipartition property (AEP) $|\mathcal{X}| \leq 2^{H(x)+\epsilon}$ $(Cover\&Thomas,\ 2012)$:

$$\begin{aligned}
|\mathbb{E}[\hat{I}(x;y) - I(x;y)]| &\leq \log\left(1 + \frac{|\mathcal{X}| - 1}{m}\right) + \log\left(1 + \frac{|\mathcal{Y}| - 1}{m}\right) + \log\left(1 + \frac{|\mathcal{X},\mathcal{Y}| - 1}{m}\right) \\
&\overset{\text{AEP}}{\leq} \log\left(1 + \frac{2^{H(x)+\epsilon} - 1}{m}\right) + \log\left(1 + \frac{2^{H(y)+\epsilon} - 1}{m}\right) + \log\left(1 + \frac{2^{H(x,y)+\epsilon} - 1}{m}\right) \\
&\overset{\text{(i, ii)}}{\leq} 2\log\left(1 + \frac{2^{H(x)+\epsilon} - 1}{m}\right) \overset{\text{(iii)}}{=} 2(H(x) + \epsilon - \log(m)))
\end{aligned}$$

□

A similar relationship can be extracted from the information bottleneck bound (Shamir et al., 2010).

Theorem 3 (Theorem 4, Shamir et al. (2010))**.** *For any probability distribution $p(x, y)$, with a probability of at least $1 - \delta$ over the draw of the sample of size m from $p(x, y)$, we have that for all bottleneck variables z,*

$$\begin{aligned} |I(y; z) - \hat{I}(y; z)| &\leq B(y)B(x, z) \\ B(y) &:= \sqrt{\frac{C \log(|\mathcal{Y}|/\delta)}{m}} \\ B(x, z) &:= \Big(C_1 \log(m)\sqrt{|\mathcal{Z}| I(x; z)} + C_2 |\mathcal{Z}|^{3/4} I(x; z)^{1/4} + C_3 \hat{I}(x; z)\Big) \end{aligned}$$

Again we consider the case where $|\mathcal{Y}|$ is small, $|\mathcal{X}|$ is large, and where the learned bottleneck variable z captures the output variable y exactly (hence $|\mathcal{Z}|$ is small). Then $I(x; z) = H(x) - \epsilon$. For large terms $H(x)$ the above bound becomes dominated by $\hat{H}(x)$ s.t. (for fixed $|\mathcal{Y}|, m, \delta$)

$$|I(y; z) - \hat{I}(y; z)| \leq \mathcal{O}(\hat{H}(x))$$

When it comes to mutual information, we care about a high mutual information between input and output, but also about a reliable estimate of this mutual information. From the above calculations, we distill that the estimated entropy $\hat{H}(x)$ has a decisive effect on the uncertainty of the achievable mutual information between bottleneck variable z and output y. As a metaphor imagine the task of predicting the birthrate in Mongolia from the movements of the stock market for a given month. Most certainly one will be able to correlate the two. This is a common occurrence called *spurious correlation* (Fan et al., 2012) making us fools of randomness (Taleb, 2005). Hence we arrive at a score for mutual information that captures both the magnitude of the estimated mutual information and an estimate of the uncertainty of this estimate.

Definition 2 (OMI)**.** *Given random variables x and y, empirical mutual information $\hat{I}(x; y)$, and empirical entropy $\hat{H}(x)$, then the* observed mutual information score *(*OMI*) is defined as*

$$\text{OMI}(x, y) := \frac{\hat{I}(x; y)}{\hat{H}(x)},$$

A similar thought process of introducing the same mutual information score has long been used for selecting features in decision trees (Quinlan, 1986). While we base our score on a whole dataset of (x, y)-pairs, the *information gain ratio* (IGR) for decision trees are feature-level criteria. Having defined our focus on mutual information and its associated *OMI* values, we turn our attention to the effect data representations may have on the learning process. We begin by defining what we mean when we talk about a *data representation*.

Definition 3. *A data representation $r \in \mathcal{R}$ is the output of a left-invertible mapping $m(\cdot) : \mathcal{X} \to \mathcal{R}$ applied to the "original" data $x \in \mathcal{X}$.*

$$r = m(x), \forall x \in \mathcal{X}, \quad x = m^{-1}(r), \forall r \in \mathcal{R}$$

Therefore, all data representations are in a sense "equivalent". Since they are invertible they share the same information content $I(x; y) = I(m(x); y))$. Yet this is only half true. Clearly, how data is represented does make a difference. As a "worst case", an encrypted version of a dataset is unlikely to be useful for a learning task. In a perfect setting, a dataset could be represented in a way that directly maps to the desired output while keeping around additional "bits" needed to reconstruct the input. In such a case, learning would only require disregarding the extra dimensions.

To understand what impact a data representation may have we will employ the idea of expected coding length $\mathbb{E}[l(x)]$ and focus on what happens when we choose the "wrong code". From Information Theory (Cover & Thomas, 2012), we learn that the most efficient encoding we can possibly find is lower bounded by the entropy of the distribution we are trying to compress. In this case, we assume that we have a candidate distribution $q(x)$ that we are trying to fit to the true distribution $p(x)$. The expected coding length of our candidate distribution can then never be smaller than the entropy of the true distribution (Cover & Thomas, 2012): $\mathbb{E}[l(x)] = H_q(x) \geq H_p(x)$.

Theorem 4 ((Wrong code) Theorem 5.4.3 (Cover & Thomas, 2012))**.** *The expected length under $p(x)$ of the code assignment $l(x) = \lceil \log \frac{1}{q(x)} \rceil$ satisfies*

$$H(p) + D(p||q) \leq \mathbb{E}_p[l(X)] = \hat{H}(x) < H(p) + D(p||q) + 1 \tag{2}$$

In the following we will assume that any function family $\mathcal{F}$ has an associated candidate distribution $q(\cdot)$ through which it measures how uncertain a variable is, e.g. linear regression uses a normal distribution and assesses uncertainty via the determinant of the covariance matrix. The difficulty with assuming a candidate distribution is that it data may not follow the same distribution. Given such a mismatch, the model will overestimate the entropy of the distribution as shown in theorem 2.

Theorem 5 (Representation-Model-Alignment)**.** *Assuming a candidate distribution $q(\cdot)$ and representations r_1, r_2 with $D(p(r_1)||q(r_1)) > D(p(r_2)||q(r_2)) + 1$ we have that*

$$\hat{H}_q(r_1) > \hat{H}_q(r_2)$$

Proof. From theorem 2 we know that $H(p(r)) + D(p(r)||q(r)) \leq \hat{H}(q(r)) < H(p(r)) + D(p(r)||q(r)) + 1$. Thus $\hat{H}_q(r_1) - \hat{H}_q(r_2) > H(p(r_1)) + D(p(r_1)||q(r_1)) - H(p(r_2)) - D(p(r_2)||q(r_2)) - 1 = D(p(r_1)||q(r_1)) - D(p(r_2)||q(r_2)) - 1 > 0$ □

A consequence of the above theorem is that the representation of the data distribution influences the observed entropy by changing the alignment of the true distribution to the assumed candidate distribution of the model. Critically for real-world situations, the wrong code theorem invalidates the assumption that the estimated mutual information does not change when an invertible transformation is applied. The true mutual information does indeed not change, yet this would be purely anecdotal if one were to encrypt the data and tried to learn from it. Changing data representations (invertible) can better align true task and assumed model distribution in the sense of minimizing relative entropy between candidate $q(\cdot)$ and true distribution $p(\cdot)$. This then has direct influence on the *observed mutual information* and the *OMI* score. The closer representation r aligns the true distribution to the model candidate distribution, the smaller the data entropies $\hat{H}(r)$ and $\hat{H}(y|r)$ will be. In this sense there are thus better and worse data representations given a model and learning task.

3 EXPERIMENTS

The theoretical findings we presented in the previous sections have to be verified in the real world. We therefore conducted a wide ranging empirical study on their relevance over a number of different datasets and network architectures. In total we evaluated over 8000 networks of varying sizes. We provide the full code and data required to replicate our experiments at the following URL:
`https://drive.google.com/open?id=1D8wICzJVPJRUWB9y5WgceslXZfurY34g`

3.1 DATASETS

To have a chance that our findings are applicable beyond the scope of this publication we chose a diverse set of four datasets that capture different vision tasks. We chose two datasets for classification (KDEF and Groceries) and two for regression (Lane Following and Drone Racing). Sample images for each of the datasets can be found in the appendix.

Table 1: Dataset overview

Dataset	input	output	split (train/val/test)	loss
Drone	$346 \times 260 \times 1$	3×1	10806/1403/1339	L1
Groceries	$120 \times 120 \times 3$	25×1	6409/1018/1016	NLL
KDEF	$91 \times 127 \times 3$	7×1	3931/470/497	NLL
LF	$160 \times 120 \times 3$	1×1	10850/1386/1364	L1

Lane Following (LF): We generated this dataset using the Duckietown Gym (Chevalier-Boisvert et al., 2018), which is a simulation environment used for the AI Driving Olympics placed in Duckietown (Paull et al., 2017; Zilly et al., 2019)[2]. It is the only simulated dataset we used in our experiments. Data is generated using a pure pursuit lane following algorithm, that returns the desired angular velocity. Longitudinal velocity is assumed to be constant. The learned model has to predict the angular velocity returned by the expert based on the image of the lane. Both train and test sets include domain randomization. To reduce the cost of training we downsampled each of the images to a third of their original size.

KDEF: This dataset is based on an emotion recognition dataset by Lundqvist et al. (1998). Each of the images shows male and female actors expressing one of seven emotions. Images are captured from a number of different fixed viewpoints and centered on the face. To add more diversity to the data we added small color and brightness perturbations and a random crop. Moreover, since the dataset provides few samples, we downsampled each of the images to a sixth of their original size.

Drone Racing: This dataset is based on the Drone Racing dataset by Delmerico et al. (2019). We use the mDAVIS data from subsets 3, 5, 6, 9, and 10. While the original dataset provides the full pose (all six DOF), we train our feedforward networks to recover only the rotational DOF (roll, pitch, and yaw) from grayscale images. We matched the IMU data, which is sampled at 1000Hz to the timestamp for each grayscale image captured at 50Hz using linear interpolation. Since the images do not have multiple color channels we did not investigate a separate YCbCr or PREC representation for this dataset.

Groceries: We use the Freiburg Groceries Dataset (Jund et al., 2016) and their original "test0/train0" split. Our only modifications are that we reserve a random subset of the test data for the evaluation of our hyperparameter optimization and that we reduce the size of the images from 256×256 to 120×120 pixels. Each of the images has to be classified into one of 25 categories.

3.2 *OMI* ESTIMATION

To estimate the *OMI* values, we calculated the individual entropies $\hat{H}(x), \hat{H}(y)$, and $\hat{H}(x, y)$. To compute $\hat{H}(x)$ and $\hat{H}(x, y)$, we assume a multivariate Gaussian distribution on the variables. The entropy is then computed as $\hat{H}(x) = \frac{1}{2}\log(\det(2\pi e\Sigma))$, where Σ denotes the covariance matrix of x. To calculate this we apply an SVD decomposition and use the sum of log singular values to estimate the entropy. To avoid severe underestimation (entropy estimation is biased to underestimate (Paninski, 2003)), entropy values are lower bounded by 2. Entropy values for y are calculated the same way for the regression task. For the classification task, we assume a multinoulli distribution and estimate the entropy of y accordingly.

3.3 BAYESIAN HYPERPARAMETER OPTIMIZATION

Manually tuning the hyperparameters for neural networks is both time consuming and may introduce unwanted bias into experiments. There is a wide range of automated methods available to mitigate these flaws (Bergstra & Bengio, 2012; Hutter et al., 2015). We utilize Bayesian optimization to find the set of optimal hyperparameters for each network. Since the initial weights of our networks are sampled from a uniform random distribution we can expect the performance to fluctuate between runs. Due to its probabilistic approach Bayesian optimization can account for this uncertainty (Shahriari et al., 2015). Moreover, it can optimize categorical and continuous dimensions concurrently. The dimensions and their constraints were chosen to be identical for each representation but were adapted to each dataset. For an in-depth introduction to Bayesian optimization we refer to Snoek et al. (2012). More information on our particular implementation of Bayesian optimization can be found in the appendix.

3.4 NETWORK ARCHITECTURES

We investigated three basic architectures. Our optimization was constrained to the same domain for all representations of a dataset for each of the network architectures. The full list of constraints for each network and dataset can be found in the code accompanying this paper. The initial learning rate was optimized for all architectures.

[2]`https://www.duckietown.org/`

Convolutional Networks: We use a variable number of convolutional layers (LeCun et al., 1998), with or without maxpooling layers between them, followed by a variable number of fully connected layers. Moreover, kernel sizes and number of filters are also parametrized.

Dense Neural Networks: These networks consist of blocks of variably sized fully connected layers. We optimize the activation function after each layer as a categorical variable.

ResNets: The Residual Neural Networks or ResNets (He et al., 2015) are made up of a variable number of residual layers. Each of the layers contains a variable number of convolutions which themselves are fully parametrized.

3.5 Representations

While one could select an arbitrary number of representations for images, we limit ourselves to five which have previously been used in image processing. Our focus is not on findings the best representations but to show how sensitive learning processes are to the representation of input data. Sample images for each of the representations can be found in the appendix.

RGB: RGB is likely the representation we use most in our everyday life. Almost all modern displays and cameras capture or display data as an overlay of red, green and blue color channels. For simplicity we refer to the grayscale images of the Drone Racing dataset as being “RGB”.

YCbCr: The YCbCr representation is used in a number of different image storage formats such as JPEG and MPEG (David S. Taubman, 2013). It represents the image as a combination of a luminance and two chrominance channels. It is useful for compression because the human eye is much less sensitive to changes in chrominance than it is to changes in luminance.

PREC: This representation partially decorrelates the color channels of the image based on previous work on preconditioning convolutional neural networks (Liu et al., 2018). For an image $x \in \mathbb{R}^{m \times n \times c}$ with c channels we first calculate the expected value of the covariance between the channels for each image in the dataset: $\Sigma = \mathbb{E}_{\mathcal{D}} \mathbb{E}_{x_{ij}}[x_{ij} x_{ij}^T] \in \mathbb{R}^{c \times c}$, where $x_{ij} \in \mathbb{R}^c$ is the channel vector at pixel (i, j) of image $x \in \mathcal{D}$. We then solve the eigenvalue problem for Σ obtaining real eigenvalues Λ and V containing the eigenvectors. A small ϵ is added for numerical stability. u is stored in memory and consecutively applied to each image in the dataset. We get $U = \text{diag}(\Lambda + \epsilon I)^{-\frac{1}{2}} V$ which yields $x_{\text{prec}} = x_{\text{rgb}} * U$.

DCT: The 2D type II discrete cosine transform (DCT) is a frequency-based representation. Low frequency coefficients are located in the top left corner of the representation and horizontal/vertical frequencies increase towards the right or down, respectively. This representation applies the DCT transform to each of the channels separately. DCT has been used extensively for face detection (Hafed & Levine, 2001; Pan et al., 2000) and all its coefficients bar one are invariant to uniform changes in brightness (Er et al., 2005).

Block DCT: Unlike for the DCT representation we apply the discrete cosine transform not to the whole image but to 8×8 non-overlapping patches of each of the channels. This exact type of DCT is widely used in JPEG compression by quantizing the coefficients of the DCT and applying Huffman encoding (David S. Taubman, 2013).

3.6 Training

Each network was trained using an Adam optimizer (Kingma & Ba, 2014). Training was terminated when there were more than 7 previous epochs without a decrease in loss on the validation set or after 30 epochs were reached.

4 Discussion and Results

After evaluating a total of 5753 networks, 3702 of which finished training, we have verified our intuition that representations are important. We see a fairly consistent pattern over all datasets of RGB and YCbCr being the best representations, followed by PREC and blockwise DCT, while DCT falls short (see Fig. 3).

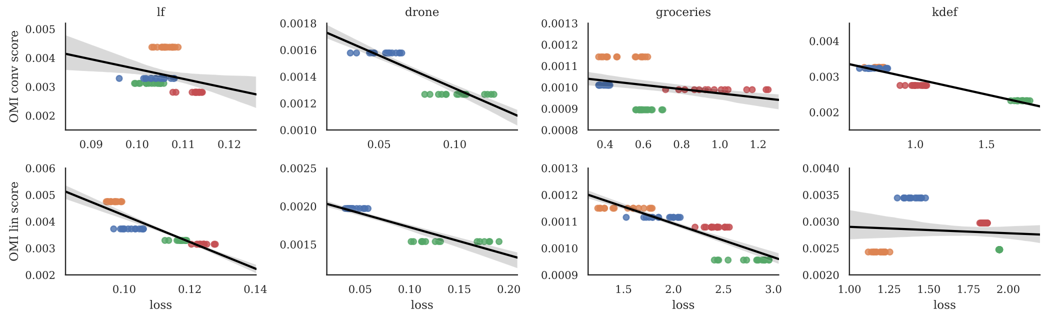

Figure 2: Performance of the RGB, YCbCr, Block DCT, PREC (blue, orange, green, red) on the datasets. From left to right: KDEF, Groceries, Lane Following, Drone Racing. DCT was omitted because of its very small *OMI* scores.

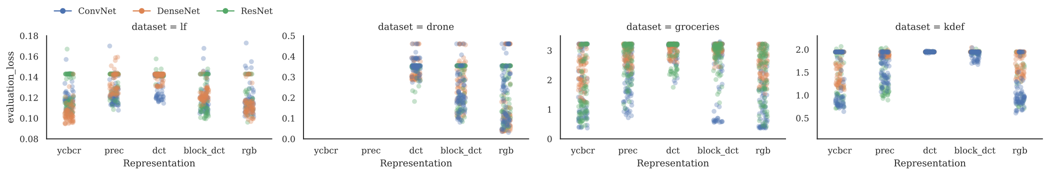

Figure 3: Performance of representations on the datasets. From left to right: KDEF, Groceries, Lane Following, Drone Racing

Moreover, we observe the great importance of hyperparameters in the spread of results for each network architecture. Had we chosen to hand-tune our parameters and accidentally picked a very poor performing network for the RGB representation we could have easily come to the conclusion that DCT achieves better results on all datasets. As we predicted, the performance of a representations also depends greatly on the network architecture. This is especially visible for the lane following dataset. We can see that, surprisingly, dense networks are among the best for both RGB and YCbCr, while they fall far behind on all other representations.

The *OMI* scores we proposed show strong correlation with the results we obtained from architecture search. They can even predict the comparatively small differences between the other representations with reasonable accuracy (see Fig. 2). It is unclear why the prediction fails for some of the datasets, especially the linear score on the KDEF dataset. Overall, we observe the significant correlated effect representation has on estimated entropy, *OMI* scores as well as performance.

5 Conclusion

This work started by trying to pave a way to the exciting task of determining the difficulty of a learning task. To achieve this we introduced *OMI*, as a score that takes both estimated mutual information and possible variance of such an estimate into account. If the score proves itself also in future works, it may serve as a useful tool for automatic representation or architecture search. As outlined, this will depend to a great deal on how well we understand the candidate distributions of network architectures and representations that are currently in use. Similarly, it will be beneficial to study further techniques on removing the bias in the estimation of mutual information and entropy. We have shown that in many problems of interests the naive computation of mutual information is biased and representation dependent; our OMI score partially remove this bias. This score also now provides a criterion to evaluate different representations in a principled way.

REFERENCES

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

Tansu Alpcan, Tom Everitt, and Marcus Hutter. Can we measure the difficulty of an optimization problem? In *2014 IEEE Information Theory Workshop (ITW 2014)*, pp. 356–360. IEEE, 2014.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

Eric Brochu, Matthew D. Hoffman, and Nando de Freitas. Hedging strategies for bayesian optimization. *CoRR*, abs/1009.5419, 2010.

Emmanuel J Candes and David L Donoho. Curvelets: A surprisingly effective nonadaptive representation for objects with edges. Technical report, Stanford Univ Ca Dept of Statistics, 2000.

Clément Chevalier and David Ginsbourger. Fast Computation of the Multi-points Expected Improvement with Applications in Batch Selection. working paper or preprint, October 2012.

Maxime Chevalier-Boisvert, Florian Golemo, Yanjun Cao, Bhairav Mehta, and Liam Paull. Duckietown environments for openai gym. `https://github.com/duckietown/gym-duckietown`, 2018.

Ole Christensen et al. *An introduction to frames and Riesz bases*. Springer, 2016.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

Michael W. Marcellin David S. Taubman. *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer, Boston, MA, 2013. ISBN 978-1-4615-0799-4.

Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. Are we ready for autonomous drone racing? the uzhfpv drone racing dataset. In *IEEE Int. Conf. Robot. Autom.(ICRA)*, 2019.

Meng Joo Er, W. Chen, and Shiqian Wu. High-speed face recognition based on discrete cosine transform and rbf neural networks. *IEEE Transactions on Neural Networks*, 16(3):679–691, May 2005. ISSN 1045-9227. doi: 10.1109/TNN.2005.844909.

Jianqing Fan, Shaojun Guo, and Ning Hao. Variance estimation using refitted cross-validation in ultrahigh dimensional regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(1):37–65, 2012.

Adam Gaier and David Ha. Weight agnostic neural networks. *arXiv preprint arXiv:1906.04358*, 2019.

Reinhard M Grassmann, Jessica Burgner-Kahrs, Mohamed Taha Chikhaoui, Sven Lilge, Simon Kleinschmidt, Jessica Burgner-Kahrs, David Black, Sven Lilge, Carolin Fellmann, Anke V Reinschluessel, et al. On the merits of joint space and orientation representations in learning the forward kinematics in se (3). *Annals of Biomedical Engineering*, 46(10):1498–1510, 2018.

Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu, and Jason Yosinski. Faster neural networks straight from jpeg. In *Advances in Neural Information Processing Systems*, pp. 3933–3944, 2018.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

Ziad M. Hafed and Martin D. Levine. Face recognition using the discrete cosine transform. *International Journal of Computer Vision*, 43(3):167–188, Jul 2001. ISSN 1573-1405. doi: 10.1023/A:1011183429707.

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

Rein-Lien Hsu, Mohamed Abdel-Mottaleb, and Anil K Jain. Face detection in color images. *IEEE transactions on pattern analysis and machine intelligence*, 24(5):696–706, 2002.

Frank Hutter, Jörg Lücke, and Lars Schmidt-Thieme. Beyond manual tuning of hyperparameters. *KI - Künstliche Intelligenz*, 29(4):329–337, Nov 2015. ISSN 1610-1987. doi: 10.1007/s13218-015-0381-0.

Philipp Jund, Nichola Abdo, Andreas Eitel, and Wolfram Burgard. The freiburg groceries dataset. *CoRR*, abs/1611.05799, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 2262–2268. IEEE, 2018.

Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *ISMIR*, volume 270, pp. 1–11, 2000.

E. Lundqvist, Flykt, and Öhman. The karolinska directed emotional faces (kdef). *CD ROM from Department of Clinical Neuroscience, Psychology section, ISBN 91-630-7164-9*, 1998.

Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

Nicolai Meinshausen and Peter Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

Zhengjun Pan, A. G. Rust, and H. Bolouri. Image redundancy reduction for neural network classification using discrete cosine transforms. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pp. 149–154 vol.3, July 2000. doi: 10.1109/IJCNN.2000.861296.

Liam Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003.

Liam Paull, Jacopo Tani, Heejin Ahn, Javier Alonso-Mora, Luca Carlone, Michal Cap, Yu Fan Chen, Changhyun Choi, Jeff Dusek, Yajun Fang, et al. Duckietown: an open, inexpensive and flexible platform for autonomy education and research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1497–1504. IEEE, 2017.

Christine Irene Podilchuk and Xiaoyu Zhang. Face recognition using dct-based feature vectors, September 1 1998. US Patent 5,802,208.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, Mar 1986. ISSN 1573-0565. doi: 10.1007/BF00116251.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Ohad Shamir, Sivan Sabato, and Naftali Tishby. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30):2696–2711, 2010.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 2951–2959. Curran Associates, Inc., 2012.

Nassim Taleb. *Fooled by randomness: The hidden role of chance in life and in the markets*, volume 1. Random House Incorporated, 2005.

E Theodossiou, E Danezis, VN Manimanis, and E-M Kalyva. From pythagoreans to kepler: the dispute between the geocentric and the heliocentric systems. *Journal of Astronomical History and Heritage*, 5:89–98, 2002.

Chris Thornton. Measuring the difficulty of specific learning problems. *Connection Science*, 7(1): 81–92, 1995.

Tin Kam Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, March 2002. doi: 10.1109/34.990132.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Julian Zilly, Jacopo Tani, Breandan Considine, Bhairav Mehta, Andrea F Daniele, Manfred Diaz, Gianmarco Bernasconi, Claudio Ruch, Jan Hakenberg, Florian Golemo, et al. The ai driving olympics at neurips 2018. *arXiv preprint arXiv:1903.02503*, 2019.

APPENDIX

RESULTS TABLE AND NOTES ON NUMERICAL STABILITY

Table 2: Estimated linear (lin) and convolutional (conv) mutual information (*MI*) and *OMI* values for all representations and datasets.

		linear			convolutional		
Dataset	Representation	HX	*MI*	*OMI*	HX	*MI*	*OMI*
LF	Block DCT	145.728	0.48	0.003	154.426	0.481	0.003
	PREC	152.586	0.48	0.003	171.6	0.482	0.003
	DCT	-65.737	-1.042	0.016	-49.506	-0.939	0.019
	RGB	127.45	0.474	0.004	146.184	0.48	0.003
	YCbCr	98.935	0.469	0.005	108.237	0.473	0.004
Groceries	Block DCT	173.03	0.165	0.001	185.891	0.166	0.001
	PREC	147.484	0.159	0.001	165.244	0.163	0.001
	DCT	-68.498	-22.308	0.326	-57.204	-19.511	0.341
	RGB	139.384	0.155	0.001	158.745	0.16	0.001
	YCbCr	114.42	0.132	0.001	128.81	0.147	0.001
Drone	Block DCT	187.756	0.289	0.002	228.591	0.29	0.001
	DCT	-78.147	-5.093	0.065	-29.227	-2.692	0.092
	RGB	142.092	0.28	0.002	182.263	0.287	0.002
KDEF	Block DCT	164.168	0.406	0.002	175.626	0.408	0.002
	PREC	134.03	0.398	0.003	145.822	0.401	0.003
	DCT	-96.852	-14.767	0.152	-79.78	-12.851	0.161
	RGB	96.9	0.333	0.003	116.146	0.375	0.003
	YCbCr	69.619	0.169	0.002	84.362	0.275	0.003

To narrate Tab. 2 we note that that estimating small entropy values is very error prone. When assuming a normal distribution, the entropy is calculated via the sum of the logarithm of eigenvalues of the covariance matrix of the data. The conditioning of the logarithm however gets worse, the closer its argument is to zero. Eigenvalues close enough to zero are thus likely to carry a significant error when used for entropy computation. This all is to say that while purely mathematically it is impossible to have negative mutual information values, numerically such things are bound to happen when dealing with small eigenvalues as is prominent with the DCT representation.

BAYESIAN OPTIMIZATION SUPPLEMENTARY INFORMATION

While there have been some theoretical proposals that would allow Bayesian optimization to be run in parallel asynchronously (Snoek et al., 2012), we restrict ourselves to a simple form of batch parallelization evaluating $n = 6$ points in parallel. We acquire the points by using the minimum constant liar strategy (Chevalier & Ginsbourger, 2012). The base estimator is first used after 10 points have been evaluated. Our acquisition function is chosen at each iteration from a portfolio of acquisition functions using a GP-Hedge strategy, as proposed in (Brochu et al., 2010). We optimize the acquisition function by sampling it at n points for categorical dimensions and 20 iterations of L-BFGS (Liu & Nocedal, 1989) for continuous dimensions.

Since the optimizer has no information about the geometric properties of the network or if the network can fit in the systems memory, some of the generated networks cannot be trained. Two common modes of failure were too many pooling layers (resulting in a layer size smaller than the kernel of subsequent layers) and running out of memory, which was especially prevalent for dense networks. In our experiments we observed that roughly 35% of all networks did not complete training. To stop the Bayesian optimizer from evaluating these points again we reported a large artificially generated loss to the optimizer at the point where the network crashed. The magnitude of this loss was chosen manually for each dataset to be roughly one order of magnitude larger than the expected loss. The influence of this practice will have to be investigated in future research.

REPRESENTATION SAMPLES

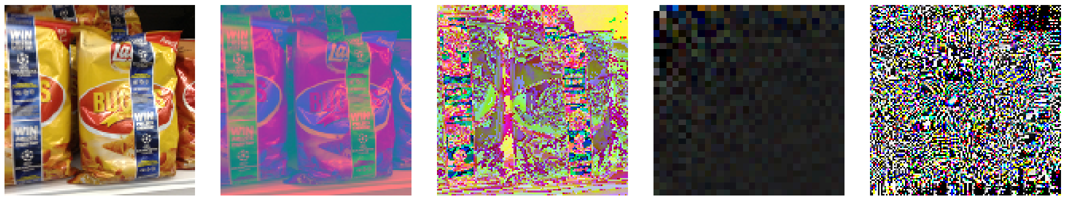

Figure 4: Image from the Groceries dataset in various representations. From left to right: RGB, YCBCR, PREC, DCT, blockwise DCT (cropped to show relevant coefficients, contrast boosted).

DATASET SAMPLES

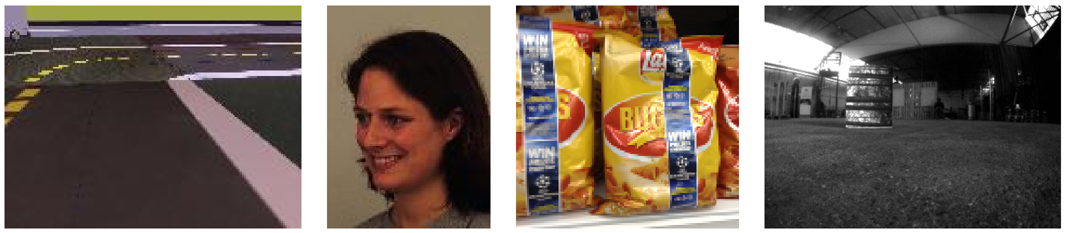

Figure 5: Samples from each of the datasets. From left to right: Lane Following, KDEF, Groceries and Drone Racing.