ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

Deep Learning Vector Quantization

Harm de Vries, Roland Memisevic and Aaron Courville

Université de Montréal

Abstract. While deep neural nets (DNN's) achieve impressive performance on image recognition tasks, previous studies have reported that DNN's give high confidence predictions for unrecognizable images. Motivated by the observation that such *fooling examples* might be caused by the extrapolating nature of the log-softmax, we propose to combine neural networks with Learning Vector Quantization (LVQ). Our proposed method, called Deep LVQ (DLVQ), achieves comparable performance on MNIST while being more robust against fooling and adversarial examples.

1 Introduction

Although Deep Neural Networks (DNN's) [1] have reached near human-level performance on challenging object recognition tasks[2, 3], recent studies highlight that there remain quite some differences with the human visual system. The first intriguing observation made is that DNN's are vulnerable to *adversarial examples* [4] – worst-case, imperceptible changes to the input that causes the DNN to label it completely different. Follow-up research [5] showed that DNN's are also easily *fooled* – images for which a DNN assigns high confidence while not coming from the data distribution.

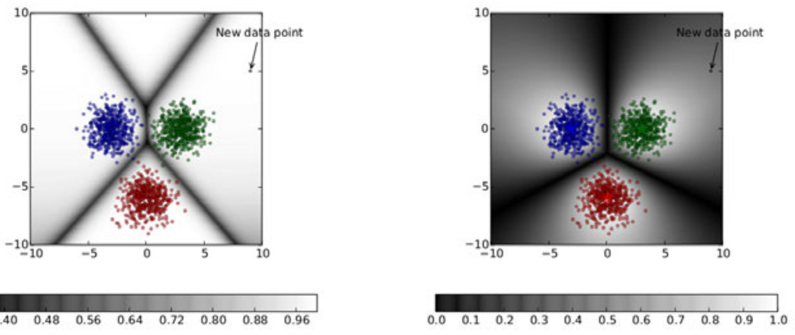
Our work departs from the observation that such fooling examples might be caused by the extrapolating nature of the log-softmax[6]:

$$L_{SM} = \sum_n \log p(\hat{y}^n|\mathbf{x}^{(n)}) = \log \frac{\exp(w_{y^{(n)}}^\top \mathbf{x} + b_{y^{(n)}})}{\sum_i \exp(w_i^\top \mathbf{x} + b_i)}, \tag{1}$$

the commonly used loss function for classication problems with neural networks. We illustrate this point with an artificially generated three-class problem shown in Fig. **??**. We can see that the softmax becomes more confident when a point is farther from the decision boundary, even though there is no data to support this decision. In contrast, a prototype based classifier like Generalized Learning Vector Quantization (GLVQ) [7] produces only high confidence values near the data points (near the prototypes). In this paper we propose to combine deep neural networks with GLVQ.

2 Generalized Learning Vector Quantization

We assume we are given training data $(\mathbf{x}^{(n)}, y^{(n)}) \in R^D \times \{0, ..., K-1\}$, $n = 1, ..., N$, where D is the dimensionality of the input, and K the number of classes. A LVQ classifier consist of a set of prototypes $\mathbf{w}_j \in R^D, j = 1, ..., M$ with an associated class label $c(w_j) \in \{1, ..., K\}$. We consider one prototype per class,



(a) Softmax (b) GLVQ

Fig. 1: An artifically generated three class problem for which we have trained (a) a softmax classifier and (b) a GLVQ classifier. The background color (white for high) indicates the confidence values for a decision, that is arg $\max_i\ p(y_j|x)$ for the softmax and $-l^{(n)}$ for the GLVQ classifier[2]. The softmax classifier will assign a high confidence value to a new data point in the right upper corner (far from the data), while GLVQ will not.

although it is straightforward to extend to multiple prototypes. Classification follows a nearest prototype scheme i.e. a new data point $\tilde{x}$ is assigned to the class of the nearest prototype $c(\arg\min_{\mathbf{w}_j} d(\tilde{x}, \mathbf{w}_j))$ according to some distance measure $d(\tilde{x}, w_j)$.

Training aims to find the locations of the prototypes such that the data points are assigned to their corresponding class labels. Generalized Learning Vector Quantization (GLVQ) [7] aims to achieve this objective by minimizing the following training criterion:

$$L_{GLVQ}(\theta) = \sum_n \phi(l^{(n)}) \quad \text{with} \quad l^{(n)} = \frac{d_+^{(n)} - d_-^{(n)}}{d_+^{(n)} + d_-^{(n)}} \tag{2}$$

where $d_+^{(n)} = \min_{c(w_j)=y} d(x^{(n)}, \mathbf{w}_j)$ and $d_-^{(n)} = \min_{c(\mathbf{w}_j)\neq y} d(x_i, \mathbf{w}_j)$ denote the distance to the closest correct and closest wrong prototype, respectively. The numerator of $l^{(n)}$ denotes the margin between the correct and wrong class, while the denominator scales the term within the interval $[-1, 1]$. The scaling function ϕ provides a handle to balance error minimization and margin maximization. Using the step function corresponds to the non-differentiable zero-one loss, and using the identity function corresponds to an average margin maximization. A trade-off between the two terms can be realised by a scaling function $\phi(x) = \exp(\gamma x)$, where $\gamma > 0$ controls the steepness of the exponential.

[2]We multiplied $l^{(n)}$ with -1 such that higher values indicates higher confidence.

Note that the numerator in $l^{(n)}$ is the reason why GLVQ only produces high confidence values when you are close to the data. Moving away from the data can be considered as adding constant $a > 0$ to the distances d^+ and d^-. This contribution will cancel in the numerator while it will add $2a$ to the denominator. For correctly classified data points (i.e. negative terms) this will decrease the confidence value.

3 Supervised neural gas

The main drawback of the GLVQ cost function is that it only displays correct training dynamics for correctly classified training examples[7]. To see this, note that for incorrectly classified examples the gradient with respect to the incorrect distance $\partial l^{(n)}/\partial d_-^{(n)} = 4d_+^{(n)}/(d_+^{(n)} - d^{(n)})^2$ is bigger than the gradient with respect to the correct class $\partial l^{(n)}/\partial d_+^{(n)} = -4d_-^{(n)}/(d_+^{(n)} - d^{(n)})^2$. The repelling force on the prototypes then dominates the attractive force which leads to prototypes that diverge from the data distribution. We empirically found this to be a problem when we tried to optimize a neural net with the GLVQ cost function.

One way to alleviate this problem is by comparing against the *average* distance to all incorrect prototypes, rather than the *closest* prototype. This smears out the repelling force over several prototypes, and makes it more likely that training dynamics are stable. This is the rationale behind Supervised Neural Gas (SNG) [8] which minimizes the following cost function[3]:

$$L_{SNG} = \sum_n \sum_{j \neq y^{(n)}} h_\tau(x^{(n)}, j)\phi\left(\frac{d_+^{(n)} - d(x^{(n)}, \mathbf{w}_j)}{d_+^{(n)} + d(x^{(n)}, \mathbf{w}_j)}\right) \tag{3}$$

The neighborhood function $h_\tau(x^{(n)}, j) = \exp(-\tau k_j)/\sum_{k=0}^{K-1}\exp(-\tau k)$ determines a (normalized) weight for each incorrect prototype loss that depends on its rank k_j, the number of prototypes that are closer to the considered data point. Our training strategy is to start with small $\tau \to 0$, essentially averaging over all incorrect prototypes, and slowly increase $\tau \to \infty$ such that we recover the GLVQ cost function.

4 Deep LVQ

It is possible to use any differentiable distance function d within the SNG and GLVQ cost functions. We propose to parameterize the distance function as:

$$d(\mathbf{x}, \mathbf{w}_j) = \|f(\mathbf{x}; \theta) - \mathbf{w}_j\|_2^2 \tag{4}$$

where f is a deep neural network with parameters θ that non-linearly projects the data points into an embedding space. During training we jointly adapt prototypes W and the neural net parameters θ top optimize Eq. **??**

[3]We note two reasonable alternatives: 1) the neighborhood function can be pushed into $l^{(n)}$ i.e. we define $d_-^{(n)} = \sum_{j \neq y^{(n)}} h_\tau(x^{(n)}, j)d(x^{(n)}, \mathbf{w}_j)$ and 2) the neighborhood function can also be replaced by a softargmin $h_\tau(x^{(n)}, j) = \exp(-\tau d(x, w_j)/\sum_k \exp(-\tau d(x^{(n)}, \mathbf{w}_k))$

Table 1: Classification results on MNIST for a feedforward net with GLVQ and softmax cost function. We compare without regularization, with batch normalization (BN), and input noise. We also report results for GMLVQ with 1 and 10 prototypes per class. Results are average test error over 3 runs, standard deviation is reported between parentheses.

	Softmax	DLVQ	GMLVQ(1)	GMLVQ (10)
No regularization	1.73 (0.02)	1.28 (0.07)	8.21 (0.01)	5.01 (0.03)
BN	1.31 (0.03)	1.22 (0.02)		
BN + input noise	1.05 (0.02)	0.98 (0.03)		

Note that a linear projection $f(x) = Ax$ boils down to Generalized Matrix Learning Vector Quantization (GMLVQ) [9, 10]. A notable difference with that work is that we directly parameterize the prototypes in embedding space. This saves computational time because we avoid a forward pass through the neural network f for the prototypes. Although the prototypes are no longer directly interpretable as class conditional exemplars in input space, recent work[11] has shown that such an interpretation is problematic due to the tendency of the GLVQ cost function to learn an injective (many-to-one) mapping f.

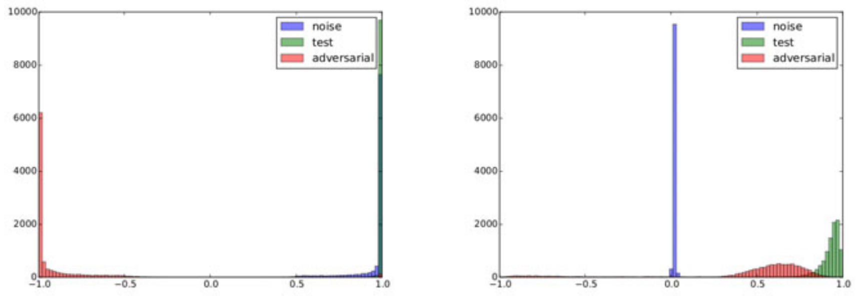
5 Experiments

The MNIST dataset consist of 70,000 handwritten digits ranging from 0 to 9. We make the usual split of $50,000$, $10,000$ and $10,000$ for the training, validation and test set, respectively. We use a fully connected neural network with $1200-1200-200$ hidden units, where all units have rectified linear activations. We initialize the prototypes for DLVQ as the class conditional means in the embedding space. We start training with a large neighborhood, i.e. $\tau = 0.1$, and linearly increase to $\tau = 30.0$. We trained our networks for 100 epochs, and present the test errors by early stopping on the validation error.

We compare training without any form of regularization, with batch normalization (BN) [3], and additive gaussian noise of standard deviation 0.5 on the inputs. The results are shown in Table 1. Without any form of regularization, DLVQ significantly outperforms the softmax. This gap reduces when we use batch normalization, which has a better regularization effect for the softmax. When we also include input noise, DLVQ still outperforms the softmax and reaches a notable 0.98% test error.

We also compare DLVQ to GMLVQ with 1 and 10 prototypes per class. Unsurprisingly, the linear transformation does not have enough capacity resulting in poor performance of 8.2%. Even increasing the number of prototypes from 1 to 10 only slightly improved the performance to 5.0%, clearly worse than DLVQ.

Fooling examples We first measure how confident the classifiers are when we feed it $10,000$ data points sampled from large uniform noise $\mathcal{U}(-10.0, 10.0)^{784}$.



(a) Softmax (b) DLVQ

Fig. 2: Histogram of the confidence values for the neural network trained with a) softmax and b) GLVQ on MNIST. The green bars indicate the confidence values for the test set, the blue bars indicate the confidence values for noise drawn from $\mathcal{U}(-10, 10)^{784}$ and the red bars denote the confidence for adversarial examples on the test set.

A histogram of the resulting confidence values is shown in Fig. 1. For the softmax, most confidence values are indistinguishable from the produced confidences on the test set. As expected, inputting large input values automatically *fools* a softmax neural network. On the other hand, DLVQ is robust against such outliers by construction.

This automatically raises the questions if DLVQ is not vulnerable to other fooling examples. We investigate this by starting from a random noise image performing gradient descent on the distance $d(x, w_j)$ to a class prototype with respect the input image. We perform 100 gradient steps such that we reached above 0.99 confidence level. The resulting images for all 10 digits are shown in Fig.2, and show remarkable resemblance with the original digits.

Adversarial examples We also verify the robustness of DLVQ against adversarial examples by the fast gradient sign method as suggested in [6]. We use $\epsilon = 0.25$, and show the resulting confidence values for the adversarial examples in Fig. 1. We made the confidence negative for the softmax if the predicted class was wrong. The softmax misclassified more than 95% of the adversarial examples, while DLVQ only misclassified 12%.

6 Conclusion

We proposed to replace the standard log-softmax loss in neural networks with GLVQ. DLVQ asks inputs of the same class to be mapped to a *point in the embedding space*, which is a harder constraint than the usual requirement of linear separability. Our experiments on MNIST show that DLVQ slightly outperforms the log-softmax, probably due to this regularization effect. More interestingly,

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

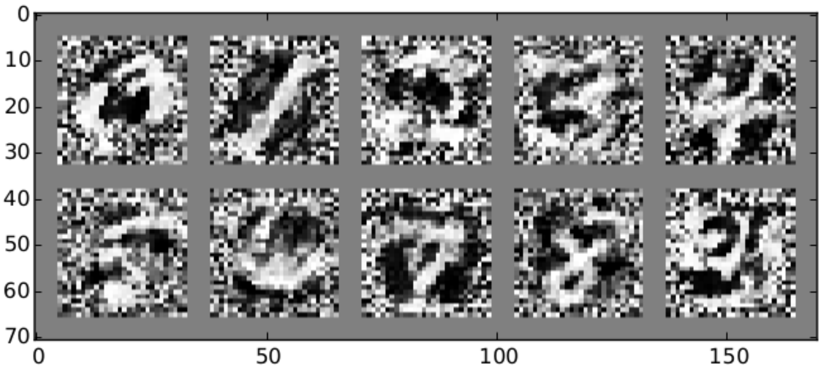


Fig. 3: High confidence (> 0.99) images for DLVQ as generated by gradient descent on random noise. All 10 digits show some resemblance with the original.

we have shown that DLVQ is more robust against fooling and adversarial examples.

References

[1] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[4] 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.

[5] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR*, abs/1412.1897, 2014.

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

[7] Atsushi Sato and Keiji Yamada. Generalized learning vector quantization. *Advances in neural information processing systems*, pages 423–429, 1996.

[8] Barbara Hammer, Marc Strickert, and Thomas Villmann. Supervised neural gas with general similarity measure. *Neural Processing Letters*, 21(1):21–44, 2005.

[9] Petra Schneider, Michael Biehl, and Barbara Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561, 2009.

[10] Kerstin Bunte, Petra Schneider, Barbara Hammer, Frank-Michael Schleif, Thomas Villmann, and Michael Biehl. Limited rank matrix learning, discriminative dimension reduction and visualization. *Neural Networks*, 26:159–173, 2012.

[11] Harm de Vries. Stationarity and uniqueness of generalized matrix learning vector quantization. *MIWOCI Workshop-2013*, page 16, 2013.