# On the Invariance of the SELU Activation Function on Algorithm and Hyperparameter Selection in Neural Network Recommenders

Flora Sakketou, Nicholas Ampazis

# On the Invariance of the SELU Activation Function on Algorithm and Hyperparameter Selection in Neural Network Recommenders

Flora Sakketou and Nicholas Ampazis

Department of Financial and Management Engineering
University of the Aegean, Chios, Greece
fmed15002@fme.aegean.gr, n.ampazis@fme.aegean.gr

**Abstract.** In a number of recent studies the Scaled Exponential Linear Unit (SELU) activation function has been shown to automatically regularize network parameters and to make learning robust due to its self-normalizing properties. In this paper we explore the utilization of SELU in training different neural network architectures for recommender systems and validate that it indeed outperforms other activation functions for these types of problems. More interestingly however, we show that SELU also exhibits performance invariance with regards to the selection of the optimization algorithm and its corresponding hyperparameters. This is clearly demonstrated by a number of experiments which involve a number of activation functions and optimization algorithms for training different neural network architectures on standard recommender systems benchmark datasets.

**Keywords:** Recommender Systems · Neural Networks · Activation Functions

## 1 Introduction

The literature review of studies on recommender systems [9, 24, 17], quickly makes apparent that latent factor models are among the ones most suited to the problem. The most successful realizations of latent factor models are based on matrix factorization. In its basic form, matrix factorization for recommender systems describes both users and items by vectors of (latent) factors which are inferred from the user-by-item rating matrix. High correlation between user and item factors leads to a recommendation of an item to a particular user [10].

The matrix factorization model can be considered as learning within a multilayer feedforward network framework where all neurons have identity activation functions, and which consists of $U$ inputs, $K$ hidden nodes and $I$ outputs, corresponding to the $U$ users and $I$ items of the recommendation problem [22]. User features $U_{uk}$ can be considered as the weights between the $u$th input and the $k$th hidden neuron, and item features $I_{ki}$ as the weights between the $k$th hidden neuron and the $i$th output neuron. For the $(u, j)$th rating, in order to obtain

output activations we can set the input $x_u$ to 1 and all other $x_{v \neq u} = 0$. In this case the network can be trained with Stochastic Gradient Descent (SGD) with regularization. In the testing phase, we set the input as in the training phase, and the network's outputs $y_i$ ($\mathrm{i} = 1 \cdots \mathrm{I}$) predict the active user's rating on the $i$th item.

In recent years other neural network architectures have been proposed for recommender systems, e.g. Restricted Boltzmann Machines (RBMs) [19], Deep AutoEncoders [12], and Variational AutoEncoders (VAE) [14]. In this work we explore the utilization of Feedforward Neural Network architectures (FNNs), and in particular architectures which take as input trainable embeddings of users and items, and propagate their contribution to subsequent layers. As is typical in FNN training, non-linear activation functions such as *sigmoid* or *tanh* are utilized for calculating the activations of the nodes. However recently a new activation function has been proposed, namely the Scaled Exponential Linear Unit (SELU) [8]. The use of the SELU activation function helps to regularize network parameters and makes learning robust due to its self-normalizing properties. SELU has been demonstrated to outperform other activation functions when utilized in deep FNN architectures in a number of standard machine learning benchmark problems such as MNIST [13] and CIFAR [11].

SELU is a variant of the Rectified Linear Unit (ReLU) activation function [16] whose derivative is constant for all positive input values. Due to the nature of the recommendation problem where the predicted values should lie within a positive range (typically in a scale between 1 and 5 "stars"), this means that positive values will remain positive during the learning process, thus reinforcing high correlations between user and item factors in suitable FNN architectures. Adding this observation to the beneficial self-normalizing properties that SELU exhibits, we expect its utilization in training FNN recommender systems to exhibit interesting characteristics. To the best of our knowledge, in the context of recommender systems, SELU has only been utilized in deep autoencoder networks [12].

In this paper we explore the utilization of SELU in training various FNN architectures for training recommender systems and demonstrate that it not only outperforms other activation functions as is already known from the literature, but that it also exhibits interesting invariance with regards to the selection of the optimization algorithm and its corresponding hyperparameters. We have performed a number of experiments with various optimization algorithms for training different FNN architectures on standard recommender systems benchmark datasets, which clearly demonstrate this invariance property.

## 2 Activation Functions

### 2.1 ReLU

FNN training with non-linear activation functions such as *sigmoid* or *tanh* suffer by the vanishing gradient problem towards the lower (input) layers [1]. Therefore

the FNNs that perform well are usually shallow. A common way to tackle the vanishing gradient problem is the utilization of the ReLU activation function [16] which prevents gradients from saturating since its gradient has a constant value. The ReLU activation function is given by:

$$f(x) = max(0, x) \tag{1}$$

Thus ReLU is linear (identity) for all positive values and returns zero if it receives any negative input hence the mean activation of the nodes is greater than zero. In essence these units act as bias for units in the next layer causing a bias shift which hinders learning.

### 2.2 ELU

In contrast to ReLU, the Exponential Linear Unit (ELU) activation function has negative values which bring mean unit activations centered around zero. Zero means can accelerate learning because they bring the gradient closer to the unit natural gradient [2]. The ELU activation function is given by the following formula:

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha e^x - \alpha, & \text{if } x \leq 0 \end{cases} \tag{2}$$

ELU is comprised of two parts: the positive part of the function is the identity and the negative part exponentially skews the negative values. The hyperparameter $\alpha$ controls the convergence rate when the network inputs tend to be negative infinite.

### 2.3 SELU

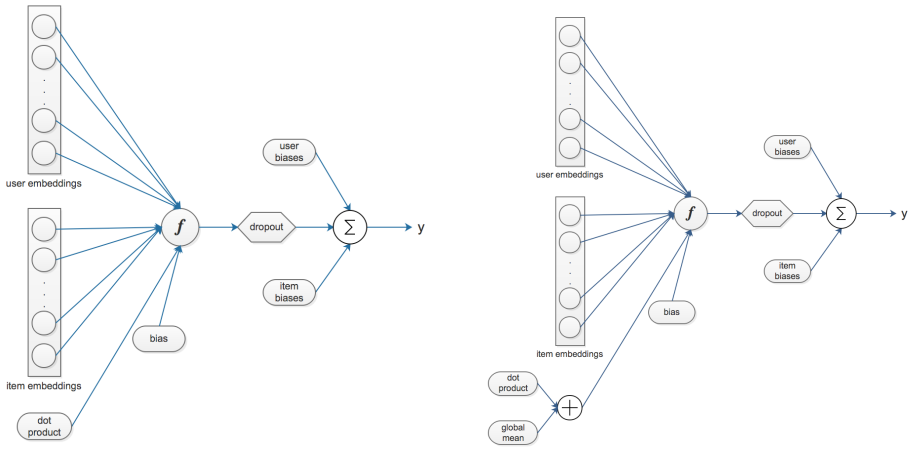
During training, the distribution of the network activations changes at every training step which may slow down training. This is known as the internal covariate shift problem [20] which renders training neural networks hard since it demands smaller learning rates and careful parameter initialization. One way to address this problem is by using Batch Normalization (BN) [5] which normalizes the layer inputs in order to follow the standard normal distribution with zero mean and unit variance. SELU [8] deals with the covariant shift problem and has additional advantages over BN since it comes with lower computational complexity and has self-normalizing properties, in the sense that node activations remain centered around zero and have unitary variance. Due to its properties, SELU makes learning highly robust and allows to train networks that have many layers. SELU is defined as:

$$selu(x) = \lambda \begin{cases} x, & \text{if } x > 0 \\ \alpha e^x - \alpha, & \text{if } x \leq 0 \end{cases} \tag{3}$$

For standard scaled inputs (zero mean and standard deviation one) the values of the parameters $\alpha$ and $\lambda$ are $\alpha \approx 1.6732$ and $\lambda \approx 1.0507$ [8]. The initial weights of nodes with the SELU activation function should be drawn from a normal distribution with zero mean and variance equal to $1/n$ where $n$ is the size of the input. In addition, in [8] a new dropout technique [21] is proposed, called Alpha Dropout. The proposed dropout technique randomly sets inputs to $-\lambda\alpha$ which is the point that the network inputs tend to be negative infinite. Then in order to ensure the self-normalizing property, an affine transformation is applied to the inputs using parameters that are relative to the dropout rate, targeted mean and variance. The proposed Alpha Dropout rates are 0.05 or 0.1.

## 3 Neural Network Architectures

### 3.1 Single Layer Feedforward Neural Network




(a) Single Layer FNN (b) Single Layer FNN with Global Mean

Fig. 1

Neural Network embeddings have been proven to be very powerful both for modeling language and for representing categorical variables. For example, the Word2Vec word embeddings [15] map a word to a vector based on training a neural network on large corpora. These embeddings can be used in any supervised model because they are just numerical representations of categorical variables. In the context of recommender systems, we can utilize the concatenation of two embedding layers [3] as input to an FNN architecture. The first embedding layer corresponds to the encoding of all users whereas the second embedding layer encodes all items. In all our experiments presented in section 4 we have utilized embeddings layers of dimensionality 100. In addition the dot product of the embeddings of each user/item pair in the training set can be added as an extra

input node to the network in order to reinforce positive correlations between user and item latent factors (embeddings) as shown in Figure 1(a). The network has only a single output node with a bias weight. Network regularization is implemented by the utilization of a dropout layer. The output of the dropout layer is then added to (trainable) user and item biases as it is customary practice in recommendendation models to account for the variations between the ratings that each user provides and conversely for the variations between the ratings that each item receives [9].

### 3.2 Single Layer Feedforward Neural Network with Global Mean

This network architecture is shown in Figure 1(b) and is a variation of the FNN shown in Figure 1(a). The variation consists of the addition of the global mean of the ratings in the training set to the dot product of the user and item embeddings. The inclusion of the global mean has been proved to have a beneficial effect to the training of many recommendation models as it maintains the output of the prediction centered around this value. This results in smaller gradients of the loss function which helps to prevent oscillations and accelerates learning [24].

### 3.3 Double Layer Feedforward Neural Network

A double layer FNN with a single hidden layer can be utilized for the recommendation problem in order to encapture higher order dependencies. This network architecture is shown in Figure 2. The input consists of the concatenation of the user and item embeddings which are fed to a hidden layer. In all our experiments we set the hidden layer size to 128 nodes. Each node also has a bias weight. A dropout layer is then added for regularization and the output of that layer is then added to the trainable user and item biases as in the case of the single layer FNN architecture.

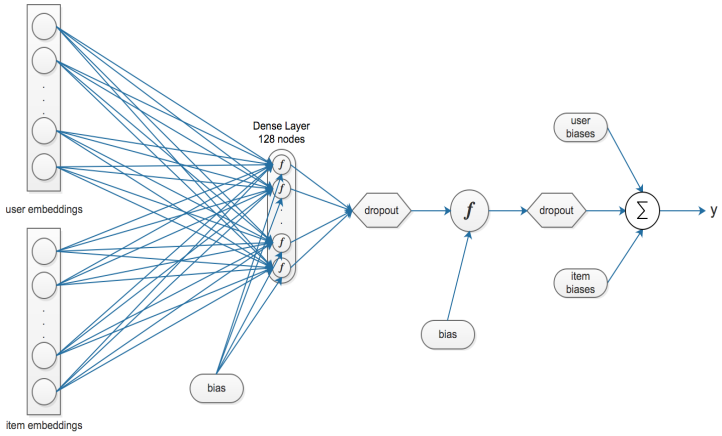



Fig. 2: Double Layer Feedforward Neural Network

## 4 Experimental Results

In this section, we evaluate the performance of ReLU, ELU and SELU for the three different neural network architectures mentioned in section 3. For the evaluation we utilized the Movielens-100K and Movielens-1M benchmark datasets [4]. For each dataset we performed $k$-fold cross validation (with $k = 5$) and split the data using stratified sampling per user, so that 80% of the ratings of each user were used for training and the remaining 20% of each user's ratings were used for validation. We report the average Mean Absolute Error (MAE) on the validation sets over the $k$ folds, as given by the following formula:

$$MAE = \frac{1}{k}\sum_{j=1}^{k}\sqrt{\frac{1}{|(R_{tr})_j|}\sum_{R_{ui}\in(R_{tr})_j}\left|(R_{ui}-\hat{R}_{ui})\right|} \tag{4}$$

where $k$ is the nuber of folds, $|(R_{tr})_j|$ is the size of each validation set $(R_{tr})_j$, and $R_{ui}$, $\hat{R}_{ui}$ are the actual and predicted ratings of user $u$ on item $i$ respectively. Three optimization algorithms were employed on the experiments, namely Stochastic Gradient Descent (SGD) [6], Adam [7] and RMSprop [23]. For each algorithm a grid search was performed, the parameters of which are shown in Table 1. The grid search was complemented by employing each activation function with various dropout rates as shown in Table 2. For each trial, the initial weights of all network architectures were drawn from the normal distribution with zero mean and standard deviation of 0.1, starting from the same random seed in order to obtain fair comparisons. For the SELU trials the weights of the network were initialized from the normal distribution with zero mean and variance equal to $1/n$ where $n$ is the size of input according to its specifications guidelines [8].

All the experiments were implemented within the Tensorflow framework [18] and we have made all the codes publicly available on a Github repository[1].

Table 1: Algorithm parameters

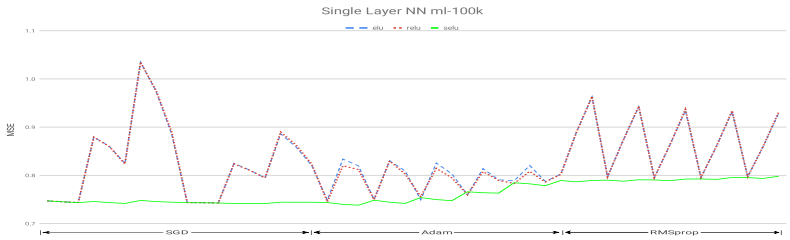
| | Learning Rates | Momentum |
|---|---|---|
| SGD | 0.01, 0.02 | 0, 0.2, 0.5 |
| Adam | 0.0005, 0.0007, 0.001, 0.002, 0.003 | - |
| RMSprop | 0.0005, 0.0007, 0.001, 0.002, 0.003 | - |

Table 2: Dropout rates

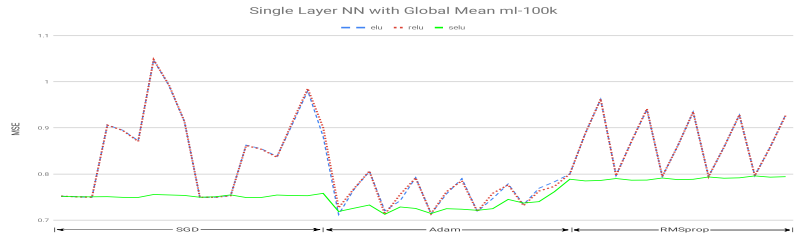
| | Dropout | Alpha Dropout |
|---|---|---|
| ReLU | 0, 0.2, 0.5 | - |
| ELU | 0, 0.2, 0.5 | - |
| SELU | - | 0, 0.05, 0.1 |

*Experimental results on the Movielens-100K dataset:* Figure 3a shows the experimental results for the Single Layer Feedforward Neural Network architecture described in section 3.1. For each activation function we report the average validation MAE over the 5-folds versus various combinations of hyperparameter choices, grouped by optimization algorithm. We can see that the MAE values achieved by ReLU and ELU are quite similar and, regardless of the choice of optimization algorithm and its hyperparameters, result to oscillations. These oscillations are mainly attributed to the choice of the dropout rate, where for a

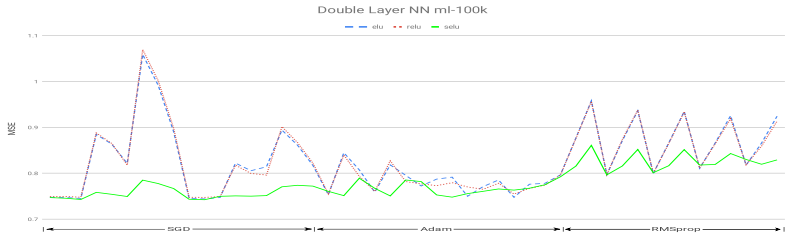
[1] `https://github.com/flo3003/recsys_fnn.git`




(a) Single Layer Feedforward Neural Network




(b) Single Layer Feedforward Neural Networkwith Global Mean




(c) Double Layer Feedforward Neural Network

Fig. 3: Experimental results on the Movielens-100K dataset for the three different neural network architectures: each subfigure shows the average validation MAE over the 5-folds versus various combinations of hyperparameters grouped by optimization algorithm (SELU is depicted with a green solid line, ELU with a blue dashed line, and ReLU with a red dotted line).

fixed parameter choice of e.g. learning rate, the error increases as the dropout rate increases. On the contrary SELU's MAE results are contained within a smaller range and consistently outperform those of both ReLU and ELU, regardless of the changes to either the optimization algorithm and/or its hyperparameters. The fact that SELU achieves consistently better results in the Single Layer Feedforward Neural Network architecture, is also depicted in the first row of Table 3 where the average MAE for the SELU is reported as 0.7627 whereas the average MAE for ReLU and ELU is reported as 0.8299 and 0.8309 respectively. The invariance property of SELU can also be demonstrated by the variance in MAE shown in the same Table. SELU's MAE variance (4.81e-04) is practically 10 times lower than the MAE variance of both ReLU (4.81e-03) and

ELU (4.69e-03). In order to evaluate the statistical significance of the results, we ranked the activation functions by the average MAE result and performed a paired Wilcoxon test [25]. The last column of that Table shows the $p-$values of the Wilcoxon test and portray that the differences of SELU's results to the other two functions is statistically significant. Furthermore, in terms of optimum results, SELU obtains better minimum MAE (0.7382) than that of both ReLU and ELU (0.7428 in both cases).

Figure 3b shows the experimental results for the Single Layer Feedforward Neural Network with global mean architecture described in section 3.2. Again, for each activation function we report the average validation MAE over the 5-folds versus various combinations of hyperparameter choices, grouped by optimization algorithm. Similarly to the previous architecture, SELU retains consistently low MAE results contrary to both ReLU and ELU which oscillate regardless of the choice of optimization algorithm and its hyperparameters. The second row of Table 3 shows that for this FNN architecture ReLU's, ELU's and SELU's minimum results are practically the same, with ELU slightly outperforming the other two. However, SELU's average MAE (0.7568) is significantly lower than the average MAE of both ReLU (0.8299) and ELU (0.8291). The invariance of SELU is reflected in its MAE variance (6.44e-04), which, in this case too, is practically 10 times lower than the MAE variances of both ReLU (6.92e-03) and ELU (6.92e-03). The $p-$values of the Wilcoxon test indicate that the differences of SELU's average MAE result compared to the other two activation functions is statistically significant.

Finally for this dataset, Figure 3c shows the experimental results for the Double Layer Feedforward Neural Network architecture described in section 3.3. Again, for each activation function we report the average validation MAE over the 5-folds versus various combinations of hyperparameter choices, grouped by optimization algorithm. Compared to the two previous architectures, SELU exhibits some small oscillations, especially when the RMSprop algorithm is used, but continues to obtain lower and more consistent MAE results than ReLU and ELU, which both oscillate practically in the same way as in the two previous cases. This behavior of SELU is depicted in the third row of Table 3, which shows an increase in the MAE variance (1.10e-03) in comparison to the previous two architectures but still remains 5 times lower than the MAE variances of both ReLU (5.30e-03) and ELU (5.13e-03). In addition SELU's average MAE (0.7807) is lower than the average MAE of ReLU (0.8283) and ELU (0.8290). We also note that SELU (0.7425) achieves better MAE than ReLU and ELU (0.7476 and 0.7444 respectively). The $p-$values of the Wilcoxon test indicate that the differences of ReLU's and ELU's results compared to the best perfoming method is statistically significant.

*Experimental results on the Movielens-1M dataset:* Figure 4a shows the experimental results for the Single Layer Neural Network architecture described in section 3.1. As in the Movielens-100K dataset, for each activation function we report the average validation MAE over the 5-folds versus various combinations of hyperparameter choices, grouped by optimization algorithm. From this Figure we can see that ReLU's and ELU's MAE results are practically the same

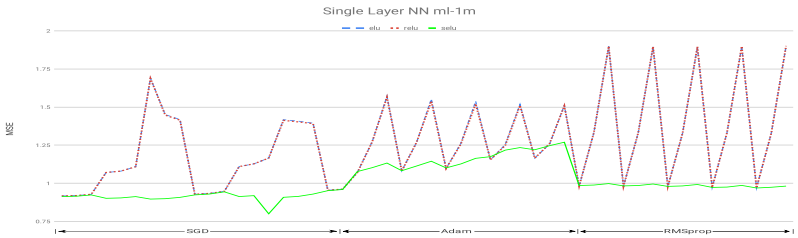
Table 3: Experimental results for MovieLens-100K. The Table shows the minimum and maximum value of MAE as well as the mean and variance of the MAE results over the grid search results shown in Figure 3. The last column shows the $p-$value of the Wilcoxon test.

| | **Activation Function** | **Minimum value** | **Maximum value** | **Mean** | **Variance** | $p-$**value** |
|---|---|---|---|---|---|---|
| Single Layer FNN | ReLU | 0.7428 | 1.0348 | 0.8299 | 4.81e-03 | 6.83e-09 |
| | ELU | 0.7428 | 1.0349 | 0.8309 | 4.69e-03 | 8.21e-09 |
| | SELU | **0.7382** | **0.7982** | **0.7627** | **4.81e-04** | |
| Single Layer FNN with global mean | ReLU | 0.7128 | 1.0477 | 0.8299 | 6.96e-03 | 6.47e-08 |
| | ELU | **0.7123** | 1.0471 | 0.8291 | 6.92e-03 | 9.64e-08 |
| | SELU | 0.7127 | **0.7959** | **0.7568** | **6.44e-04** | |
| Double Layer FNN | ReLU | 0.7476 | 1.0693 | 0.8283 | 5.30e-03 | 1.95e-06 |
| | ELU | 0.7444 | 1.0588 | 0.8290 | 5.13e-03 | 1.36e-06 |
| | SELU | **0.7425** | **0.8609** | **0.7807** | **1.10e-03** | |

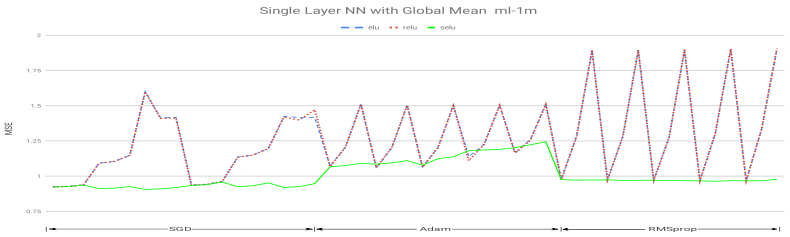
and oscillate strongly regardless of the choice of optimization algorithm and its hyperparameters. We note that the oscillations are stronger when RMSprop is used. Compared to the Movielens-100K dataset SELU's MAE results are slightly more sensitive to the choice of algorithm but still remain within a small range of the algorithm's hyperparameter grid search, and are consistently better than those of ReLU and ELU. The first row of Table 4 shows that in terms of optimum results, SELU obtains a considerably better MAE (0.8) than both ReLU and ELU ($\approx$ 0.917 in both cases). Interestingly from that Table we can see that there is also a significant difference between the average MAE of SELU (1.0107) and the average MAE of ReLU (1.2820) and ELU (1.2834). In addition SELU's MAE variance (1.27e-02) is almost 7 times lower than ReLU's (8.37e-02) and ELU's (8.41e-02) MAE variances. The $p-$values of the Wilcoxon test show that the differences of SELU's results to the other two functions is statistically significant.

Figure 4b shows the experimental results for the Single Layer Neural Network with global mean architecture described in section 3.2. For each activation function we report the average validation MAE over the 5-folds versus various combinations of hyperparameter choices, grouped by optimization algorithm. SELU retains consistently low MAE results (with a slight increase when Adam is used), contrary to ELU and ReLU which oscillate practically in the same way as in the previous FNN architecture. The second row of Table 4 shows that, similarly to the previous FNN architecture, SELU (0.9058) outperforms ReLU (0.9238) and ELU (0.9237) in terms of minimum MAE and SELU's average MAE (1.0069) is again significantly lower than the average MAE of ReLU (1.2721) and ELU (1.2727). Furthermore, the invariance property is noted in SELU's MAE variance (9.35e-03), which, in this case, is 11 times lower than ReLU's (8.10e-02) and ELU's (8.07e-02) MAE variance. Again, the $p-$values of the Wilcoxon test indicate that the differences of SELU's average MAE result to the other two functions is statistically significant.

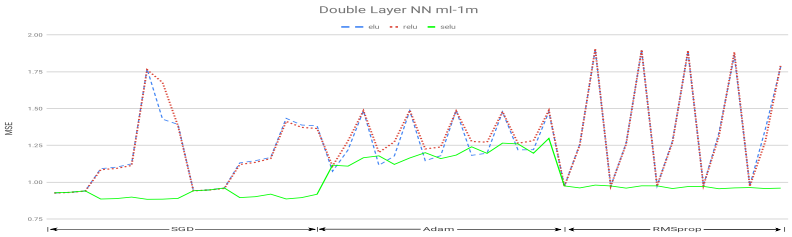
Finally, Figure 4 shows the experimental results for the Double Layer Feedforward Neural Network architecture described in section 3.3. For each activation




(a) Single Layer Feedforward Neural Network




(b) Single Layer Feedforward Neural Networkwith Global Mean




(c) Double Layer Feedforward Neural Network

Fig. 4: Experimental results on the Movielens-1M dataset for the three different neural network architectures: each subfigure shows the average validation MAE over the 5-folds versus various combinations of hyperparameters grouped by optimization algorithm (SELU is depicted with a green solid line, ELU with a blue dashed line, and ReLU with a red dotted line).

function we report the average validation MAE over the 5-folds versus various combinations of hyperparameter choices, grouped by optimization algorithm. With this FNN architecture, ELU exhibits stronger oscillatory behavior. Due to these oscillations and only when Adam is used with certain hyperparameters ELU manages to achieve slightly better results than SELU. However, except for those few occasions, SELU obtains lower and and more consistent MAE results than ReLU and ELU across the grid search, thus preserving its invariance. This is clearly depiscted in the third row of Table 4, which shows SELU's MAE variance (1.10e-03) which is almost 5 times lower than the MAE variances of ReLU (5.30e-03) and ELU (5.13e-03). As with the previous architecture cases, SELU's average MAE (1.0156) is notably lower than ReLU's (1.287) and ELU's

(1.2718) average MAE. Moreover, SELU achieves noticably better MAE (0.8838) than ReLU (0.9265) and ELU (0.9267). In this case also, the $p-$values of the Wilcoxon test show that the benefits of the utilization of the SELU as activation function are statistically significant.

Table 4: Experimental results for MovieLens-1M. The Table shows the minimum and maximum value of MAE as well as the mean and variance of the MAE results over the grid search results shown in Figure 4. The last column shows the $p-$value of the Wilcoxon test.

| | **Activation Function** | **Minimum value** | **Maximum value** | **Mean** | **Variance** | $p-$**value** |
|---|---|---|---|---|---|---|
| Single Layer FNN | ReLU | 0.9172 | 1.9015 | 1.2820 | 8.37e-02 | 2.62e-07 |
| | ELU | 0.9170 | 1.8995 | 1.2834 | 8.41e-02 | 2.22e-07 |
| | SELU | **0.8000** | **1.2695** | **1.0107** | **1.27e-02** | |
| Single Layer FNN with global mean | ReLU | 0.9238 | 1.9051 | 1.2721 | 8.10e-02 | 1.60e-07 |
| | ELU | 0.9237 | 1.9038 | 1.2727 | 8.07e-02 | 1.51e-07 |
| | SELU | **0.9058** | **1.2445** | **1.0069** | **9.35e-03** | |
| Double Layer FNN | ReLU | 0.9265 | 1.9059 | 1.2870 | 7.89e-02 | 5.14e-08 |
| | ELU | 0.9267 | 1.9016 | 1.2718 | 7.70e-02 | 3.22e-06 |
| | SELU | **0.8838** | **1.2975** | **1.0156** | **1.56e-02** | |

## 5 Conclusion

In this paper we examined the utilization of the SELU activation function in FNN architectures for recommender systems. Experimental results on standard recommender systems benchmark datasets (Movielens-100K and Movielens-1M) demonstrated that SELU is invariant on the choice of optimization algorithm and its corresponding hyperparameters. SELU performed consistently across all the combination of datasets, algorithms and hyperparameters reinforcing the belief that its self-normalizing properties for the network parameters is especially beneficial for neural network recommender systems due to the nature of the problem. Our future plans include the utilization of deeper neural network architectures which will also address recommendation as a ranking problem.

## References

1. Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.
2. D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.
3. Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc., 2016.
4. F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, Dec. 2015.

5. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.
6. J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, 23(3):462–466, 09 1952.
7. D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
8. G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. *CoRR*, abs/1706.02515, 2017.
9. Y. Koren. The BellKor Solution to the Netflix Grand Prize. August 2009.
10. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42:30–37, 2009.
11. A. Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.
12. O. Kuchaiev and B. Ginsburg. Training Deep AutoEncoders for Collaborative Filtering. *ArXiv e-prints*, Aug. 2017.
13. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.
14. D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 689–698, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.
15. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.
16. V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.
17. M. Piotte and M. Chabbert. The Pragmatic Theory Solution to the Netflix Grand Prize. August 2009.
18. B. Ramsundar and R. B. Zadeh. *TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning*. O'Reilly Media, Inc., 1st edition, 2018.
19. R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the International Conference on Machine Learning*, volume 24, pages 791–798, 2007.
20. H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function, 2000.
21. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.
22. G. Takács, I. Pilászy, B. Németh, and D. Tikk. On the Gravity recommendation system. In *Proc. of KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 22–30, San Jose, CA, USA, August 12–15, 2007.
23. T. Tieleman and G. Hinton. RMSprop Gradient Optimization.
24. A. Toscher, M. Jahrer, and R. Bell. The Big Chaos Solution to the Netflix Grand Prize. August 2009.
25. F. Wilcoxon. *Individual Comparisons by Ranking Methods*, pages 196–202. Springer New York, New York, NY, 1992.