Schedae Informaticae Vol. 25 (2016): 49–59
doi: 10.4467/20838476SI.16.004.6185

tfml 2017
theoretical foundations
of machine learning, Kraków

On Loss Functions for Deep Neural Networks in Classification

Katarzyna Janocha[1], Wojciech Marian Czarnecki[2,1]
[1]Faculty of Mathematics and Computer Science,
Jagiellonian University, Kraków, Poland
[2]DeepMind, London, UK
e-mail: *kasiajanocha@gmail.com, lejlot@google.com*

Abstract. Deep neural networks are currently among the most commonly used classifiers. Despite easily achieving very good performance, one of the best selling points of these models is their modular design – one can conveniently adapt their architecture to specific needs, change connectivity patterns, attach specialised layers, experiment with a large amount of activation functions, normalisation schemes and many others. While one can find impressively wide spread of various configurations of almost every aspect of the deep nets, one element is, in authors' opinion, underrepresented – while solving classification problems, vast majority of papers and applications simply use log loss. In this paper we try to investigate how particular choices of loss functions affect deep models and their learning dynamics, as well as resulting classifiers robustness to various effects. We perform experiments on classical datasets, as well as provide some additional, theoretical insights into the problem. In particular we show that $\mathcal{L}_1$ and $\mathcal{L}_2$ losses are, quite surprisingly, justified classification objectives for deep nets, by providing probabilistic interpretation in terms of expected misclassification. We also introduce two losses which are not typically used as deep nets objectives and show that they are viable alternatives to the existing ones.

Keywords: loss function, deep learning, classification theory.

Received: 11 December 2016 / Accepted: 30 December 2016

1. Introduction

For the last few years the Deep Learning (DL) research has been rapidly developing. It evolved from tricky pretraining routines [1] to a highly modular, customisable framework for building machine learning systems for various problems, spanning from image recognition [2], voice recognition and synthesis [3] to complex AI systems [4]. One of the biggest advantages of DL is enormous flexibility in designing each part of the architecture, resulting in numerous ways of putting priors over data inside the model itself [1], finding the most efficient activation functions [5] or learning algorithms [6]. However, to authors' best knowledge, most of the community still keeps one element nearly completely fixed – when it comes to classification, we use log loss (applied to softmax activation of the output of the network). In this paper we try to address this issue by performing both theoretical and empirical analysis of effects various loss functions have on the training of deep nets.

It is worth noting that Tang et al. [7] showed that well fitted hinge loss can outperform log loss based networks in typical classification tasks. Lee et al. [8] used squared hinge loss for classification tasks, achieving very good results. From slightly more theoretical perspective Choromanska et al. [9] also considered $\mathcal{L}_1$ loss as a deep net objective. However, these works seem to be exceptions, appear in complete separation from one another, and usually do not focus on any effect of the loss function but the final performance. Our goal is to show these losses in a wider context, comparing one another under various criteria and provide insights into when – and why – one should use them.

Table 1. List of losses analysed in this paper. $\mathbf{y}$ is true label as one-hot encoding, $\hat{\mathbf{y}}$ is true label as +1/-1 encoding, $\mathbf{o}$ is the output of the last layer of the network, $\cdot^{(j)}$ denotes jth dimension of a given vector, and $\sigma(\cdot)$ denotes probability estimate.

Symbol	Name	Equation
$\mathcal{L}_1$	L_1 loss	$\|\mathbf{y} - \mathbf{o}\|_1$
$\mathcal{L}_2$	L_2 loss	$\|\mathbf{y} - \mathbf{o}\|_2^2$
$\mathcal{L}_1 \circ \sigma$	expectation loss	$\|\mathbf{y} - \sigma(\mathbf{o})\|_1$
$\mathcal{L}_2 \circ \sigma$	regularised expectation loss[1]	$\|\mathbf{y} - \sigma(\mathbf{o})\|_2^2$
$\mathcal{L}_\infty \circ \sigma$	Chebyshev loss	$\max_j \lvert\sigma(\mathbf{o})^{(j)} - \mathbf{y}^{(j)}\rvert$
hinge	hinge [7] (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})$
hinge^2	squared hinge (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})^2$
hinge^3	cubed hinge (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})^3$
log	log (cross entropy) loss	$-\sum_j \mathbf{y}^{(j)} \log \sigma(\mathbf{o})^{(j)}$
$\log^2$	squared log loss	$-\sum_j [\mathbf{y}^{(j)} \log \sigma(\mathbf{o})^{(j)}]^2$
tan	Tanimoto loss	$\frac{-\sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}{\|\sigma(\mathbf{o})\|_2^2 + \|\mathbf{y}\|_2^2 - \sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}$
D_{CS}	Cauchy-Schwarz Divergence [10]	$-\log \frac{\sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}{\|\sigma(\mathbf{o})\|_2 \|\mathbf{y}\|_2}$

[1] See Proposition 1.

This work focuses on 12 loss functions, described in Table 1. Most of them appear in deep learning (or more generally – machine learning) literature, however some in slightly different context than a classification loss. In the following section we present new insights into theoretical properties of a couple of these losses and then provide experimental evaluation of resulting models' properties, including the effect on speed of learning, final performance, input data and label noise robustness as well as convergence for simple dataset under limited resources regime.

2. Theory

Let us begin with showing interesting properties of $\mathcal{L}_p$ functions, typically considered as purely regressive losses, which should not be used in classification. $\mathcal{L}_1$ is often used as an auxiliary loss in deep nets to ensure sparseness of representations. Similarly, $\mathcal{L}_2$ is sometimes (however nowadays quite rarely) applied to weights in order to prevent them from growing to infinity. In this section we show that – despite their regression roots – they still have reasonable probabilistic interpretation for classification and can be used as a main classification objective.

We use the following notation: $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N \subset \mathbb{R}^d \times \{0,1\}^K$ is a training set, an iid sample from unknown $P(\mathbf{x}, \mathbf{y})$ and σ denotes a function producing probability estimates (usually sigmoid or softmax).

Proposition 1. *$\mathcal{L}_1$ loss applied to the probability estimates $\hat{p}(\mathbf{y}|\mathbf{x})$ leads to minimisation of expected misclassification probability (as opposed to maximisation of fully correct labelling given by the log loss). Similarly $\mathcal{L}_2$ minimises the same factor, but regularised with a half of expected squared L_2 norm of the predictions probability estimates.*

Proof. In K-class classification dependent variables are vectors $\mathbf{y}_i \in \{0,1\}^K$ with $\mathrm{L}_1(\mathbf{y}_i) = 1$, thus using notation $\mathbf{p}_i = \hat{p}(\mathbf{y}|\mathbf{x}_i)$

$$\begin{aligned}\mathcal{L}_1 &= \tfrac{1}{N}\sum_i \sum_j |\mathbf{p}_i^{(j)} - \mathbf{y}_i^{(j)}| = \tfrac{1}{N}\sum_i \left[\sum_j \mathbf{y}_i^{(j)}(1-\mathbf{p}_i^{(j)}) + (1-\mathbf{y}_i^{(j)})\mathbf{p}_i^{(j)}\right] \\ &= \tfrac{1}{N}\sum_i \left[\sum_j \mathbf{y}_i^{(j)} - 2\sum_j \mathbf{y}_i^{(j)}\mathbf{p}_i^{(j)} + \sum_j \mathbf{p}_i^{(j)}\right] = 2 - 2\tfrac{1}{N}\sum_i \left[\sum_j \mathbf{y}_i^{(j)}\mathbf{p}_i^{(j)}\right].\end{aligned}$$

Consequently if we sample label according to $\mathbf{p}_i$ then probability that it actually matches one hot encoded label in $\mathbf{y}_i$ equals $P(\hat{l} = l|\hat{l} \sim \mathbf{p}_i, l \sim \mathbf{y}_i) = \sum_j \mathbf{y}_i^{(j)}\mathbf{p}_i^{(j)}$, and consequently

$$\mathcal{L}_1 = 2 - 2\tfrac{1}{N}\sum_i \left[\sum_j \mathbf{y}_i^{(j)}\mathbf{p}_i^{(j)}\right] \approx -2\mathbb{E}_{P(\mathbf{x},\mathbf{y})}\left[P(\hat{l} = l|\hat{l} \sim \mathbf{p}_i, l \sim \mathbf{y}_i)\right] + \text{const.}$$

Analogously for $\mathcal{L}_2$,

$$\begin{aligned}\mathcal{L}_2 &= -2\tfrac{1}{N}\sum_i \left[\sum_j \mathbf{y}_i^{(j)}\mathbf{p}_i^{(j)}\right] + \tfrac{1}{N}\sum_i \mathrm{L}_2(\mathbf{y}_i)^2 + \tfrac{1}{N}\sum_i \mathrm{L}_2(\mathbf{p}_i)^2 \\ &\approx -2\mathbb{E}_{P(\mathbf{x},\mathbf{y})}\left[P(\hat{l} = l|\hat{l} \sim \mathbf{p}_i, l \sim \mathbf{y}_i)\right] + \mathbb{E}_{P(\mathbf{x},\mathbf{y})}[\mathrm{L}_2(\mathbf{p}_i)^2] + \text{const.} \quad \square\end{aligned}$$

For this reason we refer to these losses as *expectation loss* and *regularised expectation loss* respectively. One could expect that this should lead to higher robustness to the outliers/noise, as we try to maximise the expected probability of good classification as opposed to the probability of completely correct labelling (which log loss does). Indeed, as we show in the experimental section – this property is true for all losses sharing connection with *expectation losses*.

So why is using these two loss functions unpopular? Is there anything fundamentally wrong with this formulation from the mathematical perspective? While the following observation is not definitive, it shows an insight into what might be the issue causing slow convergence of such methods.

Proposition 2. *$\mathcal{L}_1$, $\mathcal{L}_2$ losses applied to probabilities estimates coming from sigmoid (or softmax) have non-monotonic partial derivatives wrt. to the output of the final layer (and the loss is not convex nor concave wrt. to last layer weights). Furthermore, they vanish in both infinities, which slows down learning of heavily misclassified examples.*

Proof. Let us denote sigmoid activation as $\sigma(x) = (1 + e^{-x})^{-1}$ and, without loss of generality, compute partial derivative of $\mathcal{L}_1$ when network is presented with x_p with positive label. Let o_p denote the output activation for this sample.

$$\frac{\partial(\mathcal{L}_1 \circ \sigma)}{\partial o}(o_p) = \frac{\partial}{\partial o}\left(|1 - (1 + e^{-o})^{-1}|\right)(o_p) = -\frac{e^{-o_p}}{(e^{-o_p} + 1)^2}$$

$$\lim_{o \to -\infty} -\frac{e^{-o}}{(e^{-o} + 1)^2} = 0 = \lim_{o \to \infty} -\frac{e^{-o}}{(e^{-o} + 1)^2},$$

while at the same time $-\frac{e^0}{(e^0+1)^2} = -\frac{1}{4} < 0$, completing the proof of both non-monotonicity as well as the fact it vanishes when point is heavily misclassified. Lack of convexity comes from the same argument since second derivative wrt. to any weight in the final layer of the model changes sign (as it is equivalent to first derivative being non-monotonic). This comes directly from the above computations and the fact that $o_p = \langle \mathbf{w}, \mathbf{h}_p \rangle + b$ for some internal activation $\mathbf{h}_p$, layer weights $\mathbf{w}$ and layer bias b. In a natural way this is true even if we do not have any hidden layers (model is linear). Proofs for $\mathcal{L}_2$ and softmax are completely analogous. □

Given this negative result, it seems natural to ask whether a similar property can be proven to show which loss functions should lead to *fast* convergence. It seems like the answer is again positive, however based on the well known deep learning hypothesis that deep models learn well when dealing with piece-wise linear functions. An interesting phenomenon in classification based on neural networks is that even in a deep linear model or rectifier network the top layer is often non-linear, as it uses softmax or sigmoid activation to produce probability estimates. Once this is introduced, also the partial derivatives stop being piece-wise linear. We believe that one can achieve faster, better convergence when we ensure that architecture together with loss function, produces a piecewise linear partial derivatives (but not constant) wrt. to final layer activations, especially while using first order optimisation methods. This property is true only for $\mathcal{L}_2$ loss and squared hinge loss (see Figure 1) among all considered ones in this paper.

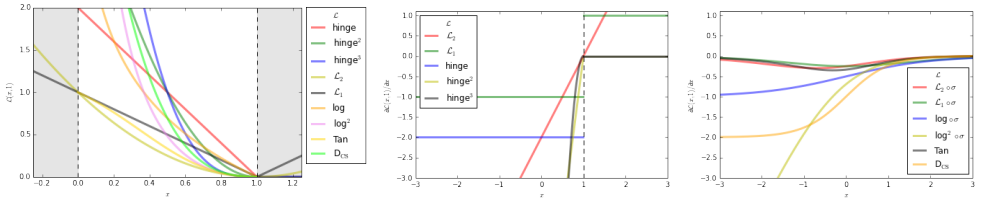

Figure 1. Left: Visualisation of analysed losses as functions of activation on positive sample. Middle: Visualisation of partial derivatives wrt. to output neuron for losses based on linear output. Right: Visualisation of partial derivatives wrt. to output neuron for losses based on probability estimates.

Finally we show relation between Cauchy-Schwarz Divergence loss and the log loss, justifying its introduction as an objective for neural nets.

Proposition 3. *Cauchy-Schwarz Divergence loss is equivalent to cross entropy loss regularised with half of expected Renyi's quadratic entropy of the predictions.*

Proof. Using the fact that $\forall_i \exists!_j : \mathbf{y}_i^{(j)} = 1$ we get that $\log \sum_j \mathbf{p}_i^{(j)} \mathbf{y}_i^{(j)} = \sum_j \mathbf{y}_i^{(j)} \log \mathbf{p}_i^{(j)}$ as well as $\|\mathbf{y}_i\|_2 = 1$, consequently

$$\begin{aligned} D_{\text{CS}} &= -\tfrac{1}{N} \sum_i \log \frac{\sum_j \mathbf{p}_i^{(j)} \mathbf{y}_i^{(j)}}{\|\mathbf{p}_i\|_2 \|\mathbf{y}_i\|_2} = -\tfrac{1}{N} \sum_i \log \sum_j \mathbf{p}_i^{(j)} \mathbf{y}_i^{(j)} + \tfrac{1}{N} \sum_i \log \|\mathbf{p}_i\|_2 \|\mathbf{y}_i\|_2 \\ &= -\tfrac{1}{N} \sum_i \sum_j \mathbf{y}_i^{(j)} \log \mathbf{p}_i^{(j)} + \tfrac{1}{2N} \sum_i \log \|\mathbf{p}_i\|_2^2 \approx \mathcal{L}_{\log} + \tfrac{1}{2} \mathbb{E}_{P(\mathbf{x},\mathbf{y})}[H_2(\mathbf{p}_i)] \end{aligned}$$

□

3. Experiments

We begin the experimental section with two simple 2D toy datasets. The first one is checkerboard – 4 class classification problem where [-1,1] square is divided into 64 small squares with cyclic class assignment. The second one, spiral, is a 4 class generalisation of the well known 2 spirals dataset. Both datasets have 800 training and 800 testing samples. We train rectifier neural network having from 0 to 5 hidden layers with 200 units in each of them. Training is performed using Adam [6] with learning rate of 0.00003 for 60,000 iterations with batch size of 50 samples. In these simple problems one can distinguish (Figure 2) two groups of losses – one able to fit to our very dense, low-dimensional data and one struggling to reduce error to 0. The second group consists of $\mathcal{L}_1$, Chebyshev, Tanimoto and expectation loss. This division becomes clear once we build a relatively deep model (5 hidden layers), while for shallow ones this distinction is not very clear (3 hidden layers) or is even completely lost (1 hidden layer or linear model). To further confirm the lack of ability to easily overfit we also ran an experiment in which we tried to fit 800 samples

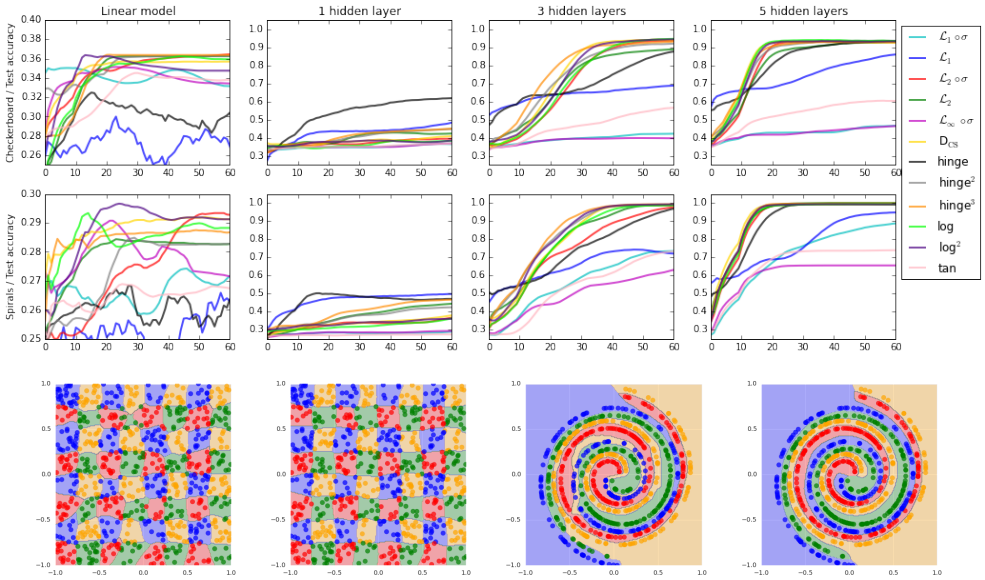

Figure 2. Top row: Learning curves for toy datasets. Bottom row: examples of decision boundaries, from left: $\mathcal{L}_1$ loss, log loss, $\mathcal{L}_1 \circ \sigma$ loss, hinge^2 loss.

from uniform distribution over $[-1, 1]$ with randomly assigned 4 labels and achieved analogous partitioning.

During following, real data-based experiments, we focus on further investigation of loss functions properties emerging after application to deep models, as well as characteristics of the created models. In particular, we show that lack of ability to reduce training error to 0 is often correlated with robustness to various types of noise (despite not underfitting the data).

Let us now proceed with one of the most common datasets used in deep learning community – MNIST [11]. We train network consisting from 0 to 5 hidden layers, each followed by ReLU activation function and dropout [12] with 50% probability. Each hidden layer consists of 512 neurons, and whole model is trained using Adam [6] with learning rate of 0.00003 for 100,000 iterations using batch size of 100 samples. There are few interesting findings, visible on Figure 3. First, results obtained for a linear model (lack of hidden layers) are qualitatively different from all the remaining ones. For example, using regularised expectation loss leads to the strongest model in terms of both training accuracy and generalisation capabilities, while the same loss function is far from being the best one once we introduce non-linearities. This shows two important things: first – observations and conclusions drawn from linear models do not seem to transfer to deep nets, and second – there seems to be an interesting co-dependence between learning dynamics coming from training rectifier nets and loss functions used. As a side note, 93% testing accuracy, obtained by $\mathcal{L}_2 \circ \sigma$ and D_{CS}, is a very strong result on MNIST using linear model without any data augmentation or model regularisation.

Second interesting observation regards the speed of learning. It appears that

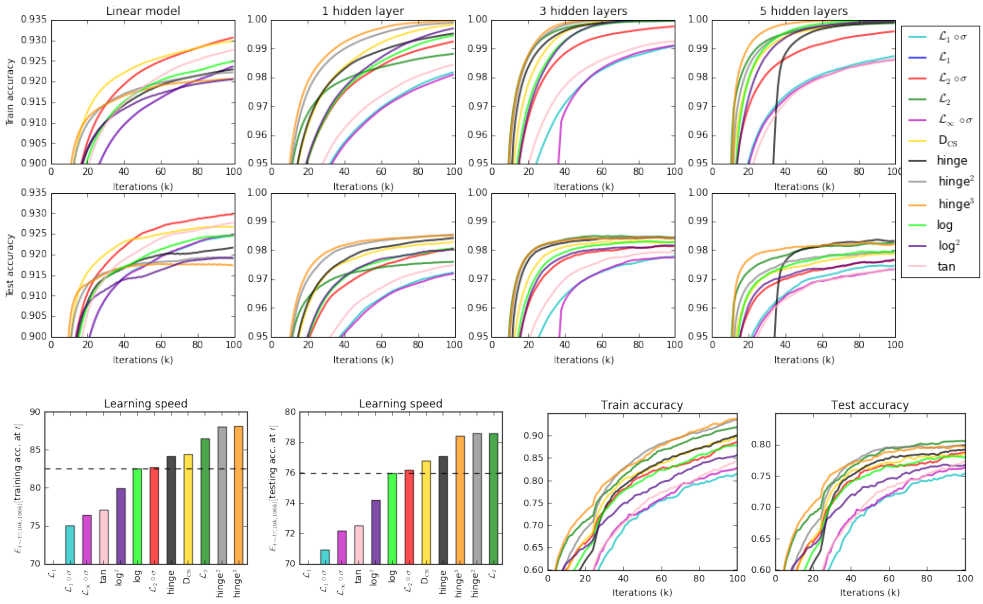

Figure 3. Top two rows: learning curves for MNIST dataset. Bottom row: (left) speed of learning expressed as expected training/testing accuracy when we sample iteration uniformly between 10k and 100k; (right) learning curves for CIFAR10 dataset.

(apart from linear models) hinge^2 and hinge^3 losses are consistently the fastest in training, and once we have enough hidden layers (basically more than 1) also $\mathcal{L}_2$. This matches our theoretical analysis of these losses in the previous section. At the same time both expectation losses are much slower to train, which we believe to be a result of their vanishing partial derivatives in heavily misclassified points (Proposition 2). It is important to notice that while higher order hinge losses (especially 2$^{\text{nd}}$) actually help in terms of both speed and final performance, the same property does not hold for higher order log losses. One possible explanation is that taking a square of log loss only reduces model's certainty in classification (since any number between 0 and 1 taken to 2$^{\text{nd}}$ power decreases), while for hinge losses the metric used for penalising margin-outliers is changed, and both L_1 metric (leading to hinge) as well as any other L_p norm (leading to hinge^p) make perfect sense.

Third remark is that pure $\mathcal{L}_1$ does not learn at all (ending up with 20% accuracy) due to causing serious "jumps" in the model because of its partial derivatives wrt. to net output always being either -1 or 1. Consequently, even after classifying a point correctly, we are still heavily penalised for it, while with losses like $\mathcal{L}_2$ the closer we are to the correct classification – the smaller the penalty is.

Finally, in terms of generalisation capabilities margin-based losses seem to outperform the remaining families. One could argue that this is just a result of lack of regularisation in the rest of the losses, however we underline that all the analysed networks use strong dropout to counter the overfitting problem, and that typical L_1 or L_2 regularisation penalties do not work well in deep networks.

For CIFAR10 dataset we used a simple convnet, consisting of 3 layers of convolutions, each of size 5x5 and 64 filters, with ReLU activation functions, batch-normalisation and pooling operations in between them (max pooling after first layer and then two average poolings, all 3x3 with stride 2), followed by a single fully connected hidden layer with 128 ReLU neurons, and final softmax layer with 10 neurons. As one can see in Figure 3, despite completely different architecture than before, we obtain very similar results – higher order margin losses lead to faster training and significantly stronger models. Quite surprisingly – $\mathcal{L}_2$ loss also exhibits similar property. Expectation losses again learn much slower (with the regularised one – training at the level of log loss and unregularised – significantly worse). We would like to underline that this is a very simple architecture, far from the state-of-the art models for CIFAR10, however we wish to avoid using architectures which are heavily overfitted to the log loss. Furthermore, the aim of this paper is not to provide any state-of-the-art models, but rather to characterise effects of loss functions on deep networks.

As the final interesting result in these experiments, we notice that Cauchy-Schwarz Divergence as the optimisation criterion seems to be a consistently better choice than log loss. It performs equally well or better on both MNIST and CIFAR10 in terms of both learning speed and the final performance. At the same time this information theoretic measure is very rarely used in DL community, and rather exploited in shallow learning (for both classification [10] and clustering [13]).

Now we focus on the impact these losses have on noise robustness of the deep nets. We start by performing the following experiment on previously trained MNIST classifiers: we add noise sampled from $\mathcal{N}(0, \epsilon\mathbf{I})$ to each $\mathbf{x}_i$ and observe how quickly (in terms of growing ϵ) network's training accuracy drops (Figure 4). The first crucial observation is that both expectation losses perform very well in terms of input noise robustness. We believe that this is a consequence of what Proposition 1 showed about their probabilistic interpretation – that they lead to minimisation of the expected misclassification, which is less biased towards outliers than log loss (or other losses that focus on maximisation of probability of correct labelling of all samples at the same time). For log loss a single heavily misclassified point has an enormous impact on the overall error surface, while for these two losses – it is minor. Secondly, margin based losses also perform well on this test, usually slightly worse than the expectation losses, but still better than log loss. This shows that despite no longer maximising the misclassification margin while being used in deep nets – they still share some characteristics with their linear origins (SVM). In another, similar experiment, we focus on the generalisation capabilities of the networks trained with increasing amount of label noise in the training set (Figure 4) and obtain analogous results, showing that robustness to the noise of expectation and margin losses is high for both input and label noise for deep nets, while again – slightly different results are obtained for linear models, where log loss is more robust than the margin-based ones. What is even more interesting, a completely non-standard loss function – *Tanimoto loss* – performs extremely well on this task. We believe that its exact analysis is one of the important future research directions.

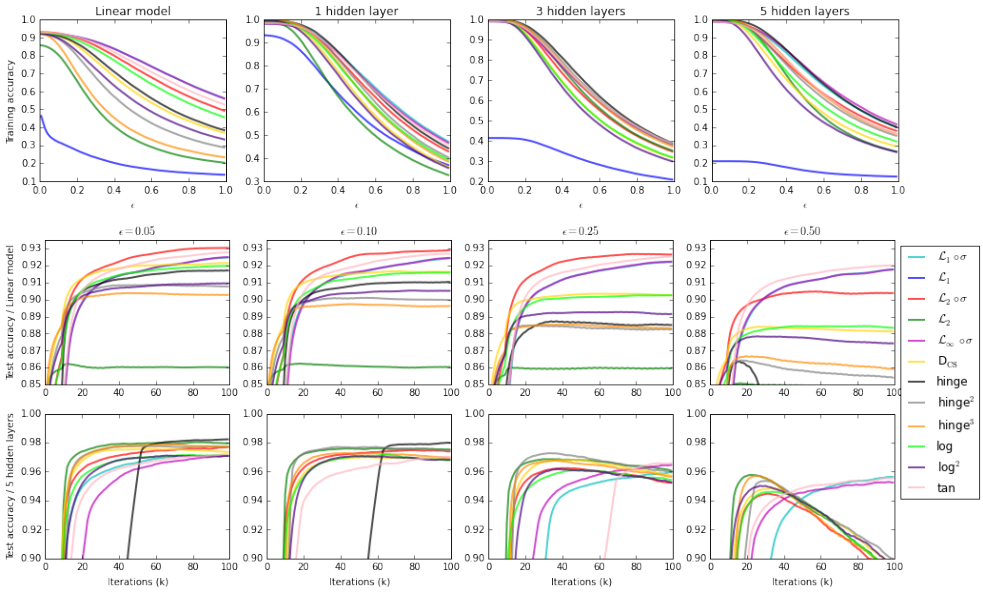

Figure 4. Top row: Training accuracy curves for the MNIST trained models, when presented with training examples with added noise from $\mathcal{N}(0, \epsilon\mathbf{I})$, plotted as a function of ϵ. Middle and bottom rows: Testing accuracy curves for the MNSIT experiment with ϵ of training labels changed, plotted as a function of training iteration. If $\mathcal{L}_1 \circ \sigma$ is not visible, it is almost perfectly overlapped by $\mathcal{L}_\infty \circ \sigma$.

4. Conclusions

This paper provides basic analysis of effects the choice of the classification loss function has on deep neural networks training as well as their final characteristics. We believe the obtained results will lead to a wider adoption of various losses in DL work – where up till now log loss is unquestionable favourite.

In the theoretical section we show that, surprisingly, losses which are believed to be applicable mostly to regression, have a valid probabilistic interpretation when applied to deep network-based classifiers. We also provide theoretical arguments explaining why using them might lead to slower training, which might be one of the reasons DL practitioners have not yet exploited this path. Our experiments lead to two crucial conclusions. First, that intuitions drawn from linear models rarely transfer to highly-nonlinear deep networks. Second, that depending on the application of the deep model – losses other than log loss are preferable. In particular, for purely accuracy focused research, squared hinge loss seems to be a better choice at it converges faster as well as provides better performance. It is also more robust to noise in the training set labelling and slightly more robust to noise in the input space. However, if one works with highly noised dataset (both input and output spaces) – the expectation losses

described in detail in this paper – seem to be the best choice, both from theoretical and empirical perspective.

At the same time this topic is far from being exhausted, with a large amount of possible paths to follow and questions to be answered. In particular, non-classical loss functions such as Tanimoto loss and Cauchy-Schwarz Divergence are worth further investigation.

5. References

[1] Larochelle H., Bengio Y., Louradour J., Lamblin P., *Exploring strategies for training deep neural networks.* Journal of Machine Learning Research, 2009, 10 (Jan), pp. 1–40.

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

[3] Oord A.v.d., Dieleman S., Zen H., Simonyan K., Vinyals O., Graves A., Kalchbrenner N., Senior A., Kavukcuoglu K., *Wavenet: A generative model for raw audio.* arXiv preprint arXiv:1609.03499, 2016.

[4] Silver D., Huang A., Maddison C.J., Guez A., Sifre L., Van Den Driessche G., Schrittwieser J., Antonoglou I., Panneershelvam V., Lanctot M., et al., *Mastering the game of go with deep neural networks and tree search.* Nature, 2016, 529 (7587), pp. 484–489.

[5] Clevert D.A., Unterthiner T., Hochreiter S., *Fast and accurate deep network learning by exponential linear units (elus).* arXiv preprint arXiv:1511.07289, 2015.

[6] Kingma D., Ba J., *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.

[7] Tang Y., *Deep learning using linear support vector machines.* arXiv preprint arXiv:1306.0239, 2013.

[8] Lee C.Y., Xie S., Gallagher P., Zhang Z., Tu Z., *Deeply-supervised nets.* In: *AISTATS.* vol. 2., 2015, pp. 6.

[9] Choromanska A., Henaff M., Mathieu M., Arous G.B., LeCun Y., *The loss surfaces of multilayer networks.* In: *AISTATS*, 2015.

[10] Czarnecki W.M., Jozefowicz R., Tabor J., *Maximum entropy linear manifold for learning discriminative low-dimensional representation.* In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2015, pp. 52–67.

[11] LeCun Y., Cortes C., Burges C.J., *The mnist database of handwritten digits*, 1998.

[12] Srivastava N., Hinton G.E., Krizhevsky A., Sutskever I., Salakhutdinov R., *Dropout: a simple way to prevent neural networks from overfitting.* Journal of Machine Learning Research, 2014, 15 (1), pp. 1929–1958.

[13] Principe J.C., Xu D., Fisher J., *Information theoretic learning.* Unsupervised adaptive filtering, 2000, 1, pp. 265–319.