INTRODUCING SAMPLE ROBUSTNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Choosing the right training set and model for a pre-defined task is one of the critical competencies in machine learning. Investigating what features of a dataset and its underlying distribution a model decodes and learns may enlighten the mysterious "black box" and guide us to a deeper and more profound understanding of the ongoing processes. More so it will help to improve the quality of a model depending directly on the samples it is trained with. In this work, we introduce the concept of sample robustness, which is based on a point-wise Lipschitz constant of the label map and which can, for a particular element in a dataset, be regarded as an intrinsic, only on the respective set dependent feature. We empirically investigate the robustness distribution underlying datasets covering both the classification- and regression setup and use (non-)deterministic models to analyse the effect of training on subsets of least- and most robust data. In some cases, we can boost model performance by choosing specifically tailored training(sub)sets and hyperparameters depending on the robustness distribution of the test(sub)sets.

1 INTRODUCTION:

In the age of automated machine learning, we shift our focus evermore towards regarding different meta-hyperparameters and building blocks such as the model-type or training- and validation budget as variables of a to-be-minimised loss function in the most abstract sense. For training sets, however, the mere number of samples often determines how well suited it is perceived for a particular task. The motivation of this work is to introduce a concept allowing for the use of datasets as variables of such a generalised loss function as well.

Imagine, for example, patients who share almost identical medical records, but react differently to some prescribed treatment. They may pose a challenge (not only to doctors but also) to machine learning models trained on a specific training set $\mathbf{x}$ as the relationship between medical features and the right personal treatment may be very sensitive towards small input variations, i.e. not very robust towards perturbations. In this paper we are, to the best of our knowledge, the first to introduce a measure for the robustness of data and show that depending on the robustness distribution underlying a test(sub)set there may be a training(sub)set $\tilde{\mathbf{x}} \subsetneq \mathbf{x}$ and more so a particular choice of hyperparameters which may help the model to decode such a hidden relationship more precisely.

Let us motivate the concept of sample robustness first on a high level. When collecting and preprocessing a dataset for a pre-defined task, we identify certain features and expressive samples such that the model (after a reasonable amount of training) may be able to abstract and generalize from these finite points to the whole space of possible inputs. Assume we have a certain rectangle-shaped data-distribution in a circle-shaped feature space and a dataset labelled according to two distinct ground truth maps $y^* \in \{\times, *\}$ and $z^* \in \{\circ, \bullet\}$ (comp. **Figure 1**). Imagine classifying images of horses and cats (assuming ground truth y^*) or classifying images of animal-human pairs (assuming ground truth z^*). Evidently, the distance between differently labelled samples depends on the ground truth map labelling them.

For every sample in a dataset, the intrinsic (i.e. only on the dataset, ground truth map and the respective metrics dependent) information of closeness to a differently labelled sample can be considered a feature itself. For regression tasks (and label distributions which are not necessarily categorical) one may also include the distance of the corresponding labels so that by taking the quotient of these two and maximising it over the dataset, i.e. calculating a point-wise Lipschitz constant of the label map, one can measure how sensitive the original sample is to perturbations causing a change of its

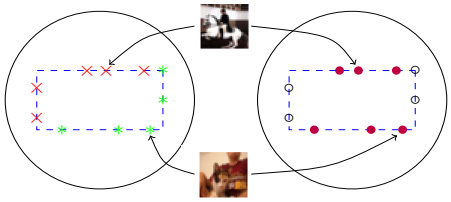

Figure 1: Images from the CIFAR-10 dataset labelled "horse" and "cat", respectively.

label. Precisely this is the non-formal basis of the concept we propose in this work, namely sample robustness, which we investigate for different datasets covering both the classification- and regression set up and which we furthermore use to analyse the effects of training on least- and most robust data(sub)sets on general performance for different (non-)deterministic models. We hypothesise that the latter, which mathematically are extensions of the label map from the (metric) subspace formed by a training set to the whole feature space, adapt to the respective robustness distribution and that the optimal choice of hyperparameter depends on the robustness distribution of both the training- and test-set.

1.1 Outline

After citing and discussing related work concerned with decision boundaries, model robustness and Lipschitz calculus in section 2, we introduce the (mathematical) framework and the main concept of sample robustness in section 3. Section 4 is completely devoted to the evaluation using different datasets (CIFAR-10, Online News Popularity Dataset) and models (Convolutional Neural Networks, K-Nearest Neighbour, Random Forest). Section 5 finally concludes the findings and discusses other research approaches. Appendix sections are referenced using letters **A** - **F**.

2 Related Work:

Analysing the data-distribution before training yields a way to investigate (and boost) model performance from an earlier stage as is done by unsupervised pre-training (Erhan et al., 2010)). Many algorithms stemming from the unsupervised setting (Chigirev & Bialek, 2003), (Cayton, 2005) are devoted to extract information about the so-called data-manifold (Fefferman et al., 2013), (Bernstein & Kuleshov, 2014). Decoding the latent features which determine the data-distribution (Bengio et al., 2013) provides valuable insight and helps to understand the decision boundaries which a model learns throughout the training phase, how they evolve and how they are related to the data-manifold so as to provide a view into the "black box" transforming inputs to outputs (Fawzi et al., 2016), (Liu et al., 2017), (Biggio et al., 2013). In this work, we want to use intrinsic information of distance between samples in feature space and relate it to the distance of the corresponding labels to introduce a new dataset-dependent feature, namely sample robustness, which is, to the best of our knowledge, unique.

As sample robustness can be regarded a measure of the susceptibility of single points in a dataset to perturbations causing their label to change, one is immediately reminded of adversarial examples (Szegedy et al., 2013) in the context of model robustness, only that we use the pre-defined labels instead of model predictions as additional input. The term robustness itself is one of the most prominent throughout the recent literature in many different contexts (Evtimov et al., 2017), (Beggel et al., 2019), (Madry et al., 2017), (Weng et al., 2018), (Tsuzuku et al., 2018), from robust attacks to robust models/defences. State-of-the-art machine learning models are susceptible to noise, especially so when crafted purposefully (Fawzi et al., 2016). It leaves these powerful machines vulnerable to attacks either at training- (Zhu et al., 2019) or at test stage (Biggio et al., 2013), independent of the architecture used (Papernot et al., 2016b). In this work, we follow the idea that models are extensions of the label map from the (metric) subspace defined by a dataset to the whole feature space and we investigate the robustness distribution of datasets and the thereon dependent model performance as the first step towards a larger investigation of the connection between the robustness of data and robustness of models conducted presently.

Lipschitz calculus yields a mathematically well-understood approach to describe and measure model robustness as in Weng et al. (2018) or Tsuzuku et al. (2018). Framing machine learning theory in terms of metric spaces (and also building robustness concepts thereon) has been done before in Wang et al. (2016), however, not explicitly connecting it to Lipschitz calculus. In this work, we build the concept of sample robustness based on a point-wise Lipschitz constant of the label map for metric- and Banach spaces such that it applies to a wide range of different feature- and label spaces including an ample variety of metrics.

Finally, investigating the dependence of model-hyperparameters on data has been done previously; for example, in Nakkiran et al. (2020), where the authors showed that in some cases, more data could hurt model performance. In this work, we see that optimal hyperparameters can depend on the robustness distribution of both the training- and test set as well as more- and less robust subsets thereof.

3 Sample Robustness

Now we will introduce the primary measure of this work, namely sample robustness. Informally, it measures how much a sample needs to be perturbed at most in feature space in order to make it change its label, relative to the magnitude of the latter.

3.1 Definition: (Framework)

Let $\mathcal{FS}$ be a feature space, considered a metric space with metric $\mathrm{d} := \mathrm{d}_{\mathcal{FS}}$, let $\mathcal{TS}$ be the target space, considered a real Banach space with norm $\|\cdot\| := \|\cdot\|_{\mathcal{TS}}$, let $\mathbf{x} \subset \mathcal{FS}$, $\#\mathbf{x} \geq 2$, be a dataset, let $y : \mathbf{x} \to \mathcal{TS}$ be a map of labels with $\#y(\mathbf{x}) \geq 2$, connecting each sample $x \in \mathbf{x}$ to some target value (or label) $y(x)$, and let the minimal- and maximal sample distance (or diameter of $\mathbf{x}$) with respect to d be defined as:

$$\min_{\mathbf{x}} := \min_{x \neq \tilde{x}} \mathrm{d}(x, \tilde{x}) \qquad \max_{\mathbf{x}} := \mathrm{diam}(\mathbf{x}) := \max_{x, \tilde{x} \in \mathbf{x}} \mathrm{d}(x, \tilde{x})$$

3.2 Definition: (Reach)

Let $x \in \mathbf{x}$ be a sample and $\Gamma_{\mathbf{x}}(x)$ defined as:

$$\Gamma_{\mathbf{x}}(x) := \{\tilde{x} \in \mathbf{x} \mid y(x) \neq y(\tilde{x})\} \subsetneq \mathbf{x}$$

Then the **reach** of x is defined as the distance of x to $\Gamma_{\mathbf{x}}(x)$:

$$\mathbf{r}_{\mathbf{x}}(x) := \mathrm{dist}\big(x, \Gamma_{\mathbf{x}}(x)\big) = \min_{\tilde{x} \in \Gamma_{\mathbf{x}}(x)} \mathrm{d}(x, \tilde{x}) \in [\min_{\mathbf{x}}, \max_{\mathbf{x}}]$$

In other words: $\Gamma_{\mathbf{x}}(x)$ is the set of samples $\tilde{x}$ in the dataset $\mathbf{x}$ that are labelled differently from x. One may notice that $\tilde{x} \in \Gamma_{\mathbf{x}}(x) \;\Leftrightarrow\; x \in \Gamma_{\mathbf{x}}(\tilde{x})$. The reach of $x \in \mathbf{x}$ is exactly the minimal distance of the point representing the image of a "cat" to a differently labelled (coloured) point in **Figure. 1**. If x is "close" to $\tilde{x}$, one would expect their labels $y(x)$ and $y(\tilde{x})$ to be "close" as well. Taking the quotient then gives a measure of this error (comp. 3.3) and normalising the latter with respect to the magnitude of y using $\|y\|_\infty = \max_{x \in \mathbf{x}} \|y(x)\|$ yields the main concept of sample robustness in 3.4.

3.3 Definition: (Point-Wise Lipschitz Constant)

Let $x \in \mathbf{x}$, then one can define a point-wise Lipschitz constant of y as:

$$\mathcal{Q}_{\mathbf{x}}(x) := \max_{\tilde{x} \in \Gamma_{\mathbf{x}}(x)} \frac{\|y(x) - y(\tilde{x})\|}{\mathrm{d}(x, \tilde{x})}$$

3.4 Definition and Proposition: (Sample Robustness)

Let $x \in \mathbf{x}$ and $\Gamma_{\mathbf{x}}(x)$ be as in **3.2**. The **robustness of the sample** x **in** $\mathcal{FS}$ with respect to d is defined as:

$$\mathcal{R}_{\mathbf{x}}(x) := \frac{\|y\|_\infty}{\mathcal{Q}_{\mathbf{x}}(x) + \|y\|_\infty} \in (0,1)$$

It is independent of rescaling the label map y (comp. **A.3**). We can see now that the almost identical medical records of patients reacting differently to the same treatment discussed in the introduction can be considered less robust samples in the above sense.

4 EVALUATION

4.1 EVALUATION MODEL

We now evaluate the introduced concept of sample robustness. For this, we determine the robustness distribution of two datasets to cover both the classification (CIFAR-10) and regression setup (Online News Popularity Dataset, short: ONP) and investigate the effect on model-performance when using training(sub)sets stemming from different parts of the robustness spectrum for the deterministic K-Nearest Neighbour (KNN) model and the non-deterministic Convolutional Neural Networks (CNN) and Random Forest (RF) approach. We measure accuracy and loss on different test(sub)sets as these indicate how much and how well a model has learned from the training data. "Best" performances are displayed in **boldface**. In **A**, we present propositions which explain the algorithms (noted in **B**) to calculate the robustness of samples. Additional analysis for the MNIST dataset is in **F**.

The CIFAR-10 data set (Krizhevsky, 2012) consists of 60000 32^2-pixel RGB-images evenly split into the classes airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. **Figure 2** provides visual examples. The ONP dataset consists of metadata of over 39000 articles published by Mashable (Fernandes, 2015) split into 58 predictive attributes (e.g. "number of words", "data channel", "polarity of positive/negative words",...), 2 non-predictive ("URL", "timedelta"), 1 goal field ("shares"). Although this is a regression setup, one can associate a pseudo-classification task with it, i.e. classifying popular and unpopular articles.[1] For both datasets all features were normalized. For CIFAR-10 we use two different CNN architectures, one "small" CNN and the more complex ResNet-56 model (see **C.1** for details), as well as a KNN classifier. For ONP, we use a KNN regressor as well as an RF regressor (see **C.2** for details). Neither for CIFAR-10 nor for ONP we used cross-validation due to the high computational effort of computing sample robustness values for each newly formed training- $\mathbf{x}$ and test set $\mathbf{t}$ after splitting. One could determine $\mathcal{R}_{\mathbf{x}\dot\cup\mathbf{t}}(x)$ for all x beforehand and then apply cross-validation to allow for feasibility, but these robustness values would no longer be independent of each other.

For both datasets models were trained and evaluated on subsets of the training- and test-sets, respectively, stemming from the extremes of the robustness distribution. For the small CNN we first estimated its overfitting threshold on the different training(sub)sets of least- and most robust training data, for the ResNet-56 we measured after how many epochs validation loss did not improve further using different validation sets. The RF regressor we trained using different amount of trees $\in \{1,2,3,4,5,6,7,8,9\}$ and both Mean Absolute Error (MAE) and Mean Squared Error (MSE) as loss criteria. The KNN classifier and regressor we constructed for K $\in \{1,2,3,\ldots,15\}$ and both uniform weights ("uni") and weights defined by the Euclidean distance $\|\cdot\|_2$ ("dist"). The non-deterministic models we trained 25 times and averaged loss and accuracy in order to avoid focus on individual performances. The details for each model are in **C.3**.

4.2 CLASSIFICATION CENTRIC ANALYSIS (CIFAR-10)

Consider the feature space $\mathcal{FS} := [0,1]^{3\cdot 32^2}$ with the Euclidean metric $\|\cdot\|_2$ and the CIFAR-10 dataset of 50.000 training images $\mathbf{x}$ and 10.000 test images $\mathbf{t}$ labelled as $k \in \{$airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck$\}$. We regard the label maps as functions $y^k : \mathbf{x} \to \mathbb{R}$, where $y^k(x)$ is 1 if the label of x is k and 0 otherwise. The target space $\mathbb{R}$ is equipped with its Euclidean norm making it a Banach space. Using Algorithm 2 (comp. **B**) we determine the

[1] As described in Fernandes (2015) we can also associate a binary classification task with the ONP dataset, where we distinguish between popular- ($\geq$ 1400 shares) and unpopular articles and which is the basis for measuring accuracy. For almost all data(sub)sets the binary labels are equally distributed (rate is 1:1). To keep the regression setup, we applied the classification based on the predicted value of shares. Thus, we refer to this as a pseudo-classification task.

robustness of all samples in both $\mathbf{x}$ and $\mathbf{t}$, respectively, and construct subsets of least- and most robust samples, the distribution of which is shown in **Table 1**. The subscript defines the subset-size in thousands (number) and whether they belong to the least- ($\mathbf{L}$) or most ($\mathbf{M}$) robust samples of $\mathbf{x}$ or $\mathbf{t}$ (e.g. $\mathbf{x_{L40}}$ are the 40000 least robust samples of the training set $\mathbf{x}$). The graphs in **D.1** compare the relative class-wise sample robustness distribution for $\mathbf{x}$ and $\mathbf{t}$.

Table 1: Sample distribution per label of all data(sub)sets.

Class	$\mathbf{x}$	$\mathbf{x_{L40}}$	$\mathbf{x_{M40}}$	$\mathbf{x_{L25}}$	$\mathbf{x_{M25}}$	$\mathbf{t}$	$\mathbf{t_{L5}}$	$\mathbf{t_{M5}}$	$\mathbf{t_{L2}}$	$\mathbf{t_{M2}}$
"airplane"	5000	4332 (87%)	3538 (71%)	3109 (62%)	1891	1000	622 (62%)	378	303 (30%)	136 (14%)
"auto"	5000	3171 (63%)	4898 (98%)	982 (20%)	4018	1000	207 (21%)	793	24 (2%)	358 (36%)
"bird"	5000	4540 (91%)	3104 (62%)	3476 (70%)	1524	1000	675 (68%)	325	366 (37%)	116 (12%)
"cat"	5000	3795 (76%)	4358 (87%)	2153 (43%)	2847	1000	435 (44%)	565	135 (14%)	224 (14%)
"deer"	5000	4704 (94%)	2904 (58%)	3844 (77%)	1156	1000	787 (79%)	213	432 (43%)	54 (5%)
"dog"	5000	3701 (74%)	4403 (88%)	2029 (41%)	2971	1000	398 (40%)	602	123 (12%)	281 (28%)
"frog"	5000	4544 (91%)	3467 (69%)	3453 (69%)	1547	1000	711 (71%)	289	312 (31%)	94 (9%)
"horse"	5000	3789 (76%)	4517 (90%)	2048 (41%)	2952	1000	387 (39%)	613	69 (7%)	245 (25%)
"ship"	5000	4323 (86%)	3927 (79%)	2921 (58%)	2079	1000	567 (57%)	433	207 (21%)	136 (14%)
"truck"	5000	3101 (62%)	4884 (98%)	985 (20%)	4015	1000	211 (21%)	789	29 (3%)	356 (36%)
Σ	50000	40000	40000	25000	25000	10000	5000	5000	2000	2000

The relative label-distribution of both $\mathbf{x}$ and $\mathbf{t}$ is similar at the 50% quartile (black percentage values are almost equal, comp. $\mathbf{x_{L25}}$ and $\mathbf{t_{L5}}$) and at the 20%- and 80% quartiles, i.e blue- (comp. $\mathbf{x_{L40}}$ and $\mathbf{t_{M2}}$) and green (comp. $\mathbf{x_{M40}}$ and $\mathbf{t_{L2}}$) percentage values add up to about 100%, respectively, although computations for $\mathbf{x}$ and $\mathbf{t}$ were conducted independently. In **Figure. 2** some of the most- and least robust samples in $\mathbf{x}$ are displayed above their robustness value. The least robust samples of "airplane" and "ship" are the closest differently labelled images in $\mathbf{x}$ with respect to each other, hence, their values coincide. The same holds for the least robust images of "bird" and "deer". In this classification setup "visual distance" is by construction correlated with robustness as the distance of labels is either 1 or 0. More examples are in **D.2**.

0.9515 0.9456 0.9461 0.9379 0.7334 0.7334 0.7526 0.7526

Figure 2: Most- (left) and least robust samples in $\mathbf{x}$ labelled "airplane", "ship", "bird", "deer".

4.2.1 SMALL CNN

The results (LOSS \ ACC) are displayed in **Table 2**. We noted the particular number of epochs it took to reach the lowest average loss on $\mathbf{t}$ (comp. **C.3**) and the corresponding performances on all the test(sub)sets. **Table 3** displays the performance matrix of an alternative session where models were trained only for the number of epochs noted in the previous table but with loss weighted according to the relative class distribution of the test(sub)sets to account for class imbalance. **D.3** displays the accuracy and loss curves of a prototype model trained on $\mathbf{x}$ for all the test(sub)sets.

Table 2: Small CNN performance matrix and number of epochs until overfitting commences.

	$\mathbf{x}$	$\mathbf{x_{L40}}$	$\mathbf{x_{M40}}$	$\mathbf{x_{L25}}$	$\mathbf{x_{M25}}$
Epochs	85	80	80	75	75
$\mathbf{t}$	**0.6804** \ **0.7822**	0.6858 \ 0.7760	0.6925 \ 0.7766	0.8123 \ 0.7428	0.7973 \ 0.7336
$\mathbf{t_{L5}}$	0.6970 \ 0.7765	**0.6819** \ **0.7772**	0.7433 \ 0.7631	0.7517 \ 0.7570	0.8884 \ 0.7006
$\mathbf{t_{M5}}$	0.6609 \ 0.7871	0.6916 \ 0.7764	**0.6417** \ **0.7900**	0.8808 \ 0.7276	0.7039 \ 0.7676
$\mathbf{t_{L2}}$	0.6538 \ 0.7909	**0.6358** \ **0.7953**	0.7578 \ 0.7643	0.6553 \ 0.7839	0.9327 \ 0.6877
$\mathbf{t_{M2}}$	0.6684 \ 0.7852	0.7122 \ 0.7708	**0.6409** \ **0.7940**	0.9742 \ 0.7166	0.6882 \ 0.7793

Table 3: Small CNN performance matrix with re-weighted LOSS.

	$\mathbf{x}$	$\mathbf{x_{L40}}$	$\mathbf{x_{M40}}$	$\mathbf{x_{L25}}$	$\mathbf{x_{M25}}$
Epochs	85	80	80	75	75
$\mathbf{t}$	**0.6807** \ **0.7833**	0.6912 \ 0.7771	0.6972 \ 0.7763	0.8071 \ 0.7419	0.7979 \ 0.7349
$\mathbf{t_{L5}}$	0.6974 \ 0.7774	**0.6956** \ **0.7788**	0.7362 \ 0.7626	0.7651 \ 0.7563	0.8571 \ 0.7045
$\mathbf{t_{M5}}$	0.7436 \ 0.7880	0.7658 \ 0.7761	**0.7228** \ **0.7900**	0.9546 \ 0.7293	0.8030 \ 0.7660
$\mathbf{t_{L2}}$	0.7559 \ 0.7927	**0.7544** \ **0.7950**	0.8335 \ 0.7623	0.8102 \ 0.7816	0.8432 \ 0.6902
$\mathbf{t_{M2}}$	0.7943 \ 0.7896	0.8634 \ 0.7688	**0.7608** \ **0.7958**	1.1018 \ 0.7171	0.8475 \ 0.7781

Small CNN observations: overall, performance on $\mathbf{t}$ decreases by downsizing the training data and, accordingly, the number of epochs necessary to reach the least loss decreased slightly. Training on $\mathbf{x_{L40}}$ and $\mathbf{x_{M40}}$ (i.e. 80% of the original training set) decreases performance on test(sub)sets of the opposite side of the robustness spectrum but has a beneficial effect on performance on test(sub)sets of the same side of the robustness spectrum. Training on the least robust data ($\mathbf{x_{L40}}, \mathbf{x_{L25}}$) yields a better accuracy on $\mathbf{t_{M5}}, \mathbf{t_{M2}}$ compared to accuracy on $\mathbf{t_{L5}}, \mathbf{t_{L2}}$ when trained on the most robust data ($\mathbf{x_{M40}}, \mathbf{x_{M25}}$)[2]. However, re-weighted loss for $\mathbf{x_{L25}}$ is far more imbalanced than for $\mathbf{x_{M25}}$. Loss is the highest on samples stemming from the extremes of the robustness spectrum (notice that $\mathbf{t_{L2}} \subset \mathbf{t_{L5}}$ and $\mathbf{t_{M2}} \subset \mathbf{t_{M5}}$), regardless of the training(sub)set. Also, loss on $\mathbf{t_{L5}}$ is the least affected by re-weighting.

4.2.2 RESNET-56

The results (LOSS \ ACC) are displayed in **Table 4** and **Table 5**. We noted the epochs of training and the average epochs in the brackets, see **C.3** for details.

Table 4: ResNet-56 performance matrix using $\mathbf{t}$ as validation set.

	$\mathbf{x}$	$\mathbf{x_{L40}}$	$\mathbf{x_{M40}}$	$\mathbf{x_{L25}}$	$\mathbf{x_{M25}}$
Epochs	20+1 (15)	20+1 (13)	15+1 (12)	15+1 (11)	15+1 (11)
$\mathbf{t}$	**0.7568** \ **0.8535**	0.8466 \ 0.8316	0.8419 \ 0.8308	1.0567 \ 0.7663	1.0567 \ 0.7656
$\mathbf{t_{L5}}$	**0.7735** \ **0.8493**	0.8404 \ 0.8332	0.8811 \ 0.8186	1.0099 \ 0.7821	1.1431 \ 0.7351
$\mathbf{t_{M5}}$	**0.7397** \ **0.8585**	0.8520 \ 0.8290	0.8045 \ 0.8417	1.1019 \ 0.7519	0.9715 \ 0.7938
$\mathbf{t_{L2}}$	**0.7236** \ **0.8570**	0.7769 \ 0.8453	0.8691 \ 0.8168	0.9177 \ 0.8034	1.1785 \ 0.7193
$\mathbf{t_{M2}}$	**0.7542** \ **0.8561**	0.8842 \ 0.8209	0.7971 \ 0.8419	1.1698 \ 0.7390	0.9576 \ 0.8010

Table 5: ResNet-56 performance matrix using $\mathbf{t_L} \setminus \mathbf{t_M}$ as validation sets, according to the robustness distribution of the training(sub)sets (left) and for the opposite setup with validation sets switched.

$\mathbf{x_{L40}}$	$\mathbf{x_{M40}}$	$\mathbf{x_{L25}}$	$\mathbf{x_{M25}}$		$\mathbf{x_{L40}}$	$\mathbf{x_{M40}}$	$\mathbf{x_{L25}}$	$\mathbf{x_{M25}}$
20+1 (14)	20+1 (15)	15+1 (8)	15+1 (8)	Epochs	20+1 (15)	20+1 (13)	15+1 (9)	15+1 (10)
0.8476 \ 0.8314	0.8349 \ 0.8316	1.0567 \ 0.7681	1.0631 \ 0.7637	$\mathbf{t}$	**0.8341** \ 0.8315	0.8389 \ **0.8326**	1.0655 \ 0.7680	1.0512 \ 0.7667
0.8351 \ **0.8335**	0.8708 \ 0.8213	1.0122 \ 0.7833	1.1543 \ 0.7341	$\mathbf{t_{L5}}$	**0.8322** \ 0.8326	0.8756 \ 0.8220	1.0179 \ 0.7823	1.1354 \ 0.7389
0.8534 \ 0.8283	0.8001 \ 0.8422	1.1085 \ 0.7525	0.9703 \ 0.7942	$\mathbf{t_{M5}}$	0.8326 \ 0.8298	**0.7992** \ **0.8424**	1.1089 \ 0.7544	0.9666 \ 0.7928
0.7788 \ 0.8460	0.8583 \ 0.8191	0.9115 \ 0.8045	1.1769 \ 0.7186	$\mathbf{t_{L2}}$	**0.7697** \ **0.8467**	0.8546 \ 0.8194	0.9362 \ 0.8011	1.1668 \ 0.7202
0.8946 \ 0.8183	**0.8017** \ 0.8408	1.1683 \ 0.7389	0.9618 \ 0.8018	$\mathbf{t_{M2}}$	0.8745 \ 0.8185	0.8094 \ **0.8428**	1.1614 \ 0.7401	0.9603 \ 0.7986

ResNet-56 observations: it exists the same performance relation depending on the robustness distribution of the training(sub)set as for the small CNN (comp. **Table 2**), only this time there is no (partial) improvement. However, the average epochs it took for the models trained on $\mathbf{x_{L25}}$ and $\mathbf{x_{M25}}$ to minimize validation loss (comp. **Table 5**) decreased with the size of the validation set and even more so if the robustness distribution of the latter corresponds to the distribution of the training(sub)set without a noticeable change in performance.

4.2.3 KNN

For KNN the highest accuracy in every case was observed for K=1, **Table 6** shows the performance matrix (note that for $K = 1$ both weights coincide). **D.4**[3] displays the accuracy values for all K and both weights using $\mathbf{x}$, $\mathbf{x_{L25}}$, $\mathbf{x_{M25}}$ as reference sets.

Table 6: 1NN accuracy matrix.

	$\mathbf{x}$	$\mathbf{x_{L40}}$	$\mathbf{x_{M40}}$	$\mathbf{x_{L25}}$	$\mathbf{x_{M25}}$
$\mathbf{t}$	0.3539	0.3434	**0.3863**	0.3135	0.3806
$\mathbf{t_{L5}}$	0.3700	0.3702	**0.3948**	0.3656	0.3356
$\mathbf{t_{M5}}$	0.3378	0.3166	0.3778	0.2614	**0.4256**
$\mathbf{t_{L2}}$	**0.4095**	**0.4095**	0.3950	**0.4095**	0.3055
$\mathbf{t_{M2}}$	0.3575	0.3155	0.3875	0.2410	**0.4320**

KNN observations: by removing 20% of the least robust samples of $\mathbf{x}$, accuracy on the whole test set $\mathbf{t}$ increased by about 3%. However, the best performances on the individual test(sub)sets

[2]E.g.: the difference between accuracy on $\mathbf{t_{L2}}$ and accuracy on $\mathbf{t_{M2}}$ differs by about 7% when trained on $\mathbf{x_{L25}}$, but about almost 11% when trained on $\mathbf{x_{M25}}$.

[3]Note that KNN performs better when information about distance is provided (i.e. using "dist" weights).

are distributed among $\mathbf{x_{M40}}$, $\mathbf{x_{L25}}$, $\mathbf{x_{M25}}$. As for the small CNN one can notice that leaving out the most- or least robust samples in the reference set decreases accuracy on the corresponding test(sub)sets (except for $\mathbf{t_{L5}}$, where accuracy increased when using $\mathbf{x_{M40}}$), but again does not affect accuracy on the opposite side of the spectrum as much. In fact, for $\mathbf{x_{M25}}$ accuracy even increases on $\mathbf{t_{M5}}$ and $\mathbf{t_{M2}}$ (similar to the CNN trained on $\mathbf{x_{M40}}$). Finally, we noted that accuracy on $\mathbf{t_{L2}}$ is nearly identical for both $\mathbf{x}$ and $\mathbf{x_{L25}}$ among all K (comp. **D.4**), which implies that in this case $\mathbf{x_{M25}}$ is superfluous data.

4.3 Regression Centric Analysis (Online News Popularity Dataset)

The Online News Popularity dataset (Fernandes, 2015) was split (randomly) into training set $\mathbf{x}$ and test set $\mathbf{t}$ consisting of 32,000 and 7,644 samples, respectively. One can consider each sample $x \in \mathbf{x} \dot\cup \mathbf{t}$ as an element in $\mathcal{FS} := [0,1]^{58}$, where we use the Euclidean metric $\|\cdot\|_2$ again. The label map $y : \mathbf{x} \to \mathbb{R}$ (where $\mathbb{R}$ is equipped with its Euclidean norm as well) assigns to each sample a normalised number of shares in $[0,1]$. This time we used algorithm 1 (comp. **B**) to calculate the sample robustness for each element in $\mathbf{x}$ and $\mathbf{t}$ (again independently), so that each value only depends on the data(sub)set the sample stems from. We then defined different subsets of both the training- and test set. For $\mathbf{x_{L24}}, \mathbf{x_{M24}}, \mathbf{x_{L16}}, \mathbf{x_{M16}}, \mathbf{x_{L2}}, \mathbf{x_{M2}}$ the letter again indicates to which robustness spectrum the set belongs and the number defines the amount of samples in thousands (e.g. $\mathbf{x_{M24}}$ are the 24,000 most robust samples in $\mathbf{x}$). The sets $\mathbf{t_L}$ and $\mathbf{t_M}$ consist of the least- and most robust half of $\mathbf{t}$, the sets $\mathbf{t_l}$ and $\mathbf{t_m}$ of the 500 least- and most robust samples of $\mathbf{t_L}$ and $\mathbf{t_M}$, respectively. **Figure 3** displays the relative distribution of shares for all training- and test(sub)sets. The black dashed line divides samples into those below 10,000 shares on the left side and those above 10,000 shares on the right (cumulated at 110), the percentage of which is noted in each box. For most of the data(sub)sets the graphs are quite similar as about 5% of the samples have above 10,000 shares. For both $\mathbf{x_{M2}}$ and $\mathbf{t_m}$ this holds for about 9.4% of samples. Conversely, while 7.6% of samples in $\mathbf{x_{L2}}$ have more than 10.000 shares, this holds for only 0.34% of $\mathbf{t_l}$ (i.e. less than half the relative amount). Indeed, this implicates that the label-distribution of the $\approx 6\%$ least robust samples[4] in $\mathbf{x}$ is quite different from that in $\mathbf{t}$. **E.1** compares the robustness distribution of $\mathbf{x}$ and $\mathbf{t}$.

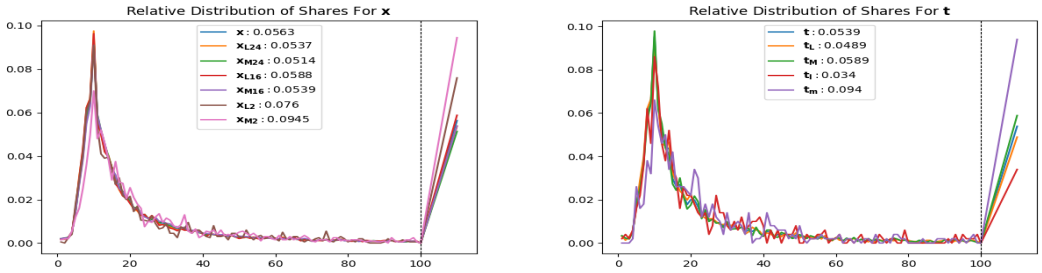

Figure 3: Relative distribution of shares for all training- and test(sub)sets.

4.3.1 KNN

Table 7 shows the performance matrix (LOSS $\setminus$ ACC) for uniform ("uni")- and distance ("dist") weights, as well as the particular choice of K where ACC was the highest on $\mathbf{t}$ (as LOSS decreases with increasing K). **Table 8** shows the performances and the specific K (in brackets) for which ACC was the highest on each respective test(sub)set. **E.2** displays the performance graphs.

KNN observations: performance is similar among both choices of weights; for "dist" weights, one can see a similar situation as for the small CNN on CIFAR-10 (comp. **Table. 2**). Training on 75% of the original dataset correlated with an increase of the optimal K, independent of whether the most- or least robust data is missing. For both weights ACC on $\mathbf{t_m}$ is the highest, however this does not hold for $\mathbf{t_l}$ and may be explained by the different distributions of the least robust samples in $\mathbf{x}$ and $\mathbf{t}$ (comp. $\mathbf{x_{L2}}$ and $\mathbf{t_l}$ in **Figure 3**). The optimal K for each test(sub)set varies between 3 and 4, except for $\mathbf{t_m}$ where it is nearly 3-4 times as high causing an increase in accuracy of about 2% -5%.

[4] $\frac{\#\mathbf{x_{L2}}}{\#\mathbf{x}} = \frac{2000}{32000} = 0.0625$ and $\frac{\#\mathbf{t_l}}{\#\mathbf{t}} = \frac{500}{7644} = 0.0654$

Table 7: ”uni” (left) / ”dist” (right) weights.

$\mathbf{x}$	$\mathbf{x_{L24}}$	$\mathbf{x_{M24}}$		$\mathbf{x}$	$\mathbf{x_{L24}}$	$\mathbf{x_{M24}}$
3	4	4	$\mathbf{k}$	3	4	4
-0.1456 \ 0.5900	**-0.1377** \ **0.5959**	-0.3945 \ 0.5907	$\mathbf{t}$	-0.1328 \ 0.5899	**-0.1303** \ **0.5942**	-0.4141 \ 0.5917
-0.0869 \ 0.5900	-0.0940 \ **0.6049**	-0.4290 \ 0.5942	$\mathbf{t_L}$	**-0.0843** \ 0.5911	-0.0899 \ **0.6023**	-0.4527 \ 0.5931
-0.3863 \ **0.5900**	-0.3169 \ 0.5869	**-0.2531** \ 0.5871	$\mathbf{t_M}$	-0.3315 \ 0.5887	-0.2962 \ 0.5861	**-0.2558** \ **0.5903**
0.0087 \ 0.5900	-0.0046 \ **0.6140**	-0.1366 \ 0.6020	$\mathbf{t_l}$	-0.0073 \ 0.5900	**-0.0046** \ **0.6120**	-0.1500 \ 0.6000
-0.1255 \ 0.6320	-0.2217 \ 0.6480	**-0.0544** \ **0.6500**	$\mathbf{t_m}$	-0.1222 \ 0.6280	-0.2224 \ 0.6460	**-0.0511** \ **0.6620**

Table 8: ”uni” (left) / ”dist” (right) weights where ACC was highest for each test(sub)set.

$\mathbf{x}$	$\mathbf{x_{L24}}$	$\mathbf{x_{M24}}$		$\mathbf{x}$	$\mathbf{x_{L24}}$	$\mathbf{x_{M24}}$
-0.1456 \ 0.5900 (3)	**-0.1377** \ **0.5959** (4)	-0.3945 \ 0.5907 (4)	$\mathbf{t}$	-0.1328 \ 0.5899 (3)	**-0.1303** \ **0.5942** (4)	-0.4141 \ 0.5917 (4)
-0.0867 \ 0.5947 (4)	-0.0940 \ **0.6049** (4)	-0.4290 \ 0.5942 (4)	$\mathbf{t_L}$	**-0.0840** \ 0.5945 (4)	-0.0899 \ **0.6023** (4)	-0.4527 \ 0.5931 (4)
-0.3863 \ 0.5900 (3)	**-0.2925** \ **0.5926** (5)	-0.3664 \ 0.5882 (3)	$\mathbf{t_M}$	-0.3315 \ 0.5887 (3)	-0.2645 \ **0.5939** (5)	**-0.2558** \ 0.5903 (4)
0.0045 \ 0.6100 (4)	-0.0046 \ **0.6140** (4)	-0.1366 \ 0.6020 (4)	$\mathbf{t_l}$	**0.0040** \ 0.6100 (4)	-0.0046 \ **0.6120** (4)	-0.1500 \ 0.6000 (4)
-0.0278 \ 0.6800 (14)	-0.1309 \ 0.6720 (13)	**-0.0165** \ **0.6820** (12)	$\mathbf{t_m}$	-0.0239 \ 0.6800 (13)	-0.1314 \ 0.6740 (14)	**-0.0185** \ **0.6840** (12)

4.3.2 Random Forest

Table 9 shows the performances matrix (LOSS \ ACC) and the number of trees where ACC on $\mathbf{t}$ was the highest. **Table 10** shows the performances and numbers of trees (in brackets) for which ACC was the highest on each test(sub)set. **E.3** displays the performance graphs.

Table 9: MAE (left) / MSE (right) as loss-criteria.

$\mathbf{x}$	$\mathbf{x_{L24}}$	$\mathbf{x_{M24}}$		$\mathbf{x}$	$\mathbf{x_{L24}}$	$\mathbf{x_{M24}}$
5	4	3	Trees	7	6	4
-0.1867 \ **0.5951**	-0.2884 \ 0.5907	**-0.1760** \ 0.5920	$\mathbf{t}$	**-0.1148** \ **0.5953**	-0.2796 \ 0.5909	-0.1305 \ 0.5928
-0.1176 \ 0.5968	-0.1583 \ 0.5964	**-0.1006** \ **0.5974**	$\mathbf{t_L}$	**-0.0590** \ **0.5998**	-0.1116 \ 0.5959	-0.0616 \ 0.5968
-0.4461 \ **0.5922**	-0.6904 \ 0.5848	-0.4743 \ 0.5864	$\mathbf{t_M}$	-0.3814 \ **0.5905**	-0.9062 \ 0.5873	**-0.3704** \ 0.5884
0.0147 \ **0.6256**	-0.0144 \ 0.6232	**0.0236** \ 0.6232	$\mathbf{t_l}$	**0.0194** \ **0.6296**	-0.0013 \ 0.6276	0.0138 \ 0.6256
-0.3125 \ **0.6564**	-0.4946 \ 0.6376	-0.3448 \ 0.6408	$\mathbf{t_m}$	**-0.3278** \ **0.6616**	-0.7771 \ 0.6456	-0.3306 \ 0.6452

Table 10: MAE (left) / MSE (right) as loss-criteria where ACC on each test(sub)set is highest.

$\mathbf{x}$	$\mathbf{x_{L24}}$	$\mathbf{x_{M24}}$		$\mathbf{x}$	$\mathbf{x_{L24}}$	$\mathbf{x_{M24}}$
-0.1867 \ **0.5951** (5)	-0.2884 \ 0.5907 (4)	**-0.1760** \ 0.5920 (3)	$\mathbf{t}$	**-0.1148** \ **0.5953** (7)	-0.2796 \ 0.5909 (6)	-0.1305 \ 0.5928 (4)
-0.2098 \ 0.5973 (3)	-0.1583 \ 0.5964 (4)	**-0.1006** \ **0.5974** (3)	$\mathbf{t_L}$	**-0.0348** \ **0.6006** (9)	-0.1617 \ 0.5963 (4)	-0.0616 \ 0.5968 (4)
-0.4461 \ **0.5922** (5)	-0.8000 \ 0.5867 (3)	**-0.3172** \ 0.5899 (4)	$\mathbf{t_M}$	-0.5619 \ **0.5925** (4)	-0.9062 \ 0.5873 (6)	**-0.2197** \ 0.5895 (7)
0.0065 \ **0.6316** (3)	-0.0198 \ 0.6288 (3)	**0.0199** \ 0.6244 (5)	$\mathbf{t_l}$	-0.0003 \ 0.6352 (6)	-0.0034 \ **0.6356** (7)	**0.0172** \ 0.6288 (5)
-0.2920 \ **0.6620** (7)	-0.2708 \ 0.6512 (9)	**-0.2054** \ **0.6620** (8)	$\mathbf{t_m}$	-0.3217 \ **0.6692** (8)	-0.6496 \ 0.6520 (9)	**-0.2261** \ 0.6672 (8)

RF observations: using only 75 % of the original training data correlates with a smaller optimal number of trees and a slight decrease in performance, independent of the metric. For $\mathbf{x_{M24}}$ the highest accuracy on $\mathbf{t}$ was achieved using about half as many trees as for $\mathbf{x}$. As for KNN one may explain the difference in ACC on $\mathbf{t_l}$ and $\mathbf{t_m}$ by the different distributions of $\mathbf{x_{L2}}$ and $\mathbf{t_l}$ (comp. **Figure 3**). LOSS increases significantly when using $\mathbf{x_{L24}}$ compared to $\mathbf{x_{M24}}$ for both MAE and MSE. The optimal number of trees is generally higher among all test(sub)sets when using MSE instead of MAE. However, for $\mathbf{t_m}$ this hyperparameter is the highest and almost identical for both metrics, similar to KNN (comp. **Table. 8**).

5 Conclusion and Future Work

We introduced the concept of sample robustness for measuring how sensitive elements of a dataset are towards label-changing perturbations. We then analysed the robustness distribution of different datasets and the relationship to the performance of (non-)deterministic models when using more- or less robust training(sub)sets. In conclusion, we found that it is possible to boost model performance by choosing particular training(sub)sets and hyperparameters depending on the robustness distribution of the test(sub)sets. Furthermore, it was possible to detect a distribution shift between two particular data(sub)sets (which correlated with diminished model accuracy) by comparing their individual robustness distributions. Possible future research directions are: **(i)** expanding experiments (using cross-validation, more datasets and models, different metrics,...); **(ii)** analyse optimal model-hyperparameters as functions h($\mathbf{x_{AB}}$, $\mathbf{t_{AB}}$) for training- $\mathbf{x_{AB}}$ and test(sub)sets $\mathbf{t_{AB}}$ of all different parts of the robustness spectrum; **(iii)** explore the relation between training on more- or less robust data and model-robustness, e.g. susceptibility to adversarial examples (Szegedy et al., 2013).

REFERENCES

Laura Beggel, Michael Pfeiffer, and Bernd Bischl. Robust anomaly detection in images using adversarial autoencoders, 01 2019.

Y. Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35:1798–1828, 08 2013. doi: 10.1109/TPAMI.2013.50.

Alexander Bernstein and Alexander Kuleshov. Low-dimensional data representation in data analysis. pp. 47–58, 10 2014. ISBN 978-3-319-11655-6. doi: 10.1007/978-3-319-11656-3_5.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. *ArXiv*, abs/1708.06131, 2013.

Lawrence Cayton. Algorithms for manifold learning. 07 2005.

Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *CoRR*, abs/1901.03407, 2019. URL `http://arxiv.org/abs/1901.03407`.

Denis Chigirev and William Bialek. Optimal manifold representation of data: An information theoretic approach. 01 2003.

Dumitru Erhan, Aaron C. Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, 2010.

Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Xiaodong Song. Robust physical-world attacks on deep learning models. 2017.

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *NIPS*, 2016.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29, 10 2013. doi: 10.1090/jams/852.

Kelwin Fernandes. A proactive intelligent decision support system for predicting the popularity of online news. 08 2015. doi: 10.1007/978-3-319-23485-4_53.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998. doi: 10.1109/5.726791.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Xiaodong Song. Delving into transferable adversarial examples and black-box attacks. *ArXiv*, abs/1611.02770, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2017.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1g5sA4twr`.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. pp. 372–387, 03 2016a. doi: 10.1109/EuroSP.2016.36.

Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *ArXiv*, abs/1605.07277, 2016b.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. ISSN 0036-8075. doi: 10.1126/science.290.5500.2319. URL https://science.sciencemag.org/content/290/5500/2319.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. *ArXiv*, abs/1802.04034, 2018.

Beilun Wang, Ji Gao, and Yanjun Qi. A theoretical framework for robustness of (deep) classifiers against adversarial examples, 2016.

Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *ArXiv*, abs/1801.10578, 2018.

Chen Zhu, W. Huang, Ali Shafahi, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets, 05 2019.

A ADDITIONAL THEORY AND PROOFS:

A.1 PROPOSITION:

Let $x \in \mathbf{x}$ have reach $\mathbf{r}(x)$. Then there exists a distinct sample $\tilde{x}^* \in \mathbf{x} - \{x\}$ that has reach at most $\mathbf{r}(x)$, i.e.:

$$\mathbf{r}_{\mathbf{x}}(\tilde{x}^*) \leq \mathbf{r}_{\mathbf{x}}(x)$$

Proof. Since $\Gamma_{\mathbf{x}}(x)$ is compact (even finite), there exists $\tilde{x}^* \neq x$ for every x such that:

$$\mathbf{r}(x) := \mathrm{dist}\big(x, \Gamma_{\mathbf{x}}(x)\big) = \min_{\tilde{x} \in \Gamma_{\mathbf{x}}(x)} \mathrm{d}(x, \tilde{x}) = \mathrm{d}(x, \tilde{x}^*) = \mathrm{d}(\tilde{x}^*, x)$$

But then for this particular x and $\tilde{x}^*$ it only holds that:

$$\mathbf{r}(x) = \mathrm{d}(\tilde{x}^*, x) \geq \min_{x \in \Gamma_{\mathbf{x}}(\tilde{x}^*)} \mathrm{d}(\tilde{x}^*, x) = \mathrm{dist}\big(\tilde{x}^*, \Gamma_{\mathbf{x}}(\tilde{x}^*)\big) = \mathbf{r}(\tilde{x}^*)$$

□

A.2 PROPOSITION:

Let $L(y)$ be the Lipschitz constant of the label map, than the following equality holds:

$$\max_{x \in \mathbf{x}} \mathcal{Q}_{\mathbf{x}}(x) = L(y)$$

Proof. One has $L(y) \geq \max_{x \in \mathbf{x}} \mathcal{Q}_{\mathbf{x}}(x)$ by construction. Furthermore:

$$L(y) = \max_{x \neq \tilde{x}} \frac{\|y(x) - y(\tilde{x})\|}{\mathrm{d}(x, \tilde{x})} = \frac{\|y(x_0) - y(\tilde{x}_0)\|}{\mathrm{d}(x_0, \tilde{x}_0)}$$

for some $x_0 \neq \tilde{x}_0$ such that $y(x_0) \neq y(\tilde{x}_0)$. Thus:

$$\frac{\|y(x_0) - y(\tilde{x}_0)\|}{\mathrm{d}(x_0, \tilde{x}_0)} \leq \mathcal{Q}_{\mathbf{x}}(x_0) \leq \max_{x \in \mathbf{x}} \mathcal{Q}_{\mathbf{x}}(x)$$

□

A.3 PROPOSITION:

W.L.O.G. one can assume that the labels lie in the unit ball of $\mathcal{TS}$, i.e. $\|y\|_\infty = 1$ (otherwise regard the map $\|y\|_\infty^{-1} \cdot y$). Thus:

$$\mathcal{R}_{\mathbf{x}}(x) = \Big(\mathcal{Q}_{\mathbf{x}}(x) + 1\Big)^{-1}$$

Proof. Suppose we have a label map $\tilde{y} := ay$ for $a \in \mathbb{R} - \{0\}$, then:

$$\tilde{\mathcal{R}}_{\mathbf{x}}(x) := \frac{\|\tilde{y}\|_\infty}{\dfrac{\|\tilde{y}(x) - \tilde{y}(\tilde{x})\|}{\mathrm{d}(x, \tilde{x})} + \|\tilde{y}\|_\infty}$$

$$= \frac{|a| \, \|y\|_\infty}{|a| \, \dfrac{\|y(x) - \tilde{y}(\tilde{x})\|}{\mathrm{d}(x, \tilde{x})} + |a| \, \|y\|_\infty} = \mathcal{R}_{\mathbf{x}}(x)$$

□

A.4 PROPOSITION:

For every $x \in \mathbf{x}$ there exists a distinct sample $\tilde{x}^* \in \mathbf{x} - \{x\}$ which is at most as robust as x, i.e.:

$$\mathcal{R}_{\mathbf{x}}(\tilde{x}^*) \leq \mathcal{R}_{\mathbf{x}}(x)$$

Proof. The term

$$\frac{\|y(x) - y(\tilde{x})\|}{\mathrm{d}(x, \tilde{x})}$$

is symmetric in x and $\tilde{x}$. Because $\Gamma_{\mathbf{x}}(x)$ is finite, one can choose $\tilde{x}^* \in \Gamma_{\mathbf{x}}(x)$ such that:

$$\mathcal{Q}_{\mathbf{x}}(x) = \max_{\tilde{x} \in \Gamma_{\mathbf{x}}(x)} \frac{\|y(x) - y(\tilde{x})\|}{\mathrm{d}(x, \tilde{x})} = \frac{\|y(x) - y(\tilde{x}^*)\|}{\mathrm{d}(x, \tilde{x}^*)}$$

But then again:

$$\frac{\|y(x) - y(\tilde{x}^*)\|}{\mathrm{d}(x, \tilde{x}^*)} = \frac{\|y(\tilde{x}^*) - y(x)\|}{\mathrm{d}(\tilde{x}^*, x)} \leq \max_{z \in \Gamma_{\mathbf{x}}(\tilde{x}^*)} \frac{\|y(\tilde{x}^*) - y(z)\|}{\mathrm{d}(\tilde{x}^*, z)} = \mathcal{Q}_{\mathbf{x}}(\tilde{x}^*)$$

□

A.5 PROPOSITION

Let $z \in \mathcal{FS} \setminus \mathbf{x}$ with label $y(z) \in y(\mathbf{x})$ and define $\tilde{\mathbf{x}} := \mathbf{x} \dot{\cup} \{z\}$. To update $\mathcal{R}_{\mathbf{x}}(x)$ to $\mathcal{R}_{\tilde{\mathbf{x}}}(x)$ for $x \in \mathbf{x} \setminus y^{-1}\big(y(z)\big)$ one comparison suffices (the proof is constructive).

Proof. Let $x \in \mathbf{x} \setminus y^{-1}\big(y(z)\big)$, then there is a $\tilde{x}^* \in \Gamma_{\mathbf{x}}(x)$ such that

$$\mathcal{Q}_{\mathbf{x}}(x) = \max_{\tilde{x} \in \Gamma_{\mathbf{x}}(x)} \frac{\|y(x) - y(\tilde{x})\|}{\mathrm{d}(x, \tilde{x})} = \frac{\|y(x) - y(\tilde{x}^*)\|}{\mathrm{d}(x, \tilde{x}^*)}$$

and $\mathcal{R}_{\tilde{\mathbf{x}}}(x) = \dfrac{\|y\|_\infty}{\mathcal{Q}_{\tilde{\mathbf{x}}}(x) + \|y\|_\infty}$, where:

$$\mathcal{Q}_{\tilde{\mathbf{x}}}(x) := \max\{\frac{\|y(x) - y(\tilde{x}^*)\|}{\mathrm{d}(x, \tilde{x}^*)}, \frac{\|y(x) - y(z)\|}{\mathrm{d}(x, z)}\}$$

□

A.6 PROPOSITION:

Let $x \in \mathbf{x}$ with reach $\mathbf{r}(x)$. If $y(x) \in \{0, e\}$ for all $x \in \mathbf{x}$, where e lies on the unit sphere in $\mathcal{TS}$ (i.e. y is a binary label map), then:

$$\mathcal{R}_{\mathbf{x}}(x) = \frac{\mathbf{r}_{\mathbf{x}}(x)}{\mathbf{r}_{\mathbf{x}}(x) + 1}$$

Proof. It holds that:

$$\mathcal{R}_{\mathbf{x}}(x) = \Big(\max_{\tilde{x} \in \Gamma_{\mathbf{x}}(x)} \frac{\|y(x) - y(\tilde{x})\|}{\mathrm{d}(x, \tilde{x})} + 1\Big)^{-1} = \Big(\max_{\tilde{x} \in \Gamma_{\mathbf{x}}(x)} \frac{1}{\mathrm{d}(x, \tilde{x})} + 1\Big)^{-1}$$

$$= \Big(\frac{1}{\min\limits_{\tilde{x}\in\Gamma_{\mathbf{x}}(x)} \mathrm{d}(x,\tilde{x})} + 1\Big)^{-1} = \Big(\frac{1}{\mathbf{r}_{\mathbf{x}}(x)} + 1\Big)^{-1} = \Big(\frac{1+\mathbf{r}_{\mathbf{x}}(x)}{\mathbf{r}_{\mathbf{x}}(x)}\Big)^{-1}$$

□

A.7 REMARK:

By construction, $\mathcal{R}_{\mathbf{x}}$ is independent of isometries of the feature space (assuming that the corresponding labels do not change). Using algorithms for manifold learning such as ISOMAP (see Tenenbaum et al. (2000)) beforehand may yield the "true" distance information of samples and hence the "true" sample robustness.

B ALGORITHM (SAMPLE ROBUSTNESS)

Let $\mathbf{x} \subset \mathcal{FS}$ be a dataset in some feature space with metric d and label map y taking values in some target space $\mathcal{TS}$ with norm $\|\cdot\|$. For a sample $x \in \mathbf{x}$ one can calculate its robustness using the algorithms below. Note that the second algorithm is a special case of the first.

Algorithm 1 Calculate $\mathcal{R}_{\mathbf{x}}(x)$ (General y)

Require: $\#y(\mathbf{x}) \geq 2$, $\|y\|_\infty = 1$
  $y \leftarrow y(x)$
  $y_j \leftarrow y(x_j)$ for any $x_j \in \mathbf{x}$ s.t. $y_j \neq y$
  $Q \leftarrow \dfrac{\|y - y_j\|}{\mathrm{d}(x, x_j)}$
  for $x_i \in \mathbf{x}$ **do**
    $y_i \leftarrow y(x_i)$
    if $y \neq y_i$ **then**
      $Q \leftarrow \max\{Q, \dfrac{\|y - y_i\|}{\mathrm{d}(x, x_i)}\}$
    end if
  end for
  print $(Q+1)^{-1}$

Algorithm 2 Calculate $\mathcal{R}_{\mathbf{x}}(x)$ (Binary y)

Require: $y(\mathbf{x}) = \{0, e\}$ with $\|e\| = 1$
  $y \leftarrow y(x)$
  $r \leftarrow \max\limits_{\mathbf{x}}$
  for $x_i \in \mathbf{x}$ **do**
    $y_i \leftarrow y(x_i)$
    if $y \neq y_i$ **then**
      $r \leftarrow \min\{r, \mathrm{d}(x, x_i)\}$
    end if
  end for
  print $\dfrac{r}{r+1}$

C MODELS

C.1 CIFAR-10

SMALL CNN

The small CNN architecture was originally taken from the Keras homepage[5] and consists of two convolutional blocks, each built using two convolutional layers with kernel size (3,3) and RELU activation function followed by MaxPooling with pool size (2,2) and with Dropout ratio of 0.25. After this, there are two dense layers, 512/RELU and 10/Softmax, using Dropout of 0.5 in between them. It was compiled using the RMSpropo optimizer with learning rate 10^{-4} and decay of 10^{-6}, categorical cross entropy loss and accuracy as the metric. We furthermore used a batch size of 32 when fitting the model.

RESNET-56

The ResNet-56 architecture (He et al., 2016) was also taken directly from the Keras homepage[6]. We used n=9, batch sizes of 32 and the pre-defined learning rate of 10^{-3}.

KNN

For the KNN classifier we used the scikit-learn pre-defined algorithm[7] both with uniform weights ("uni") and weights defined by the Euclidean metric $\|\cdot\|_2$ ("dist").

C.2 ONLINE NEWS POPULARITY DATA SET (ONP)

KNN

For the KNN regressor we used the scikit-learn pre-defined algorithm[8] both with uniform weights ("uni") and weights defined by the Euclidean metric $\|\cdot\|_2$ ("dist"). We measured the loss defined as

$$\text{LOSS} := 1 - \frac{\sum_i (y_i^{true} - y_i^{pred})^2}{\sum_i \left(y_i^{true} - \text{MEAN}(y^{true})\right)^2} \in (-\infty, 1],$$

where 1 is the optimum. Accuracy is based on the pseudo-classification task (see 4.1).

RANDOM FOREST

To construct a random forest regressor we also used the scikit-learn pre-defined algorithm[9]. We measured the same loss as for KNN and again accuracy is based on the pseudo classification task (see 4.1).

C.3 PROCEDURE DETAILS FOR THE NON-DETERMINISITC MODELS

SMALL CNN

For the small CNN we trained multiple models for an increasing amount of epochs e until we could roughly pinpoint the overfitting-threshold e_{over} with respect to the test set $\mathbf{t}$ for every training(sub)set. Then, in a second session, we trained 25 independent models (always starting with different weights) for e_{over} epochs and collected their performances on all of the test(sub)sets

[5] It was recently removed.

[6] https://keras.io/zh/examples/cifar10_resnet/

[7] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[8] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

[9] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

$\mathbf{t}, \mathbf{t_{L5}}, \mathbf{t_{M5}}, \mathbf{t_{L2}}, \mathbf{t_{M2}}$ in five accuracy- and five loss-lists, respectively. From each list, we discarded the ten highest and the ten lowest values of the 25 and then averaged the remaining five in order to avoid focus on individual model performance. After this we repeated the procedure for $e_k := e_{over} \pm 5k$, $k \in \mathbb{Z}$, epochs until we found a local loss-minimum on $\mathbf{t}$ away from the lowest and highest number of epochs in each case. Still, with all these measures taken it is possible to fall victim to the stochastic behaviour of neural networks. In some cases, we repeated second sessions to account for that. The optimal number of epochs may vary in a ± 5 vicinity. Also, since double-descent has been discovered (Nakkiran et al., 2020), one may always question the optimal overfitting threshold.

RESNET-56

The ResNet-56 model we trained 15 times on each of the different training(sub)sets for 80 epochs noting the particular epoch for which the loss on $\mathbf{t}$ was the lowest to gain an individual upper threshold. We defined the latter as the smallest $5k, k \in \mathbb{N}$, such that at most 3 of the 15 values lay above it to account for outliers. Then we conducted a second session similar to that for the small CNN, only that we trained the model for the number of epochs indicated by this upper threshold, saved the best weights as before and rebuilt the model using these weights afterwards. Finally, we topped this procedure off with an additional epoch of training using a lower learning rate of 10^{-4}. This we then repeated where we changed the validation set to $\mathbf{t_{L5}}$, respectively $\mathbf{t_{M5}}$, when trained on $\mathbf{x_{M40}}$, $\mathbf{x_{M25}}$, respectively $\mathbf{x_{M40}}$, $\mathbf{x_{M25}}$. Finally, we repeated this one last time where we changed the validation set to $\mathbf{t_{L5}}$, respectively $\mathbf{t_{M5}}$, when trained on $\mathbf{x_{M40}}$, $\mathbf{x_{M25}}$, respectively $\mathbf{x_{M40}}$, $\mathbf{x_{M25}}$ (we validated with respect to the most robust half of the test-set, when trained on the less robust data and vice versa). The epoch-history throughout training the 15 models per training(sub)set is shown below. The number in the brackets is the rounded average of the 15 values after we discarded the highest- and lowest three entries for each training(sub)set.

When loss was measured on $\mathbf{t}$:

- $\mathbf{x}$: [13, 17, 20, 12, 12, 15, 15, 18, 13, 19, 12, 23, 23, 17, 8] (15)
- $\mathbf{x_{L40}}$: [9, 17, 14, 15, 12, 18, 13, 16, 11, 12, 64, 7, 13, 12, 9] (13)
- $\mathbf{x_{M40}}$: [14, 11, 21, 11, 9, 10, 77, 15, 11, 14, 10, 8, 14, 8, 13] (12)
- $\mathbf{x_{L25}}$: [12, 7, 9, 14, 12, 77, 9, 14, 12, 21, 5, 7, 14, 7, 11] (11)
- $\mathbf{x_{M25}}$: [6, 15, 9, 13, 13, 9, 7, 10, 15, 11, 13, 17, 10, 10, 9] (11)

When loss was measured on $\mathbf{t_{L5}}$ / $\mathbf{t_{M5}}$:

- $\mathbf{x_{L40}}$: [22, 9, 13, 17, 16, 12, 80, 11, 13, 26, 7, 14, 15, 14, 13] (14)
- $\mathbf{x_{M40}}$: [26, 12, 13, 12, 15, 16, 8, 51, 16, 14, 14, 16, 6, 17, 80] (15)
- $\mathbf{x_{L25}}$: [6, 5, 9, 78, 73, 8, 7, 6, 6, 10, 13, 6, 5, 16, 8] (8)
- $\mathbf{x_{M25}}$: [12, 8, 15, 5, 9, 7, 6, 14, 5, 4, 9, 7, 8, 26, 10] (8)

When loss was measured on $\mathbf{t_{M5}}$ / $\mathbf{t_{L5}}$ (REVERSED):

- $\mathbf{x_{L40}}$: [17, 17, 16, 12, 12, 18, 13, 13, 17, 6, 22, 8, 14, 20, 15] (15)
- $\mathbf{x_{M40}}$: [13, 10, 9, 16, 9, 13, 11, 11, 7, 11, 22, 17, 50, 13, 54] (13)
- $\mathbf{x_{L25}}$: [11, 73, 14, 6, 9, 13, 8, 7, 7, 11, 7, 15, 7, 11, 6] (9)
- $\mathbf{x_{M25}}$: [10, 11, 15, 13, 13, 9, 9, 9, 7, 9, 8, 18, 7, 9, 15] (10)

RANDOM FOREST

For every number of trees, we used the same second session procedure as for the small CNN, only tailored for the ONP training- and test(sub)sets.

D CIFAR-10: ADDITIONAL MATERIAL

D.1 LABEL-WISE ROBUSTNESS DISTRIBUTION

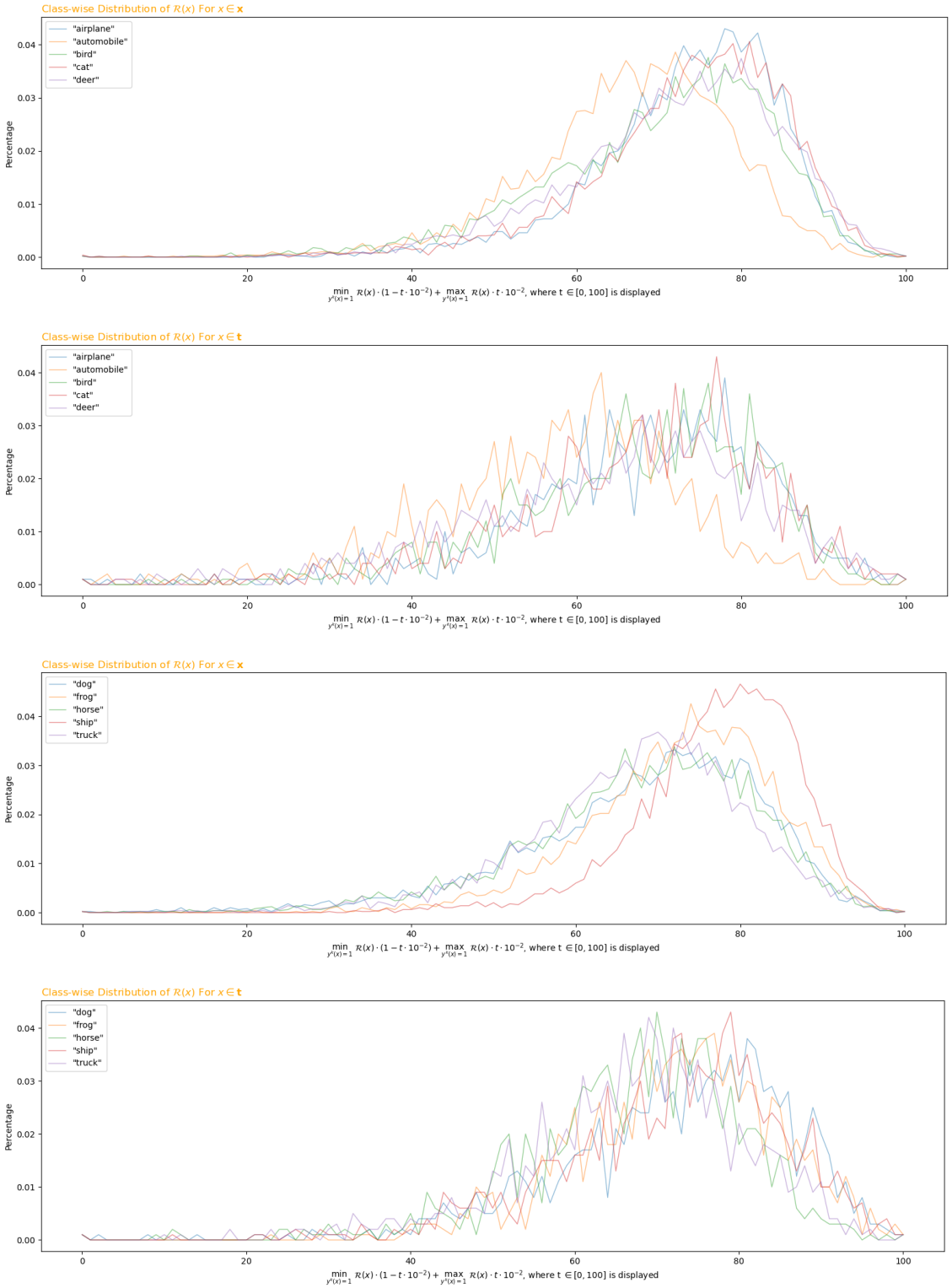

Figure 4: Relative class-wise sample robustness distribution for $\mathbf{x}$ and $\mathbf{t}$.

D.2 LEAST- AND MOST ROBUST SAMPLES IN THE CIFAR-10 TRAININGSET

Below is the juxtaposition of the least- and most robust samples in $\mathbf{x}$ for each class together with their respective sample robustness and the average model output probability (in the brackets) of each image representing its respective class (here, average is determined using the small CNN and the same procedure as for the second sessions described in **C.3**). Notably, the background of each animal or object displayed in the image impacts robustness as differently coloured surroundings already increase distance w.r.t. $\|\cdot\|_2$. The robustness values of the least robust images of "airplane" and "ship" coincide as they are the closest differently labelled samples with respect to each other. The same holds for the least robust images of "bird" and "deer". Judging by the robustness and average model probability for the "bird"- and "frog" images, there is no definitive relationship between those values.

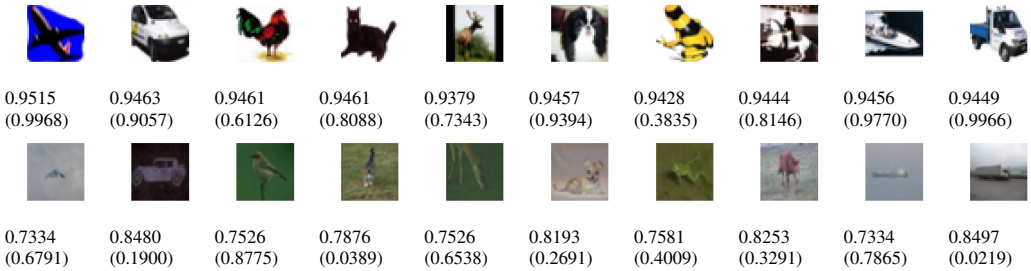

Figure 5: Most- (upper row) and least robust samples (lower row) in the CIFAR-10 training set.

D.3 PROTOTYPE MODEL LOSS AND ACCURACY OVER 85 EPOCHS

We trained a single small CNN (see C.1) on $\mathbf{x}$ for 85 epochs (which approximately showed the average performance in **Table 2**) and monitored its performance throughout the process. The graphs below show the loss and accuracy values for all test(sub)sets. From the beginning, the model did perform better on the more robust half of test set $\mathbf{t_{M5}}$ than on the less robust half $\mathbf{t_{L5}}$. The loss on $\mathbf{t_{L2}}$ is almost always the least of all five, though this does not hold for accuracy. Indeed, the subset on which the model has the highest accuracy changes every few epochs between $\mathbf{t_{L2}}$ and $\mathbf{t_{M2}}$ throughout the whole training process.

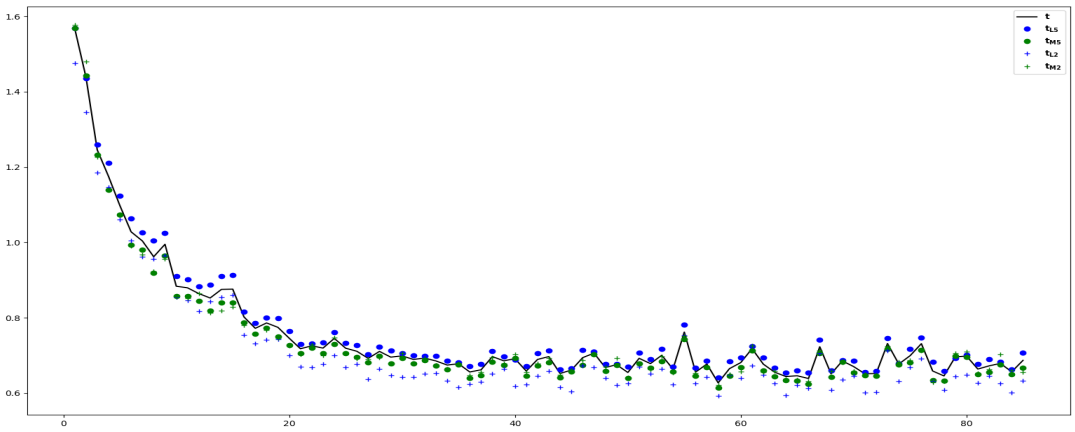

Figure 6: Epoch-wise learning behaviour of a small CNN trained on the CIFAR-10 training set $\mathbf{x}$ approximately expressing the loss in **Table 2**.

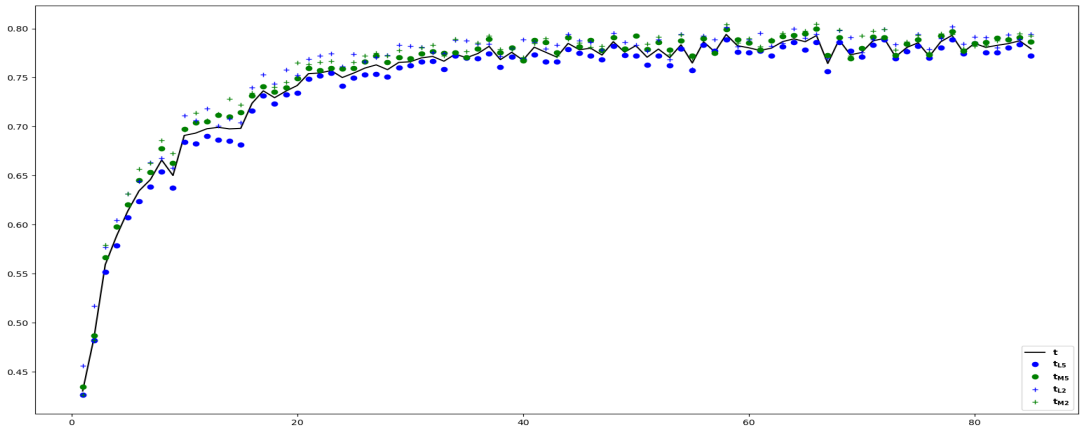

Figure 7: Epoch-wise learning behaviour of a small CNN trained on the CIFAR-10 training set $\mathbf{x}$ approximately expressing the accuracy in **Table 2**.

D.4 KNN

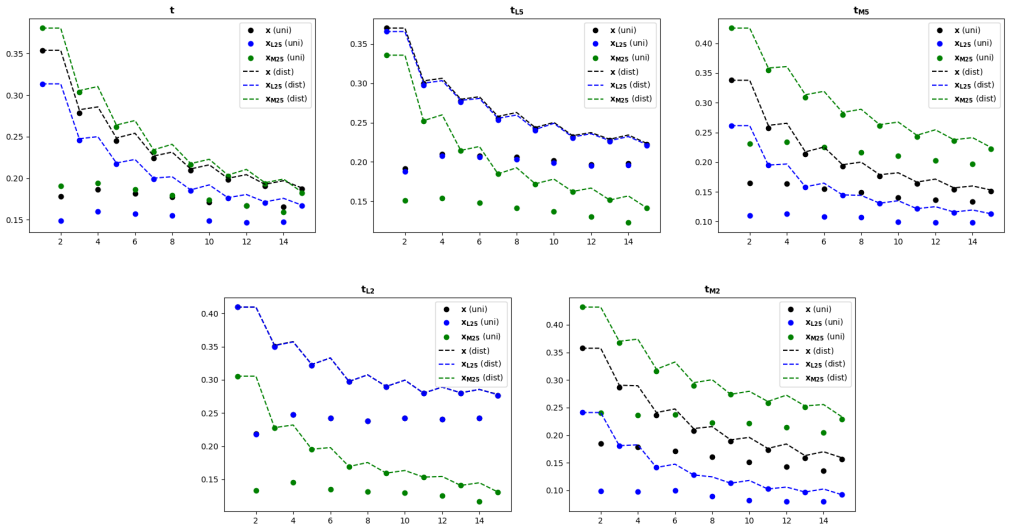

Figure 8: KNN for CIFAR-10, split for each test(sub)set. The colour of the graph corresponds to the reference training set.

E ONP: Additional Material

E.1 Relative Robustness Distribution

The minimum- and maximum sample robustness for $\mathbf{x}$ and $\mathbf{t}$ are:

$$\min_{x \in \mathbf{x}} \mathcal{R}(x) = 0.2607, \quad \max_{x \in \mathbf{x}} \mathcal{R}(x) = 0.7616, \quad \min_{x \in \mathbf{t}} \mathcal{R}(x) = 0.3792, \quad \max_{x \in \mathbf{t}} \mathcal{R}(x) = 0.8195$$

In **Figure 9** the relative sample robustness distributions normalised with respect to these values are displayed. Again, although we conducted the computations independently, they show a very similar behaviour overall (the peak for $\mathbf{x}$ is slightly higher).

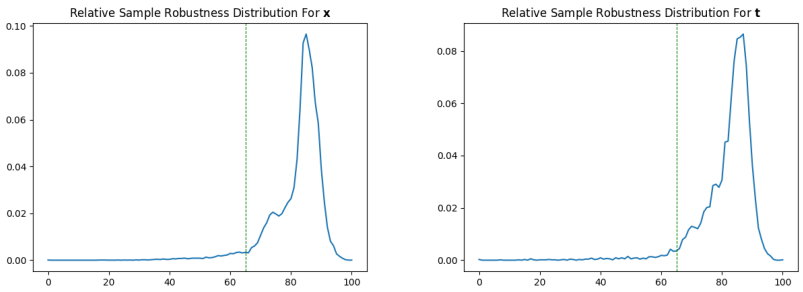

Figure 9: Relative robustness distribution for $\mathbf{x}$ and $\mathbf{t}$. The green dashed line marks the 65% threshold. For $\mathbf{x}$, 1272 samples are below this robustness level ($\approx 4\%$); for $\mathbf{t}$, 267 ($\approx 3.5\%$).

E.2 KNN

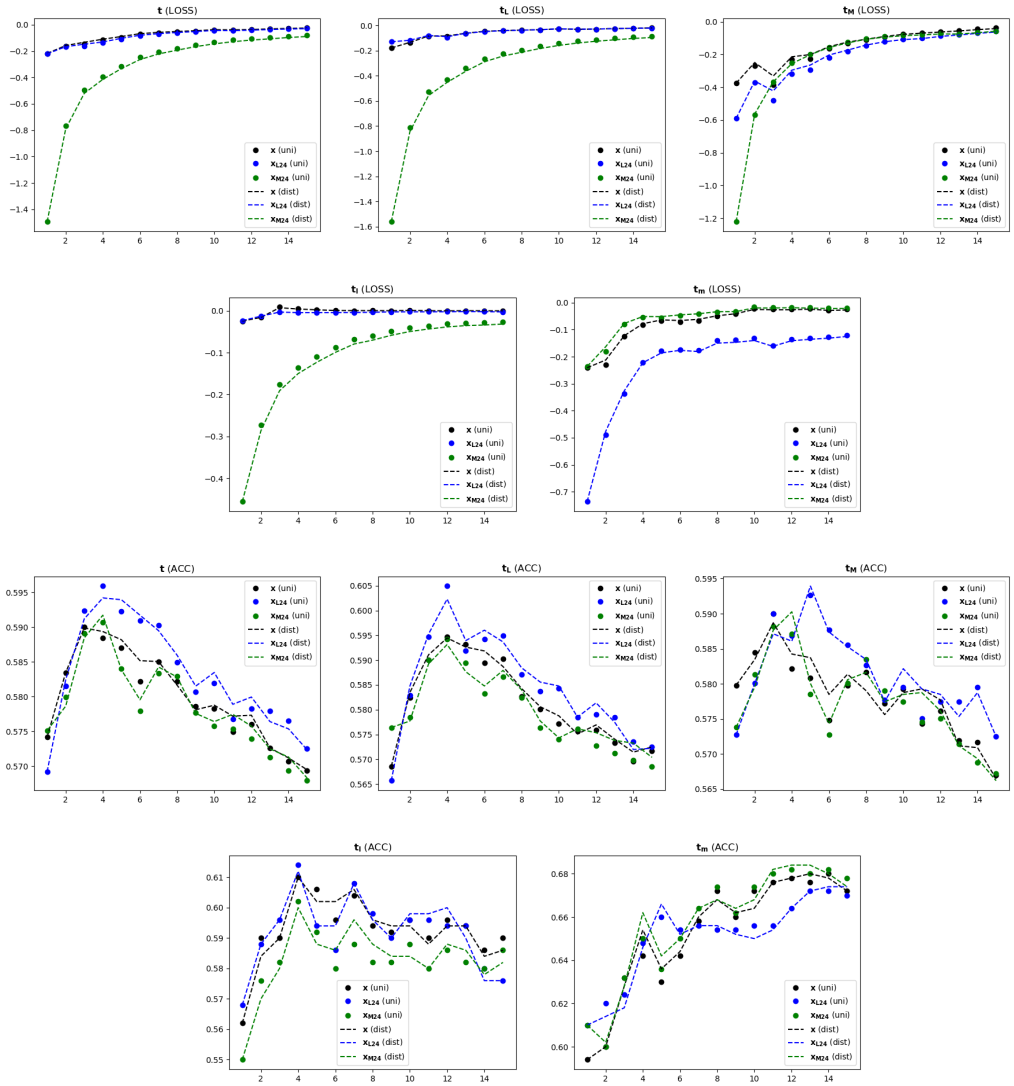

Figure 10: KNN LOSS and ACC for ONP, split for each test(sub)set. The colour of the graph corresponds to the reference set.

E.3 RANDOM FOREST

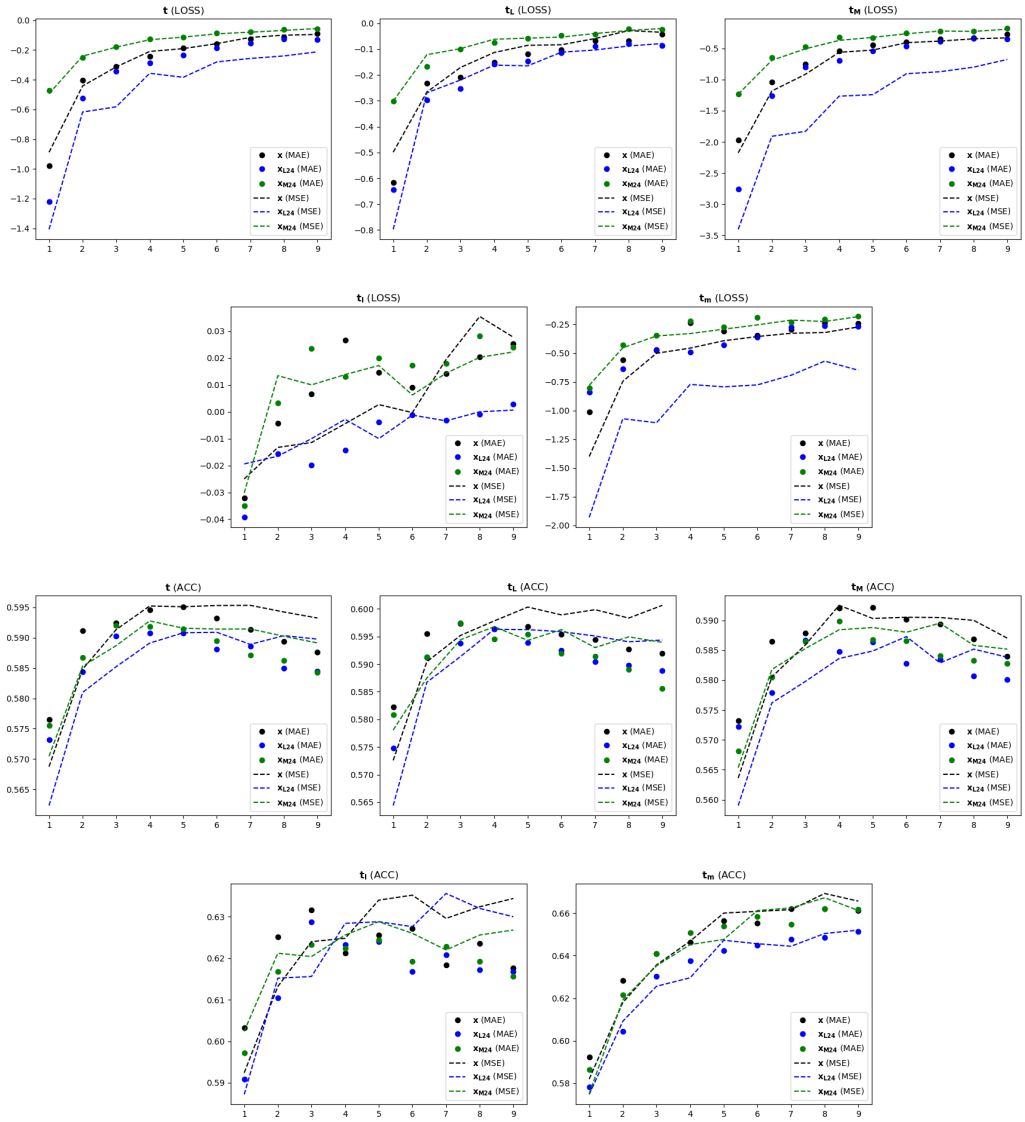

Figure 11: RF LOSS and ACC for ONP, split for each test(sub)set. The colour of the graph corresponds to the reference training set.

F MNIST

The MNIST data set (Lecun et al., 1998) consists of 60000 training- ($\mathbf{x}$) and 10000 test- ($\mathbf{t}$) 28^2-pixel greyscale images of handwritten digits. **Table. 11** shows the label distribution of subsets of most- and least robust sampes in $\mathbf{x}$ and $\mathbf{t}$ (the notation is the same as for the CIFAR-10 set). The relative label-distribution of both $\mathbf{x}$ and $\mathbf{t}$ is similar at the 50% quartile (black percentage values are almost equal, comp. $\mathbf{x_{L30}}$ and $\mathbf{t_{L5}}$) and at the 20%- and 80% quartiles as blue ($\mathbf{x_{L48}}$, $\mathbf{t_{M2}}$) and green ($\mathbf{x_{M48}}$, $\mathbf{t_{L2}}$) percentage values add up to about 100%, respectively. The class "1" is an exception as adding the green values yields a value of 120 % which indicates a certain distribution shift. Overall, images displaying the digits "1", "4", "7" and "9" were very prominent in $\mathbf{x_{L48}}, \mathbf{x_{L30}}, \mathbf{t_{L5}}, \mathbf{t_{L2}}$, whereas images in $\mathbf{x_{M48}}, \mathbf{x_{M30}}, \mathbf{t_{M5}}, \mathbf{t_{M2}}$ mainly consist of "0"s, "2"s and "6"s.

Table 11: Sample distribution per label of all data(sub)sets.

Digit	$\mathbf{x}$	$\mathbf{x_{L48}}$	$\mathbf{x_{M48}}$	$\mathbf{x_{L30}}$	$\mathbf{x_{M30}}$	$\mathbf{t}$	$\mathbf{t_{L5}}$	$\mathbf{t_{M5}}$	$\mathbf{t_{L2}}$	$\mathbf{t_{M2}}$
"0"	5923	2632 (44%)	5853 (99%)	632 (11%)	5291	980	94 (10%)	886	2 (0%)	562 (57%)
"1"	6742	6739 (100%)	2760 (41%)	6702 (99%)	40	1135	1125 (99%)	10	896 (79%)	1 (0%)
"2"	5958	2931 (49%)	5813 (98%)	815 (14%)	5143	1032	176 (17%)	856	14 (1%)	471 (46%)
"3"	6131	5040 (82%)	5828 (95%)	2219 (36%)	3912	1010	393 (39%)	617	34 (3%)	161 (16%)
"4"	5842	5333 (91%)	4310 (74%)	3819 (65%)	2023	982	656 (67%)	326	236 (24%)	91 (9%)
"5"	5421	4567 (84%)	5164 (95%)	1895 (35%)	3526	892	292 (33%)	600	18 (2%)	150 (17%)
"6"	5918	4285 (72%)	5504 (93%)	2004 (34%)	3914	958	265 (28%)	693	18 (2%)	331 (35%)
"7"	6265	5862 (94%)	4257 (68%)	4566 (73%)	1699	1028	743 (72%)	285	289 (28%)	47 (5%)
"8"	5851	4832 (83%)	5577 (95%)	2185 (37%)	3666	974	387 (40%)	587	22 (2%)	155 (16%)
"9"	5949	5779 (97%)	2934 (49%)	5163 (87%)	786	1009	869 (86%)	140	471 (47%)	31 (3%)
Σ	60000	48000	48000	30000	30000	10000	5000	5000	2000	2000

The authors of Papernot et al. (2016a) showed that when crafting source-target pairs of adversarial examples from the MNIST test set, they noticed "that classes "0", "2" and "8" are hard to start with, while classes "1", "7" and "9" are easy to start with". Interestingly, comparing the compositions of the 2000 least- and most robust samples we can see that the three most prominent classes in $\mathbf{t_{L2}}$ were indeed "1", "7" and "9" (and close thereafter "4"), while the two most prominent classes in $\mathbf{x_{M2}}$ were "0" and "2" (followed by "6" and "3" before "8").

Figure 12 displays the most and least robust samples of each class together with their sample robustness and the average probability of a shallow CNN with an average accuracy of about 98% (25 models were trained independently, accuracy values were collected in the same manner as for the small CNN on CIFAR-10). Evidently, the boldness of the written digit has a beneficial impact on robustness as the relative amount of change that altering one pixel has depended on the values of all other pixels. However, the least robust image of a "3" presents itself as a mislabelled "9" and it makes sense for this particular sample to be the least robust of its label as its distance in feature space to another image of a "9" in $\mathbf{x}$ is relatively small.

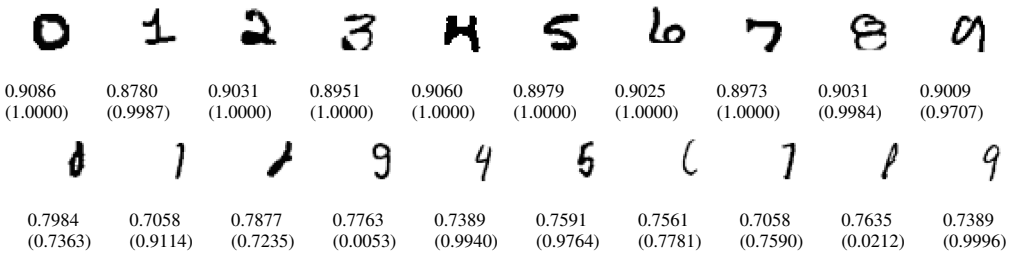

Figure 12: Most- (upper row) and least robust (lower row) samples of each label in the MNIST training set ("0" - "9" from left to right). The upper number displays the sample robustness, the lower number in the brackets is the average model probability (of a shallow CNN) of the image displaying the respective digit.

One may be tempted to regard sample robustness as a tool of anomaly detection (Chalapathy & Chawla, 2019), (Beggel et al., 2019) as it rightfully identifies the mislabelled image of a "9". To underline that it is an utterly intrinsic and metric-dependent concept and should not be used for this purpose, consider the set of "handwritten digits" in **Figure 13** and their respective robustness values (note that the image of the "4" is not missing but plain white). The relatively high value of the "3" being a "4" may be caused by the missing normalization of size and position (as was done for the MNIST set).

Figure 13: Robust samples of "handwritten digits" labelled as the number in the brackets.