Robustness of Generalized Learning Vector Quantization Models Against Adversarial Attacks

Sascha Saralajew[1(✉)], Lars Holdijk[1,2], Maike Rees[1], and Thomas Villmann[3]

[1] Dr. Ing. h.c. F. Porsche AG, Weissach, Germany
{sascha.saralajew,lars.holdijk,maike.rees}@porsche.de
[2] University of Groningen, Groningen, Netherlands
[3] Saxony Institute for Computational Intelligence and Machine Learning, University of Applied Sciences Mittweida, Mittweida, Germany
thomas.villmann@hs-mittweida.de

Abstract. Adversarial attacks and the development of (deep) neural networks robust against them are currently two widely researched topics. The robustness of Learning Vector Quantization (LVQ) models against adversarial attacks has however not yet been studied to the same extent. We therefore present an extensive evaluation of three LVQ models: Generalized LVQ, Generalized Matrix LVQ and Generalized Tangent LVQ. The evaluation suggests that both Generalized LVQ and Generalized Tangent LVQ have a high base robustness, on par with the current state-of-the-art in robust neural network methods. In contrast to this, Generalized Matrix LVQ shows a high susceptibility to adversarial attacks, scoring consistently behind all other models. Additionally, our numerical evaluation indicates that increasing the number of prototypes per class improves the robustness of the models.

1 Introduction

The robustness against adversarial attacks of (deep) neural networks (NNs) for classification tasks has become one of the most discussed topics in machine learning research since it was discovered [1,2]. By making almost imperceptible changes to the input of a NN, attackers are able to force a misclassification of the input or even switch the prediction to any desired class. With machine learning taking a more important role within our society, the security of machine learning models in general is under more scrutiny than ever.

To define an adversarial example, we use a definition similar to [3]. Suppose we use a set of scoring functions $f_j : \mathcal{X} \to \mathbb{R}$ which assign a score to each class $j \in \mathcal{C} = \{1, \ldots, N_c\}$ given an input $\mathbf{x}$ of the data space $\mathcal{X}$. Moreover, the predicted class label $c^*(\mathbf{x})$ for $\mathbf{x}$ is determined by a winner-takes-all rule $c^*(\mathbf{x}) = \arg\max_j f_j(\mathbf{x})$ and we have access to a labeled data point $(\mathbf{x}, y)$ which is correctly classified as $c^*(\mathbf{x}) = y$. An adversarial example $\tilde{\mathbf{x}}$ of the sample $\mathbf{x}$ is defined as the minimal required perturbation of $\mathbf{x}$ by $\boldsymbol{\epsilon}$ to find a point at the decision boundary or in the classification region of a different class than y, i.e.

A. Vellido et al. (Eds.): WSOM 2019, AISC 976, pp. 189–199, 2020.
https://doi.org/10.1007/978-3-030-19642-4_19

$$\min_{\boldsymbol{\epsilon}} \|\boldsymbol{\epsilon}\|, \text{ s.t. } f_j(\tilde{\mathbf{x}}) \geq f_y(\tilde{\mathbf{x}}) \text{ and } \tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon} \in \mathcal{X} \text{ and } j \neq y. \tag{1}$$

Note that the magnitude of the perturbation is measured regarding a respective norm $\|\cdot\|$. If $f_j(\tilde{\mathbf{x}}) \approx f_y(\tilde{\mathbf{x}})$, an adversarial example close to the decision boundary is found. Thus, adversarials are also related to the analysis of the decision boundaries in a learned model. It is important to define the difference between the ability to generalize and the robustness of a model [4]. Assume a model trained on a finite number of data points drawn from an unknown data manifold in $\mathcal{X}$. Generalization refers to the property to correctly classify an *arbitrary* point *from* the unknown data manifold (so-called on-manifold samples). The robustness of a model refers to the ability to correctly classify on-manifold samples that were *arbitrarily disturbed*, e.g. by injecting Gaussian noise. Depending on the kind of noise these samples are on-manifold or off-manifold adversarials (not located on the data manifold). Generalization and robustness have to be learned explicitly because the one does not imply the other.

Although Learning Vector Quantization (LVQ), as originally suggested by Kohonen in [5], is frequently claimed as one of the most robust crisp classification approaches, its robustness has not been actively studied yet. This claim is based on the characteristics of LVQ methods to partition the data space into Vorono? cells (receptive fields), according to the best matching prototype vector. For the Generalized LVQ (GLVQ) [6], considered as a differentiable cost function based variant of LVQ, robustness is theoretically anticipated because it maximizes the hypothesis margin in the *input space* [7]. This changes if the squared Euclidean distance in GLVQ is replaced by adaptive dissimilarity measures such as in Generalized Matrix LVQ (GMLVQ) [8] or Generalized Tangent LVQ (GTLVQ) [9]. They first apply a projection and measure the dissimilarity in the corresponding *projection space*, also denoted as feature space. A general robustness assumption for these models seems to be more vague.

The **observations** of this paper are: **(1)** GLVQ and GTLVQ have a high robustness because of their hypothesis margin maximization in an appropriate space. **(2)** GMLVQ is susceptible to adversarial attacks and hypothesis margin maximization does not guarantee a robust model in general. **(3)** By increasing the number of prototypes the robustness *and* the generalization ability of a LVQ model increases. **(4)** Adversarial examples generated for GLVQ and GTLVQ often make semantic sense by interpolating between digits.

2 Learning Vector Quantization

LVQ assumes a set $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{N_w}\}$ of prototypes $\mathbf{w}_k \in \mathbb{R}^n$ to represent and classify the data $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ regarding a chosen dissimilarity $d(\mathbf{x}, \mathbf{w}_k)$. Each prototype is responsible for exactly one class $c(\mathbf{w}_k) \in \mathcal{C}$ and each class is represented by at least one prototype. The training dataset is defined as a set of labeled data points $X = \{(\mathbf{x}_i, y_i) \,|\mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{C}\}$. The scoring function for the class j yields $f_j(\mathbf{x}) = -\min_{k:c(\mathbf{w}_k)=j} d(\mathbf{x}, \mathbf{w}_k)$. Hence, the predicted class $c^*(\mathbf{x})$ is the class label $c(\mathbf{w}_k)$ of the closest prototype $\mathbf{w}_k$ to $\mathbf{x}$.

Generalized LVQ: GLVQ is a cost function based variant of LVQ such that stochastic gradient descent learning (SGDL) can be performed as optimization strategy [6]. Given a training sample $(\mathbf{x}_i, y_i) \in X$, the two *closest* prototypes $\mathbf{w}^+ \in \mathcal{W}$ and $\mathbf{w}^- \in \mathcal{W}$ with correct label $c(\mathbf{w}^+) = y_i$ and incorrect label $c(\mathbf{w}^-) \neq y_i$ are determined. The dissimilarity function is defined as the squared Euclidean distance $d_E^2(\mathbf{x}, \mathbf{w}_k) = (\mathbf{x} - \mathbf{w}_k)^T (\mathbf{x} - \mathbf{w}_k)$. The cost function of GLVQ is

$$E_{GLVQ}(X, \mathcal{W}) = \sum_{(\mathbf{x}_i, y_i) \in X} l(\mathbf{x}_i, y_i, \mathcal{W}) \tag{2}$$

with the local loss $l(\mathbf{x}_i, y_i, \mathcal{W}) = \varphi(\mu(\mathbf{x}_i, y_i, \mathcal{W}))$ where φ is a monotonically increasing differentiable activation function. The classifier function μ is defined as

$$\mu(\mathbf{x}_i, y_i, \mathcal{W}) = \frac{d^+(\mathbf{x}_i) - d^-(\mathbf{x}_i)}{d^+(\mathbf{x}_i) + d^-(\mathbf{x}_i)} \in [-1, 1] \tag{3}$$

where $d^{\pm}(\mathbf{x}_i) = d_E^2(\mathbf{x}_i, \mathbf{w}^{\pm})$. Thus, $\mu(\mathbf{x}_i, y_i, \mathcal{W})$ is negative for a correctly classified training sample $(\mathbf{x}_i, y_i)$ and positive otherwise. Since $l(\mathbf{x}_i, y_i, \mathcal{W})$ is differentiable, the prototypes $\mathcal{W}$ can be learned by a SGDL approach.

Generalized Matrix LVQ: By substituting the dissimilarity measure d_E^2 in GLVQ with an adaptive dissimilarity measure

$$d_{\boldsymbol{\Omega}}^2(\mathbf{x}, \mathbf{w}_k) = d_E^2(\boldsymbol{\Omega}\mathbf{x}, \boldsymbol{\Omega}\mathbf{w}_k), \tag{4}$$

GMLVQ is obtained [8]. The relevance matrix $\boldsymbol{\Omega} \in \mathbb{R}^{r \times n}$ is learned during training in parallel to the prototypes. The parameter r controls the projection dimension of $\boldsymbol{\Omega}$ and must be defined in advance.

Generalized Tangent LVQ: In contrast to the previous methods, GTLVQ [9] defines the prototypes as affine subspaces in $\mathbb{R}^n$ instead of points. More precisely, the set of prototypes is defined as $\mathcal{W}_T = \{(\mathbf{w}_1, \mathbf{W}_1), \ldots, (\mathbf{w}_{N_w}, \mathbf{W}_{N_w})\}$ where $\mathbf{W}_k \in \mathbb{R}^{n \times r}$ is the r-dimensional basis and $\mathbf{w}_k$ is the translation vector of the affine subspace. Together with the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^r$, they form the prototype as affine subspace $\mathbf{w}_k + \mathbf{W}_k\boldsymbol{\theta}$. The tangent distance is defined as

$$d_T^2(\mathbf{x}, (\mathbf{w}_k, \mathbf{W}_k)) = \min_{\boldsymbol{\theta} \in \mathbb{R}^r} d_E^2(\mathbf{x}, \mathbf{w}_k + \mathbf{W}_k\boldsymbol{\theta}) \tag{5}$$

where r is a hyperparameter. Substituting d_E^2 in GLVQ with d_T^2 and redefining the set of prototypes to $\mathcal{W}_T$ yields GTLVQ. The affine subspaces defined by $(\mathbf{w}_k, \mathbf{W}_k)$ are learned by SGDL.

3 Experimental Setup

In this section adversarial attacks as well as robustness metrics are introduced and the setup of the evaluation is explained. The setup used here follows the one

presented in [10] with a few minor modifications to the study of LVQ methods. All experiments and models were implemented using the KERAS framework in PYTHON on top of TENSORFLOW.[1] All evaluated LVQ models are made available as pretrained TENSORFLOW graphs and as part of the FOOLBOX ZOO[2] at https://github.com/LarsHoldijk/robust_LVQ_models.

The FOOLBOX [11] implementations with default settings were used for the attacks. The evaluation was performed using the MNIST dataset as it is one of the most used datasets for robust model evaluation in the literature. Despite being considered by many as a solved 'toy' dataset with state-of-the-art (SOTA) deep learning models reaching close to perfect classification accuracy, the defense of adversarial attacks on MNIST is still far from being trivial [10]. The dataset consists of handwritten digits in the data space $\mathcal{X} = [0, 1]^n$ with $n = 28 \cdot 28$. We trained our models on the 60K training images and evaluated all metrics and scores on the *complete* 10K test images.

3.1 Adversarial Attacks

Adversarial attacks can be grouped into two different approaches, white-box and black-box, distinguished by the amount of knowledge about the model available to the attacker. White-box or gradient-based attacks are based on exploiting the interior gradients of the NNs, while black-box attacks rely only on the output of the model, either the logits, the probabilities or just the predicted discrete class labels. Each attack is designed to optimize the adversarial image regarding a given norm. Usually, the attacks are defined to optimize over L^p norms (or p-norms) with $p \in \{0, 2, \infty\}$ and, therefore, are called L^p-attacks.

In the evaluation, nine attacks including white-box and black-box attacks were compared. The white-box attacks are: Fast Gradient Sign Method (FGSM) [1], Fast Gradient Method (FGM), Basic Iterative Method (BIM) [12], Momentum Iterative Method (MIM) [13] and Deepfool [14]. The black-box attacks are: Gaussian blur, Salt-and-Pepper (S&P), Pointwise [10] and Boundary [15]. See Table 1 for the L^p definition of each attack. Note that some of the attacks are defined for more than one norm.

3.2 Robustness Metrics

The robustness of a model is measured by four different metrics, all based on the *adversarial distances* $\delta_A(\mathbf{x}, y)$. Given a labeled test sample $(\mathbf{x}, y)$ from a test set T and an adversarial L^p-attack A, $\delta_A(\mathbf{x}, y)$ is defined as: **(1)** zero if the data sample is misclassified $c^*(\mathbf{x}) \neq y$; **(2)** $\|\boldsymbol{\epsilon}\|_p = \|\tilde{\mathbf{x}} - \mathbf{x}\|_p$ if A found an adversary $\tilde{\mathbf{x}}$ and $c^*(\mathbf{x}) = y$; **(3)** ∞ if no adversary was found by A and $c^*(\mathbf{x}) = y$.

For each attack A the *median-δ_A* score is defined as median $\{\delta_A(\mathbf{x}, y) \mid (\mathbf{x}, y) \in T\}$, describing an averaged δ_A over T robust to outliers.[3] The *median-δ_p^**

[1] TENSORFLOW: www.tensorflow.org; KERAS: www.keras.io.

[2] https://foolbox.readthedocs.io/en/latest/modules/zoo.html.

[3] Hence, *median-δ_A* can be ∞ if for over 50% of the samples no adversary was found.

score is computed for all L^p-attacks as the median $\{\delta_p^*(\mathbf{x}, y) \mid (\mathbf{x}, y) \in T\}$ where $\delta_p^*(\mathbf{x}, y)$ is defined as $\min\{\delta_A(\mathbf{x}, y) \mid A \text{ is a } L^p\text{-attack}\}$. This score is a worst-case evaluation of the median-δ_A, assuming that each sample is disturbed by the respective worst-case attack A_p^* (the attack with the smallest distance). Additionally, the threshold accuracies *acc-A* and *acc-*A_p^* of a model over T are defined as the percentage of adversarial examples found with $\delta_A(\mathbf{x}, y) \leq t_p$, using either the given L^p-attack A for all samples or the respective worst-case attack A_p^* respectively. This metric represents the remaining accuracy of the model when only adversaries under a given threshold are considered valid. We used the following thresholds for our evaluation: $t_0 = 12$, $t_2 = 1.5$ and $t_\infty = 0.3$.

Table 1. The results of the robustness evaluation. Attacks are clustered by their L^p class, the boxes denote the type of the attack (white- or black-box). Accuracies are given in percentages and the #prototypes is recorded per class. All scores are evaluated on the test set. For each model we report the clean accuracy (clean acc.), the median-δ_A (left value) and acc-A score (right value) for each attack and the worst-case (worst-c.) analysis over all L^p-attacks by presenting the median-δ_p^* (left value) and acc-A_p^* score (right value). Higher scores mean higher robustness of the model. The median-δ_A of the most robust model in each attack is highlighted in bold. Overall, the model with the best (highest) worst-case median-δ_p^* is underlined and highlighted.

			CNN		Madry		GLVQ				GMLVQ				GTLVQ			
#prototypes							1		128		1		49		1		10	
Clean acc.				99		99		83		95		88		93		95		97
L^2	FGM	□	2.1	73	∞	96	∞	63	∞	76	0.6	7	0.8	15	∞	71	∞	81
	Deepfool	□	1.9	70	**5.5**	94	1.6	53	2.3	73	0.5	26	0.7	27	2.3	73	2.5	81
	BIM	□	1.5	50	**4.9**	94	1.5	50	2.1	68	0.6	6	0.7	8	2.1	68	2.3	77
	Gaussian	■	6.4	99	6.6	98	6.8	83	6.7	68	6.3	88	6.2	92	**7.1**	94	6.9	97
	Pointwise	■	4.2	96	2.1	80	4.5	79	5.4	92	1.6	54	2.4	78	5.5	92	**5.6**	95
	Boundary	■	1.9	76	1.5	52	2.1	61	**3.2**	76	0.6	7	0.8	7	2.8	78	3.1	86
	worst-c.		1.5	50	1.5	52	1.5	49	2.1	68	0.5	3	0.6	3	2.1	68	**2.2**	77
L^∞	FGSM	□	.17	7	**.52**	96	.17	11	.29	43	.04	0	.05	0	.22	18	.25	26
	Deepfool	□	.16	1	**.49**	95	.13	7	.22	21	.04	27	.05	19	.19	9	.22	19
	BIM	□	.12	0	**.41**	94	.12	3	.20	9	.04	0	.05	0	.17	3	.20	5
	MIM	□	.13	0	**.38**	93	.12	3	.19	9	.04	0	.05	0	.17	3	.20	5
	worst-c.		.12	0	**.38**	93	.11	2	.19	5	.03	0	.04	0	.17	3	.19	4
L^0	Pointwise	■	19	73	4	1	22	64	32	79	3	6	6	18	34	80	**35**	85
	S&P	■	65	94	17	63	126	77	**188**	92	8	37	17	61	155	91	179	95
	worst-c.		19	73	4	1	22	64	32	79	3	6	6	18	34	80	**35**	85

3.3 Training Setup and Models

All models, except the Madry model, were trained with the Adam optimizer [16] for 150 epochs using basic data augmentation in the form of random shifts by ± 2 pixels and random rotations by $\pm 15^\circ$.

NN Models: Two NNs are used as baseline models for the evaluation. The first model is a convolutional NN, denoted as CNN, with two convolutional layers and two fully connected layers. The convolutional layers have 32 and 64 filters with a stride of one and a kernel size of 3×3. Both are followed by max-pooling layers with a window size and stride each of 2×2. None of the layers use padding. The first fully connected layer has 128 neurons and a dropout rate of 0.5. All layers use the ReLU activation function except for the final fully connected output layer which uses a softmax function. The network was trained using the categorical cross entropy loss and an initial learning rate of 10^{-4} with a decay of 0.9 at plateaus.

The second baseline model is the current SOTA model for MNIST in terms of robustness proposed in [17] and denoted as Madry. This model relies on a special kind of adversarial training by considering it as a min-max optimization game: before the loss function is minimized over a given training batch, the original images are partially substituted by perturbed images with $\|\boldsymbol{\epsilon}\|_\infty \leq 0.3$ such that the loss function is *maximized* over the given batch. The Madry model was downloaded from https://github.com/MadryLab/mnist_challenge.

LVQ Models: All three LVQ models were trained using an initial learning rate of 0.01 with a decay of 0.5 at plateaus and with φ defined as the identity function. The prototypes (translation vectors) of all methods were class-wise initialized by k-means over the training dataset. For GMLVQ, we defined $\boldsymbol{\Omega}$ with $n = r$ and initialized $\boldsymbol{\Omega}$ as a scaled identity matrix with Frobenius norm one. After each update step, $\boldsymbol{\Omega}$ was normalized to again have Frobenius norm one. The basis matrices $\mathbf{W}_k$ of GTLVQ were defined by $r = 12$ and initialized by a singular value decomposition with respect to each initialized prototype $\mathbf{w}_k$ over the set of class corresponding training points [9]. The prototypes were not constrained to $\mathcal{X}$ ('box constrained') during the training, resulting in possibly non-interpretable prototypes as they can be points in $\mathbb{R}^n$.[4]

Two versions of each LVQ model were trained: one with one prototype per class and one with multiple prototypes per class. For the latter the numbers of prototypes were chosen such that all LVQ models have roughly 1M parameters. The chosen number of prototypes per class are given in Table 1 by #prototypes.

4 Results

The results of the model robustness evaluation are presented in Table 1. Figure 1 displays adversarial examples generated for each model. Below, the four most notable observations that can be made from the results are discussed.

[4] A restriction to $\mathcal{X}$ leads to an accuracy decrease of less than 1%.

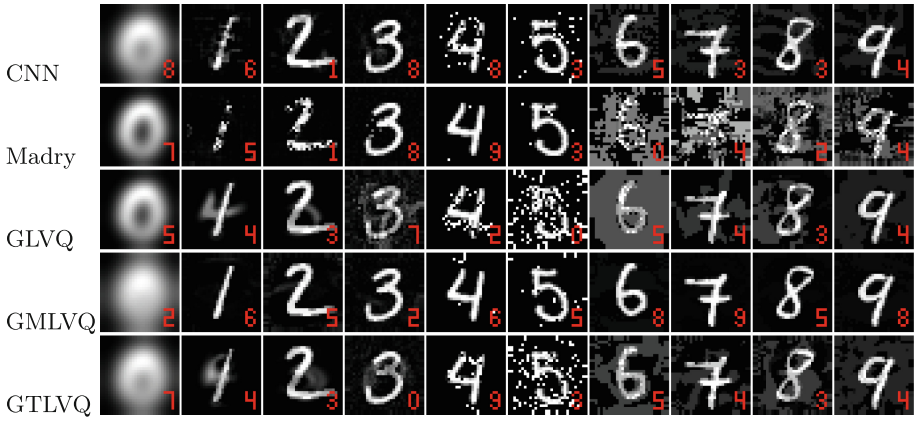

Fig. 1. For each model, adversarial examples generated by the attacks (from left to right): Gaussian, Deepfool (L_2), BIM (L_2), Boundary, Pointwise (L_0), S&P, FGSM, Deepfool (L_∞), BIM (L_∞) and MIM. For the LVQ models the version with more prototypes per class was used. The ten digits were randomly selected under the condition that every digit was classified correctly by all models. The original images are 0, 1, ..., 9 from left to right. The red digits in the lower right corners indicate the models prediction after the adversarial attack.

Hypothesis Margin Maximization in the Input Space Produces Robust Models (GLVQ and GTLVQ Are Highly Robust): Table 1 shows outstanding robustness against adversarial attacks for GLVQ and GTLVQ. GLVQ with multiple prototypes and GTLVQ with both one or more prototypes per class, outperform the NN models by a large difference for the L^0- and L^2-attacks while having a considerably lower clean accuracy. This is not only the case for individual black-box attacks but also for the worst-case scenarios. For the L^0-attacks this difference is especially apparent. A possible explanation is that the robustness of GLVQ and GTLVQ is achieved due to the input space hypothesis margin maximization [7].[5] In [7] it was stated that the hypothesis margin is a lower bound for the sample margin which is, *if defined in the input space*, used in the definition of adversarial examples (1). *Hence, if we maximize the hypothesis margin in the input space we guarantee a high sample margin and therefore, a robust model.* A first attempt to transfer this principle was made in [3] to create a robust NN by a first order approximation of the sample margin in the input space.

However, the Madry model still outperforms GLVQ and GTLVQ in the L^∞-attacks as expected. This result is easily explained using the manifold based definition of adversarial examples and the adversarial training procedure of the Madry model, which optimizes the robustness against $\|\boldsymbol{\epsilon}\|_\infty \leq 0.3$. Considering the manifold definition, one could say that Madry augmented the original

[5] Note that the results of [7] hold for GTLVQ as it can be seen as a version of GLVQ with infinitely many prototypes learning the affine subspaces.

MNIST manifold to include small L^∞ perturbations. Doing so, Madry creates a new *training*-manifold in addition to the original MNIST manifold. In other words, the L^∞ robustness of the adversarial trained Madry model can be seen as its generalization on the new training-manifold (this becomes clear if one considers the high acc-A scores for L^∞). For this reason, the Madry model is only robust against off-manifold examples that are on the generated training-manifold. As soon as off-training-manifold examples are considered the accuracy will drop fast. This was also shown in [10], where the accuracy of the Madry model is significantly lower when considering a threshold $t_\infty > 0.3$.[6]

Furthermore, the Madry model has outstanding robustness scores for gradient-based attacks in general. We accredit this effect to potential obfuscation of gradients as a side-effect of the adversarial training procedure. While [18] was not able to find concrete evidence of gradient obfuscation due to adversarial training in the Madry model, it did list black-box-attacks outperforming white-box attacks as a signal for its occurrence.

Hypothesis Margin Maximization in a Space Different to the Input Space Does Not Necessarily Produce Robust Models (GMLVQ Is Susceptible for Adversarial Attacks): In contrast to GLVQ and GTLVQ, GMLVQ has the lowest robustness score across all attacks and all methods. Taking the strong relation of GTLVQ and GMLVQ into account [9], it is a remarkable result.[7] One potential reason is, that GMLVQ maximizes the hypothesis margin in a projection space which differ in general from the input space. The margin maximization in the projection space is used to construct a model with good generalization abilities, which is why GMLVQ usually outperforms GLVQ in terms of accuracy (see the clean accuracy for GLVQ and GMLVQ with one prototype per class). However, a large margin in the projection space does not guarantee a big margin in the input space. Thus, GMLVQ does not implicitly optimize the separation margin, as used in the definition of an adversarial example (1), in the input space. Hence, GMLVQ is a good example to show that a model, which generalizes well, is not necessarily robust.

Another effect which describes the observed lack of robustness by GMLVQ is its tendency to oversimplify (to collapse data dimensions) without regularization. Oversimplification may induce heavy distortions in the mapping between

[6] For future work a more extensive evaluation should be considered: including not only the norm for which a single attack was optimized but rather a combination of all three norms. This gives a better insight on the characteristics of the attack and the defending model. The L^0 norm can be interpreted as the number of pixels that have to change, the L^∞ norm as the maximum deviation of a pixel and the L^2 norm as a kind of average pixel change. As attacks are optimized for a certain norm, only considering this norm might give a skewed impression of their attacking capability. Continuing, calculating a threshold accuracy including only adversaries that are below all three thresholds may give an interesting and more meaningful metric.

[7] GTLVQ can be seen as localized version of GMLVQ with the constraint that the $\boldsymbol{\Omega}$ matrices must be orthogonal projectors.

input and projection space, potentially creating dimensions in which a small perturbation in the input space can be mapped to a large perturbation in the projection space. These dimensions are later used to efficiently place the adversarial attack. This effect is closely related to theory known from metric learning, here oversimplification was used by [19] to optimize a classifier over $d^2_{\boldsymbol{\Omega}}$, which *maximally collapses (concentrates) the classes to single points* (related to the prototypes in GMLVQ). It is empirically shown that this effect helps to achieve a good generalization.

To improve the robustness of GMLVQ penalizing the collapsing of dimensions may be a successful approach. A method to achieve this is to force the eigenvalue spectrum of the mapping to follow a uniform distribution, as proposed in [20]. This regularization technique would also strengthen the transferability between the margin in the projection and input space. Unfortunately, it requires the possibly numerical instable computation of the derivative of a determinant of a product of $\boldsymbol{\Omega}$ which makes it impossible to train an appropriate model for MNIST using this regularization so far. The fact that GTLVQ is a constrained version of GMLVQ gives additional reason to believe that regularizations/constraints are able to force a model to be more robust.

Increasing the Number of Prototypes Improves the Ability to Generalize and the Robustness: For all three LVQ models the robustness improves if the number of prototypes per class increases. Additionally, increasing the number of prototypes leads to a better ability to generalize. This observation provides empirical evidence supporting the results of [4]. In [4] it was stated that generalization and robustness are not necessarily contradicting goals, which is a topic recently under discussion.

With multiple prototypes per class, the robustness of the GLVQ model improves by a significantly larger margin than GTLVQ. This can be explained by the high accuracy of GTLVQ with one prototype. The high accuracy with one prototype per class indicates that the data manifold of MNIST is almost flat and can therefore be described with one tangent such that introducing more prototypes does not improve the model's generalization ability. If we add more prototypes in GLVQ, the prototypes will start to approximate the data manifold and with that implicitly the tangent prototypes used in GTLVQ. With more prototypes per class, the scores of GLVQ will therefore most likely converge towards those of GTLVQ.

GLVQ and GTLVQ Require Semantically Correct Adversarial Examples: Figure 1 shows a large semantic difference between the adversarial examples generated for GLVQ / GTLVQ and the other models. A large portion of the adversarial examples generated for the GLVQ and GTLVQ models look like interpolations between the original digit and another digit.[8] This effect is especially visible for the Deepfool, BIM and Boundary attacks. In addition to this,

[8] A similar effect was observed in [10] for k-NN models.

the Pointwise attack is required to generate features from other digits to fool the models, e.g. the horizontal bar of a two in the case of GLVQ and the closed ring of a nine for GTLVQ (see digit four). In other words, for GLVQ and GTLVQ some of the attacks generate adversaries that closer resemble on-manifold samples than off-manifold. For the other models, the adversaries are more like off-manifold samples (or in the case of Madry, off-training-manifold).

5 Conclusion

In this paper we extensively evaluated the robustness of LVQ models against adversarial attacks. Most notably, we have shown that there is a large difference in the robustness of the different LVQ models, even if they all perform a hypothesis margin maximization. GLVQ and GTLVQ show high robustness against adversarial attacks, while GMLVQ scores the lowest across all attacks and all models. The discussion related to this observation has lead to four important **conclusions**: **(1)** For (hypothesis) margin maximization to lead to robust models the space in which the margin is maximized matters, this must be the same space as where the attack is placed. **(2)** Collapsed dimensions are beneficial for the generalization ability of a model. However, they can be harmful for the model's robustness. **(3)** It is possible to derive a robust model by applying a fitting regularization/constraint. This can be seen in the relation between GTLVQ and GMLVQ and is also studied for NNs [21]. **(4)** Our experimental results with an increased number of prototypes support the claim of [4], that the ability to generalize and the robustness are principally not contradicting goals.

In summary, the overall robustness of LVQ models is impressive. Using only one prototype per class and no purposefully designed adversarial training, GTLVQ is on par with SOTA robustness on MNIST. With further research, the robustness of LVQ models against adversarial attacks can be a valid reason to deploy them instead of NNs in security critical applications.

References

1. Goodfellow I, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: International conference on learning representations
2. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2014) Intriguing properties of neural networks. In: International conference on learning representations
3. Elsayed G, Krishnan D, Mobahi H, Regan K, Bengio S (2018) Large margin deep networks for classification. In: Advances in neural information processing systems, pp 850–860
4. Stutz D, Hein M, Schiele B (2018) Disentangling adversarial robustness and generalization. arXiv preprint arXiv:1812.00740
5. Kohonen T (1988) Learning vector quantization. Neural networks, 1(Supplement 1)
6. Sato A, Yamada K (1996) Generalized learning vector quantization. In: Advances in neural information processing systems, pp 423–429

7. Crammer K, Gilad-Bachrach R, Navot A, Tishby N (2003) Margin analysis of the LVQ algorithm. In: Advances in neural information processing systems, pp 479–486
8. Schneider P, Biehl M, Hammer B (2009) Adaptive relevance matrices in learning vector quantization. Neural Comput 21(12):3532–3561
9. Saralajew S, Villmann T (2016) Adaptive tangent distances in generalized learning vector quantization for transformation and distortion invariant classification learning. In: 2016 international joint conference on neural networks (IJCNN). IEEE, pp 2672–2679
10. Schott L, Rauber J, Bethge M, Brendel W (2019) Towards the first adversarially robust neural network model on MNIST. In: International conference on learning representations
11. Rauber J, Brendel W, Bethge M (2017) Foolbox: a python toolbox to benchmark the robustness of machine learning models. arXiv preprint arXiv:1707.04131
12. Kurakin A, Goodfellow I, Bengio S (2016) Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533
13. Dong Y, Liao F, Pang T, Su H, Zhu J, Hu X, Li J (2018) Boosting adversarial attacks with momentum. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 9185–9193
14. Moosavi-Dezfooli S-M, Fawzi A, Frossard P (2016) DeepFool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2574–2582
15. Brendel W, Rauber J, Bethge M (2018) Decision-based adversarial attacks: reliable attacks against black-box machine learning models. In: Proceedings of the 6th international conference on learning representations
16. Kingma DP, Ba JL (2015) Adam: a method for stochastic optimization. In: Proceedings of the international conference on learning representations, pp 1–13
17. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2018) Towards deep learning models resistant to adversarial attacks. In: International conference on learning representations
18. Athalye A, Carlini N, Wagner D (2018) Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples. In: Proceedings of the 35th international conference on machine learning
19. Globerson A, Roweis S (2006) Metric learning by collapsing classes. In: Advances in neural information processing systems, pp 451–458
20. Schneider P, Bunte K, Stiekema H, Hammer B, Villmann T, Biehl M (2010) Regularization in matrix relevance learning. IEEE Trans Neural Netw 21(5):831–840
21. Croce F, Andriushchenko M, Hein M (2018) Provable robustness of ReLU networks via maximization of linear regions. arXiv preprint arXiv:1810.07481