MLP Coursework 1: Activation Functions

s1740192

Abstract

Deep learning is a fast-paced research field where several models, training schemes and applications are being actively investigated. Unlike traditional machine learning systems, deep neural networks are able to automatically learn expressive representations of the data. The choice of activation functions, number of layers and initialization schemes directly affect this capability, and therefore its analysis is of great importance. The purpose of this study is threefold. First, we present several activation functions, ranging from sigmoid to more recent developments such as Exponential Linear Units (ELU) or Scaled Exponential Linear Units (SELU), and measure their impact on a handwritten digit recognition task for a fixed architecture. Second, we evaluate the behavior of these activation functions in the context of deeper models and, finally, we extend the analysis to multiple initialization schemes. We find that the new developments in both activation functions and initialization methods lead to deeper models that are easier to train.

1. Introduction

One field that has gained a tremendous amount of attention among machine learning researchers and practitioners alike is that of Deep Learning, or deep neural networks (LeCun et al., 2015). Indeed, deep neural networks seem to excel at perception tasks and have revolutionized several disciplines, ranging from computer vision, natural language processing or time series analysis. In essence, these models are comprised of simple nodes arranged in a network, whose topology is often problem-specific and can be interpreted as the prior belief of the modeller on how the data should be perceived. For instance, feedforward neural networks are a particular type of networks, where each node, after receiving information from a previous layer, applies an affine transformation followed by an activation function (nonlinear map) and propagates the information to the next layer. One property of deep learning methods is that they are able to automatically learn more abstract, and hence expressive, representations of the input data at each layer, unlike traditional machine learning systems that require careful feature engineering in order to find a suitable representation that can be fed to the next learning module (Goodfellow et al., 2016). The nonlinear transformations and the depth of a network directly affect this capability and are therefore important design aspects.

In this work, we are interested in measuring the impact of different activation functions on handwritten digit recognition. For this purpose, in Section 2, we first present the activation functions and, in Section 3, we analyze the performance of these functions for a fixed architecture. We expand this analysis by considering deeper models in Section 4 and also take into account the effect of different initialization strategies in Section 5. In particular, we assess the performance of several feedforward neural networks with respect to the cross entropy error and accuracy on the Modified National Institute of Standards and Technology (MNIST) digit classification dataset, a 10-class problem. In this dataset, each sample is a 28×28 pixel grayscale image of a handwritten digit, which has been stored as a 784 dimensional vector. There are $60,000$ examples as training data and $10,000$ as test data. The original training data is further split into two parts: $50,000$ and $10,000$, with the first being used to train our models and the second as a validation set. The reported performances correspond to these two parts.

2. Activation functions

Traditional neural networks used to employ a sigmoid as the nonlinear activation function. This is a real-valued differentiable function that monotonically increases from zero to one:

$$\text{sigmoid}(x) = \frac{e^x}{e^x + 1} = \frac{1}{1 + e^{-x}}, \tag{1}$$

and whose derivative can be shown to be:

$$\frac{d}{dx}\,\text{sigmoid}(x) = \text{sigmoid}(x) \cdot (1\,\text{sigmoid}(x))\,. \tag{2}$$

However, neural networks using sigmoidal activations were observed to suffer from the vanishing gradient problem due to the saturation regions – a consequence of mapping the extended real line onto a small output range. The vanishing gradients, i.e. gradients of nearly zero, cause slow optimization convergence, and hence learning, possibly leading to poor local minima (Maas et al., 2013). A different activation function, Rectified Linear Unit (ReLU), was then shown to improve restricted Boltzmann machines (Nair & Hinton, 2010) and its use in discriminative deep neural networks was advocated by (Glorot et al., 2011). The ReLU tries to address the problem by defining a linear relationship (identity) when the input is positive (active) and discarding the information of the input otherwise:

$$\text{relu}(x) = \max(0, x), \tag{3}$$

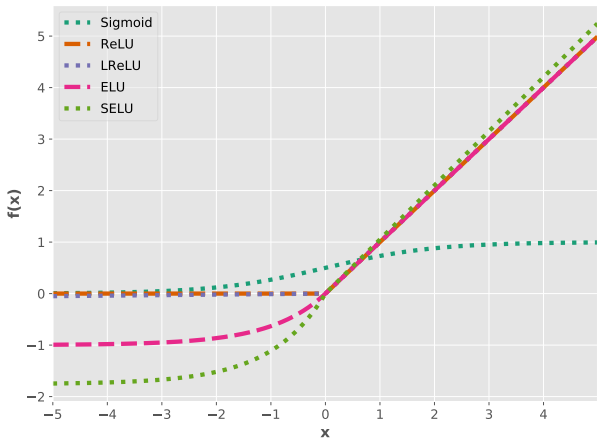

Figure 1. Nonlinear activation function curves: sigmoid, ReLU, LReLU ($\alpha = 0.01$), ELU ($\alpha = 1$) and SELU ($\alpha \simeq 1.6733$ and $\lambda \simeq 1.0507$).

$$\frac{d}{dx}\text{relu}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{4}$$

Notice that the gradient of active units cannot vanish since the output does not saturate. On the other hand, the gradient is zero when the unit is inactive, which has been argued by (Maas et al., 2013) as being a potential drawback during training because a unit that does not activate initially will remain inactive. Consequently, the authors proposed a modification to this unit by allowing non-zero gradients when inactive. The Leaky ReLU (LReLU) is as follows:

$$\text{lrelu}(x) = \begin{cases} \alpha x & \text{if } x \leq 0 \\ x & \text{if } x > 0, \end{cases} \tag{5}$$

with the corresponding derivative being:

$$\frac{d}{dx}\text{lrelu}(x) = \begin{cases} \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0, \end{cases} \tag{6}$$

where α is a small positive constant, e.g. $\alpha = 0.01$.

Recently, (Clevert et al., 2015) have reiterated that the vanishing gradient problem can be mitigated by using the identity relationship for positive inputs and argue that it is important to have negative values when the unit is inactive. In particular, the bias present in non-zero mean activation functions has as an impact during learning and correcting this bias, by pushing it towards zero, can speed up this process. The authors further assert that LReLUs are not robust to noise during the inactive state because there is no negative saturation. This in turn is critical because it allows the inactive state to be uninformative, i.e. invariance/robustness of the output for small variations of the input. The authors propose the Exponential Linear Unit (ELU):

$$\text{elu}(x) = \begin{cases} \alpha\,(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0, \end{cases} \tag{7}$$

whose derivative is:

$$\frac{d}{dx}\text{elu}(x) = \begin{cases} \text{elu}(x) + \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0, \end{cases} \tag{8}$$

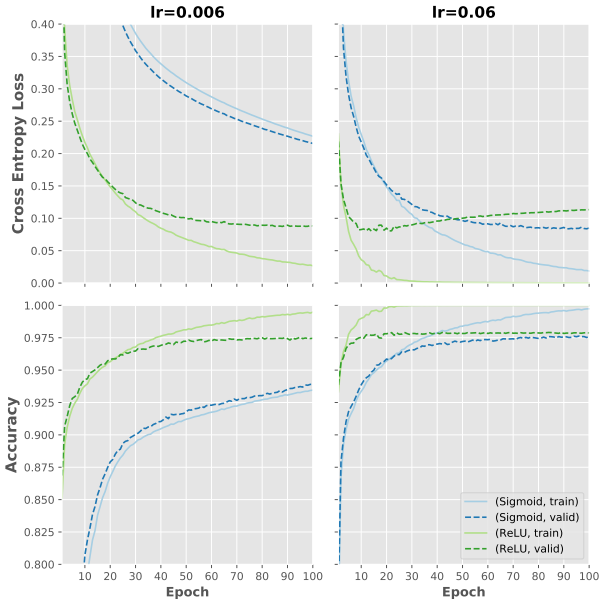

Figure 2. Cross entropy loss and accuracy curves of neural networks with 2 hidden layers (100 units each) trained for 100 epochs on MNIST data. Weights initialized with Glorot Uniform scheme. *Left*: Optimal learning rate for ReLU (*green*). *Right*: Optimal learning rate for sigmoid (*blue*).

where α is a positive constant that controls the negative saturation, e.g. $\alpha = 1$. In addition, (Klambauer et al., 2017) claim that an appropriate activation function should also have a derivative larger than one to increase the variance for positive inputs. The Scaled Exponential Linear Unit (SELU) has the following form:

$$\text{selu}(x) = \lambda \begin{cases} \alpha\,(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0, \end{cases} \tag{9}$$

$$\frac{d}{dx}\text{selu}(x) = \begin{cases} \text{selu}(x) + \lambda\alpha & \text{if } x \leq 0 \\ \lambda & \text{if } x > 0, \end{cases} \tag{10}$$

where α and λ are positive scalars. For instance, following the theoretical argument presented by the authors $\alpha \simeq 1.6733$ and $\lambda \simeq 1.0507$. Figure 1 shows the curves for all the activation functions considered in this work.

3. Experimental comparison of activation functions

Before proceeding with the analysis of our results, we note that all experiments in this section have been conducted using feedforward neural networks of two hidden layers, with 100 units per hidden layer (784/100/100/10). The network parameters are learned via backpropagation using Stochastic Gradient Descent (SGD) with a batch size of 50 for a total of 100 epochs (no early stopping). The learning rates ranged from 0.001 to 0.2, for a total of 20 different values. In particular, these rates were generated by taking the product $(1, 2, ..., 9)\times(10^{-3}, 10^{-2}, 10^{-1})$ and trucating the maximum value at 0.2. In addition, the weights are initialized using the Glorot Uniform scheme (Glorot & Bengio, 2010), i.e. these are drawn from a Uniform distribution with variance $2/(n_{in}+n_{out})$, where n_{in} and n_{out} denote the number

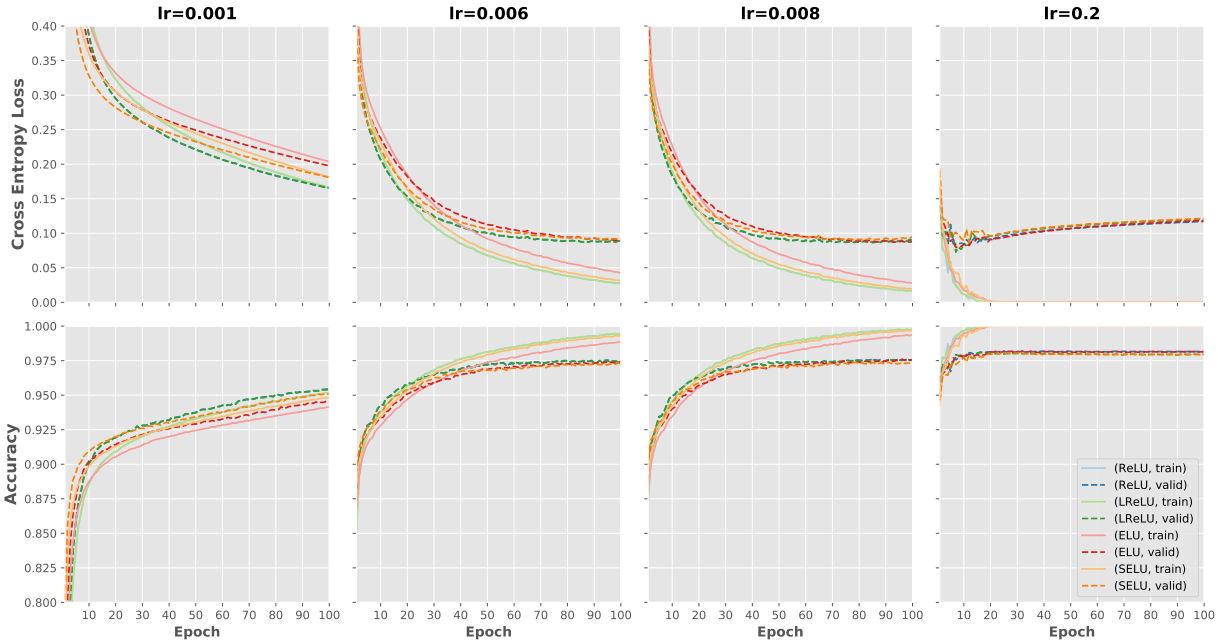

Figure 3. Cross entropy loss and accuracy curves of neural networks with 2 hidden layers (100 units each) trained for 100 epochs on MNIST data. Weights initialized with Glorot Uniform. From *left* to *right*, we have the following learning rates (SGD): 0.001 (lowest); 0.006 (optimal for ReLU, LReLU and SELU); 0.008 (optimal for ELU); 0.2 (highest).

of incoming and outgoing connections for a particular layer of neurons, respectively.

We begin our set of experiments by comparing the performance of sigmoidal neural networks with others that use ReLU nonlinearities. A summary of the results can be found in Table 1, where we report the validation error (cross entropy loss) and accuracy observed after training the networks for 100 epochs. In order to determine the optimal learning rate for each activation function, we decide to select the point at which the validation error stops decreasing. We believe this selection is justified because we are unable to perform early stopping or any other form of regularization, but still are interested in networks that are not overfitted. In particular, we note that the effect of overfitting occurs predominantly for higher learning rates, but, if low learning rates are to be chosen, then underfitting might also occur. Returning to the comparison between sigmoid and ReLU, we observe that, for the same learning rate, the former is not as accurate as the latter and that the optimal learning rate for ReLU neural networks is 0.006, whereas for a sigmoidal neural network of similar architecture we have 0.06. This tenfold increase can perhaps be explained by the vanishing gradient problem mentioned in the previous section since the optimization updates are scaled by the learning rates, acting as a possible countermeasure against small gradients.

A more thorough analysis can be made by plotting the performance curves for both activation functions at their optimal points. Indeed, Figure 2 shows that ReLU performs better than sigmoid since it allows a significantly faster convergence. Specifically, we observe that, for a learning rate of 0.006, the sigmoid shows signs of underfitting, whereas the ReLU managed to converge to almost the final state within 50 epochs. On the other hand, for 0.06, the sigmoid takes almost all 100 epochs to find an acceptable solution, while the ReLU is capable of reaching such state within the first 20 epochs.

In the remainder of this section, we compare the behavior of the other nonlinear activation functions with ReLU as to ascertain whether the introduction of these recent variants translate into a tangible boost in performance. Before proceeding with this analysis, we would like to point that a careful investigation of this matter also requires the evaluation of these systems under multiple restarts and on several recognition tasks involving different levels of complexity. However, such study is beyond the scope of this work due to time and computational constraints. Thus, in the analysis that follows, we must take this into account as to not draw premature interpretations.

In order to compare the performance of LReLU, ELU and SELU, we proceed as before: for each activation function, we train multiple networks to determine the optimal learning rate. These results can again be found in Table 1. We find that the optimal rates for LReLU, ELU and SELU are 0.006, 0.008 and 0.006, respectively. This is to a certain degree reassuring as we were expecting these learning rates to be of equal or lower order to that of ReLU. Perhaps surprisingly, we also notice that the difference in performance between these variants and ReLU is not as expressive as to when we compared sigmoid to ReLU. In fact, it is difficult to state confidently which activation function might perform the best without investigating this matter further.

Figure 3 shows the learning curves of these different systems for the lowest (0.001), optimal (0.006, 0.008) and highest (0.2) rates. For the lowest learning rate, we observe that all systems are underfitted and that ReLU and LReLU seem to have a slight advantage over ELU and SELU, but

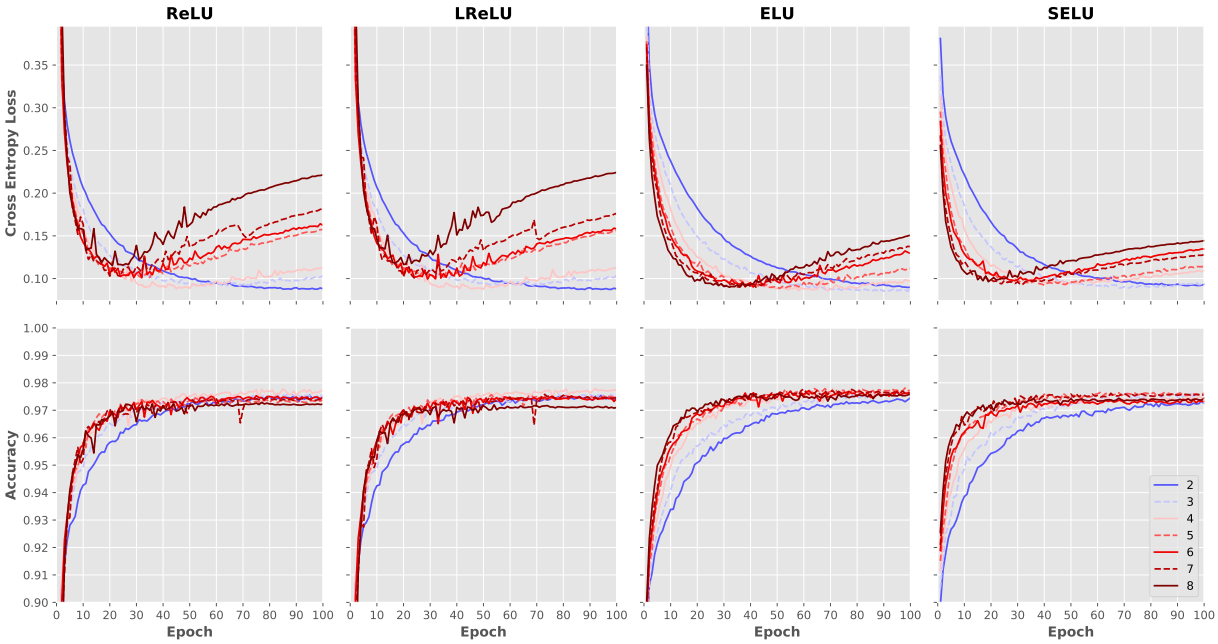

Figure 4. Validation cross entropy loss and accuracy curves of deep neural networks, ranging from 2 to 8 hidden layers (100 units each), trained for 100 epochs on the MNIST task. The learning rate (SGD) was fixed at 0.006.

it is unclear whether this trend can be extended to the general case. On the other hand, for the highest learning rate, all systems show signs of overfitting, reaching the lowest validation error within the first 20 epochs. SELU seems to be the most sensitive to high learning rates, but again it is uncertain if this holds in general. Perhaps the most interesting plots are those related to what we defined to be the optimal learning rates. For these both, ReLU and LReLU show signs of faster convergence, followed by SELU and then ELU. In terms of accuracy, SELU performs the worst, but only marginally. Unlike sigmoids, some of these differences are incredibly subtle and, consequently, we should be cautious in our inferences. However, at this point, we believe that it might be safe to assert that ReLU and LReLU behave similarly as there is a close match between their performances curves – note that (Maas et al., 2013) have reached a similar conclusion.

4. Deep neural network experiments

First, we would like to motivate the experiments that follow by building on the results from the previous section. We observed that, among the activation functions under analysis, sigmoids lead to the worst outcome. This is unsurprising as it can be attributed to the vanishing gradient problem. On the other hand, we were not able to confidently state which activation function is the best performer. ReLU and LReLU achieved slightly better results on the previous section, but the difference between these and ELU/SELU was not as expressive as the one observed between ReLU and sigmoid. Additionally, we should note that while both ELU and SELU have been advocated by their authors as being suitable for deeper models, the neural networks in the previous section were not particularly deep. Therefore, in this section, we do not limit ourselves to the analysis of one particular activation function. Instead, we decide to evaluate the behavior of all these functions in the context of deeper models. In particular, we consider networks that range from 2 to 8 hidden layers, using 100 units per hidden layer. As before, we use SGD with batches of size 50 for a total of 100 epochs. We initialize the weights using the Glorot Uniform scheme and we now fix the learning rate at 0.006, which, in the previous section, was found to yield good results for most activation functions.

Figure 4 shows the performance curves of these deeper models using ReLU and variants. Moving from left to right, perhaps the most noticeable feature is that ReLU and LReLU networks continue to behave similarly, even for deeper architectures. Particularly, for both functions, as the model increases its depth, the performance curves become noisier, indicating a greater sensitivity to updates. This effectively means that deeper models are more difficult to train. As we would expect, deeper models also begin to overfit sooner because there are more free parameters. In addition, the optimal depth for these settings seems to be 4 hidden layers, but if we were to choose this model, we should perform early stopping somewhere between epochs 40 to 60, or apply other forms of regularization, as there are signs of overfitting by the end of 100 epochs. In comparison to ReLU and LReLU, another salient feature is that ELU and SELU indeed seem more suitable for deeper models. The performance curves are noticeably less noisy which, in turn, means that the models are easier to train. Deeper models also appear to be able to converge faster, while taking longer to overfit. For instance, this is clearly observable if we consider the model with 8 hidden layers. In this case, both ELU and SELU lead to a better solution in terms of error and accuracy, as it can be seen during the first 20 epochs. This behavior does not occur for ReLU/LReLU, which can perhaps be attributed to the lack of bias correction or lack of robustness of the inactive state mentioned by (Clevert

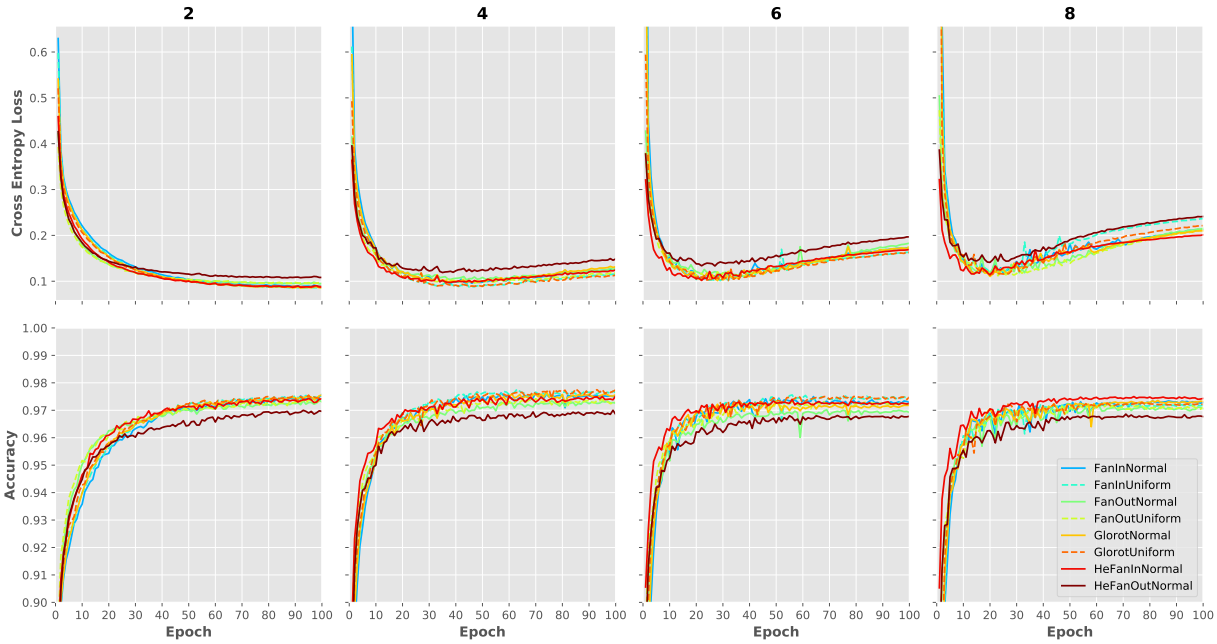

Figure 5. Validation cross entropy loss and accuracy curves of deep neural networks using ReLU, ranging from 2 to 8 hidden layers (100 units each), trained for 100 epochs on the MNIST task. The learning rate (SGD) was fixed at 0.006. Different initialization schemes are tested.

et al., 2015).

In comparison to SELU, we observe that networks using the ELU activation function were eventually able to reach marginally better solutions and those with 6 or more hidden layers seem to take longer to start overfitting. Deep architectures using SELU show however signs of faster convergence (see the first 10 to 20 epochs). For this relatively simple task of handwritten digit classification, the architecture of 5 hidden layers seems to lead to one of the best performances for both activation functions, but this does not necessarily mean that we should choose this architecture over the others. In fact, if we apply the Occam's razor principle, we note that, after convergence, the differences in performance for most architectures do not appear to be significant. Therefore, a good compromise between performance and complexity is reached if we select the networks with 3 hidden layers.

5. Experimental comparison of initializations

Unlike the previous sections, where we used the Glorot Uniform as our sole method of weight initialization, we now investigate the effect of different initialization strategies on handwritten digit recognition. For this purpose, we have again considered all activation functions, but we only present the results obtained for ReLU and SELU due to space constraints. We found, however, that ReLU and LReLU behave similarly and that, apart from certain fluctuations, many of the conclusions that will be drawn for SELU will also hold for ELU. In addition, instead of focusing on one particular architecture, we analyze the effect of different initialization schemes for networks of increasing depth, i.e. models that range from 2 to 8 hidden layers with 100 units each. As in the previous section, we use SGD with a learning rate of 0.006 and batches of size 50 for a total of 100 epochs.

Regarding weight initialization, we explore a total of 8 methodologies. In particular, we are interested in the following schemes: fan-in Normal and Uniform with zero mean and variance $1/n_{in}$; fan-out Normal and Uniform with zero mean and variance $1/n_{out}$; Glorot Normal and Uniform, (Glorot & Bengio, 2010), where the mean is again zero and variance $2/(n_{in} + n_{out})$. Moreover, we consider the strategy proposed by (He et al., 2015), i.e. the weights are drawn from a Normal with zero mean and variance $2/n_{in}$. The theoretical justification for such methodology is that the ReLU activation does not have zero mean, and therefore the analysis presented by (Glorot & Bengio, 2010) does not hold, leading to suboptimal learning in ReLU networks. Finally, and despite lacking a sound theoretical argument, we also propose and investigate the fan-out case, i.e. zero-mean Normal with a variance of $2/n_{out}$.

Figure 5 shows the performance curves of deep neural networks using ReLU activations under different initialization schemes. We observe two major trends: the initialization proposed by (He et al., 2015) seems to be among the top performers and its usefulness becomes more apparent with increasing network depth; the corresponding fan-out initialization is clearly the worst performer and should not be used. The first observation is perhaps unsurprising as it reinforces the theory and the empirical results shown by (He et al., 2015). Regarding the second observation, it is not completely clear why the drop in performance is so significant. Overall, if we compare the fan-out with the corresponding fan-in initializations, it can be seen that, at least for networks of lower depth, the initial convergence seems to be faster, but these methods converge to worse local minima. Perhaps this can be attributed to variance over-

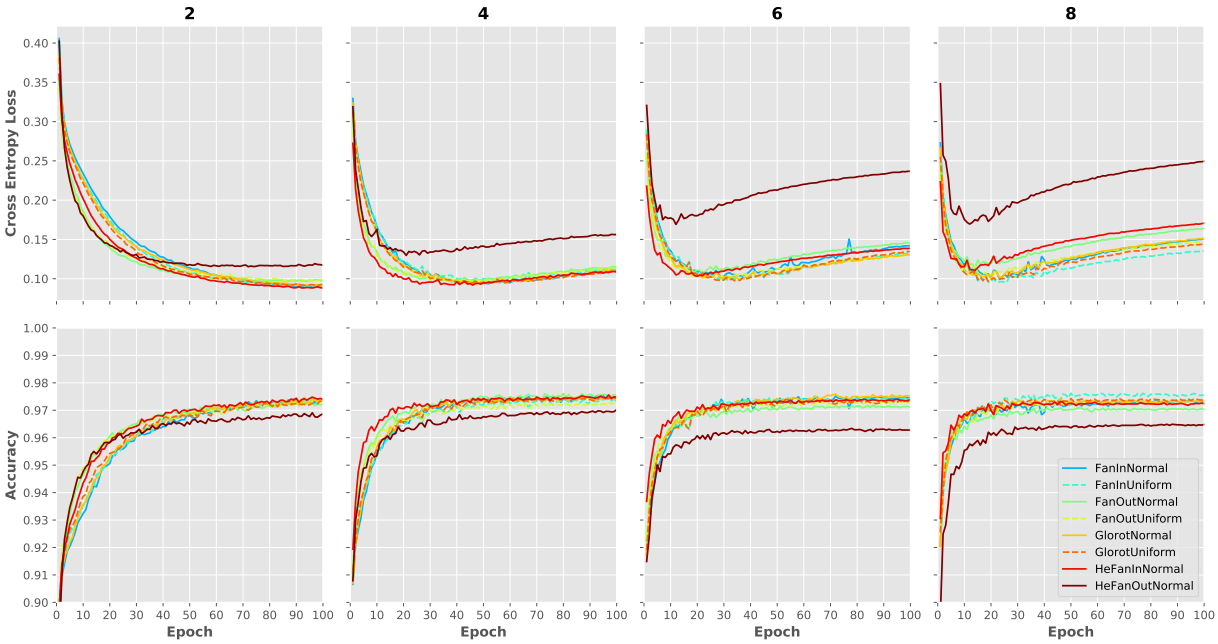

Figure 6. Validation cross entropy loss and accuracy curves of deep neural networks using SELU, ranging from 2 to 8 hidden layers (100 units each), trained for 100 epochs on the MNIST task. The learning rate (SGD) was fixed at 0.006. Different initialization schemes are tested.

estimation since, for these particular networks, in the first hidden and output layers, we have $n_{out} \ll n_{in}$. The factor of 2 in what we denoted by HeFanOutNormal aggravates the problem further, but this is only a speculative hypothesis. In this regard, the combined n_{in} and n_{out} approach taken by (Glorot & Bengio, 2010) might be interesting to test in the future, i.e. a variance of $4/(n_{in} + n_{out})$. In fact, despite the theory not holding for ReLU, the Uniform Glorot initialization achieved excellent results, particularly so for networks of 2, 4 and 6 hidden layers. Therefore, it remains to be seen whether combining this approach with the factor of 2 introduced by (He et al., 2015) might lead to a further boost in performance for ReLU networks. On the other hand, we believe that little can be said about Uniform versus Normal intializations without further testing. In particular, the Uniform initializations seem to lead to slightly better results, but we are unsure whether this trend holds in general. Again, for a future study, and in addition to other recognition tasks, it might be interesting to test He initialization with a Uniform distribution.

Figure 6 shows the corresponding validation error and accuracy curves for SELU networks. In comparison to ReLU, the curves are significantly less noisy, an observation that we had already reported in the previous section – but it is reassuring to see that this also holds for other initializations. Moreover, many of the comments we made regarding ReLU can be extended to SELU. In particular, HeFanInNormal seems to be one of the best methods, especially so for networks up to 6 hidden layers, and HeFanOutNormal remains the worst, with the drop in performance being even more noticeable for SELU networks. An interesting observation is that, for deeper architectures (e.g. 8 hidden layers), fan-in Uniform achieved outstanding results, better than that of fan-in Normal – note that the latter has been recommended by their authors. Thus, we would like to conclude this section by reiterating that further testing should be done in order to extract more definite conclusions.

6. Conclusions

In this work, we have studied the impact of different activation functions on handwritten digit recognition. We started this investigation by considering feedforward neural networks with 2 hidden layers and then expanded the analysis to deeper models. We have seen the effect of the gradient vanishing problem and how it degrades the performance of sigmoid activation functions when compared to ReLU. We have also observed that ReLU and LReLU networks exhibit a similar behavior and that this trend seems to hold for deeper models, reinforcing the conclusion in (Maas et al., 2013). In addition, it has been difficult to ascertain whether ELU and SELU provide a boost in performance in neural networks that are not too deep. However, for neural networks with multiple hidden layers, there are clear benefits in using these activation functions. This in turn supports the findings presented by their authors – they have assessed the performance of ELU, (Clevert et al., 2015), and SELU, (Klambauer et al., 2017), in the context of networks of at least 8 layers. Our last section analyzed different initialization schemes and we have been able to verify that the initialization of (Glorot & Bengio, 2010) yields good results, but the scheme introduced by (He et al., 2015) can be better.

Throughout the paper, we have left several questions and pointers to future research. In particular, we believe it would be interesting to perform some form of sensitivity analysis and test these methods on other recognition tasks involving different levels of complexity.

ACTIVATION	SIGMOID		RELU		LRELU		ELU		SELU	
VALIDATION	ERROR	ACCURACY	ERROR	ACCURACY	ERROR	ACCURACY	ERROR	ACCURACY	ERROR	ACCURACY
LR										
0.001	0.52818	0.86350	0.16530	0.95440	0.16591	0.95430	0.19779	0.94560	0.18107	0.95090
0.002	0.34114	0.90370	0.11789	0.96720	0.11848	0.96730	0.13825	0.96130	0.12696	0.96380
0.003	0.28905	0.91890	0.09951	0.97160	0.09997	0.97120	0.11253	0.96880	0.10664	0.96790
0.004	0.25800	0.92530	0.09203	0.97340	0.09240	0.97330	0.10021	0.97120	0.09832	0.97020
0.005	0.23454	0.93160	0.08909	0.97360	0.08921	0.97400	0.09347	0.97290	0.09468	0.97150
0.006	0.21562	0.93890	0.08815	0.97460	0.08789	0.97440	0.08988	0.97450	0.09322	0.97250
0.007	0.19996	0.94270	0.08879	0.97490	0.08883	0.97470	0.08833	0.97540	0.09362	0.97270
0.008	0.18674	0.94810	0.08942	0.97580	0.08991	0.97530	0.08808	0.97580	0.09482	0.97300
0.009	0.17540	0.95230	0.09130	0.97620	0.09134	0.97700	0.08865	0.97570	0.09585	0.97380
0.010	0.16559	0.95420	0.09270	0.97700	0.09263	0.97650	0.08971	0.97570	0.09698	0.97310
0.020	0.11413	0.96890	0.10418	0.97680	0.10403	0.97800	0.10286	0.97680	0.10872	0.97510
0.030	0.09678	0.97170	0.11313	0.97660	0.11251	0.97710	0.11082	0.97730	0.11456	0.97620
0.040	0.08908	0.97460	0.11448	0.97740	0.11415	0.97700	0.11519	0.97780	0.11964	0.97670
0.050	0.08578	0.97590	0.11348	0.97760	0.11355	0.97820	0.11825	0.97830	0.11757	0.97670
0.060	0.08492	0.97590	0.11336	0.97870	0.11581	0.97910	0.12057	0.97860	0.11756	0.97610
0.070	0.08519	0.97620	0.11520	0.97950	0.11318	0.97990	0.12213	0.97890	0.12149	0.97700
0.080	0.08599	0.97620	0.11764	0.97930	0.11637	0.97940	0.12258	0.97910	0.12396	0.97720
0.090	0.08710	0.97690	0.11347	0.97970	0.11743	0.97920	0.12275	0.97920	0.12234	0.97760
0.100	0.08832	0.97720	0.11653	0.97960	0.11213	0.98010	0.12292	0.97920	0.12179	0.97910
0.200	0.09801	0.97910	0.11702	0.98160	0.12036	0.97960	0.11825	0.98140	0.12140	0.97930

Table 1. Validation cross entropy loss and accuracy observed after training neural networks of 2 hidden layers with 100 units per hidden layer for 100 epochs on the MNIST task. Each row corresponds to a different learning rate (SGD). The best validation errors per activation function occurred for the following learning rates: sigmoid (0.060), ReLU (0.006), LReLU (0.006), ELU (0.008), SELU (0.006). These are also the points at which the validation error starts to increase for higher learning rates.

References

Clevert, Djork-Arné, Unterthiner, Thomas, and Hochreiter, Sepp. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv:1511.07289*, 2015.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *PMLR*, volume 9, pp. 249–256, 2010.

Glorot, Xavier, Bordes, Antoine, and Bengio, Yoshua. Deep sparse rectifier neural networks. In *PMLR*, volume 15, pp. 315–323, 2011.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv: 1502.01852*, 2015.

Klambauer, Günter, Unterthiner, Thomas, Mayr, Andreas, and Hochreiter, Sepp. Self-Normalizing Neural Networks. *arXiv: 1706.02515*, 2017.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature*, 521:436–444, 5 2015. doi: 10.1038/nature14539.

Maas, Andrew L., Hannun, Awni Y., and Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, 2013.

Nair, Vinod and Hinton, Geoffrey E. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proc. ICML*, pp. 807–814, 2010.