CancelOut: A Layer for Feature Selection in Deep Neural Networks

Vadim Borisov[1(✉)], Johannes Haug[1], and Gjergji Kasneci[1,2]

[1] Eberhard Karls University of Tübingen, Tübingen, Germany
{vadim.borisov,johannes-christian.haug,gjergji.kasneci}@uni-tuebingen.de
[2] SCHUFA Holding AG, Wiesbaden, Germany

Abstract. Feature ranking (FR) and feature selection (FS) are crucial steps in data preprocessing; they can be used to avoid the curse of dimensionality problem, reduce training time, and enhance the performance of a machine learning model. In this paper, we propose a new layer for deep neural networks - CancelOut, which can be utilized for FR and FS tasks, for supervised and unsupervised learning. Empirical results show that the proposed method can find feature subsets that are superior to traditional feature analysis techniques. Furthermore, the layer is easy to use and requires adding only a few additional lines of code to a deep learning training loop. We implemented the proposed method using the PyTorch framework and published it online (The code is available at: www.github.com/unnir/CancelOut).

Keywords: Deep learning · Feature ranking · Feature selection · Unsupervised feature selection · Machine learning explainability

1 Introduction

Feature importance and interpretability of machine learning (ML) models have received much attention in the recent years due to the fact that accurate estimations are not always enough to solve a data problem. An explanation of machine learning model outcomes may help not only to understand the model's results, but also to introduce new tests, better understand the data, and as a consequence from above, improve trust in the model, which is important when the model is used by specialists from other fields. However, most accurate and robust ML models usually cannot be interpreted [4].

One of the most effective ML methods nowadays is deep learning (DL), which can be explained in terms of the universal approximation theorem [2]. Which principally states that any compactly supported continuous function on $\mathbb{R}^n$ can be approximated with a single hidden layer feed-forward neural network (NN). However, due to DL's high inherent complexity, most DL models are primarily handled as a black box. Even though recent attempts have been made to address the issue of their interpretability and feature selection [1,4], existing methods are complicated.

I. V. Tetko et al. (Eds.): ICANN 2019, LNCS 11728, pp. 72–83, 2019.
https://doi.org/10.1007/978-3-030-30484-3_6

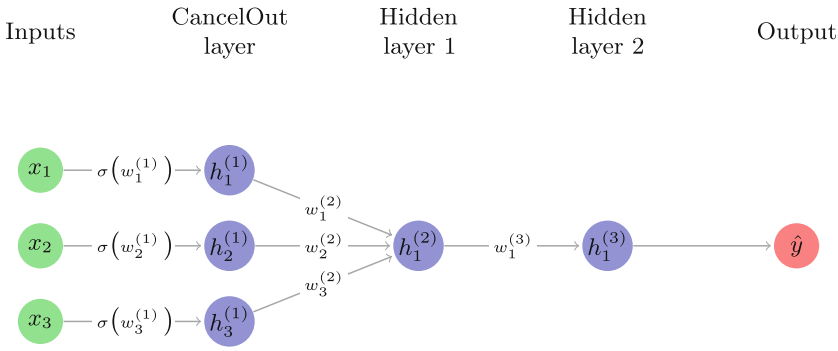


Fig. 1. A deep neural network with a CancelOut layer as an input layer.

2 Related Work

Many research articles on feature ranking (FR) and feature selection (FS) using DL propose the permutation method, which is based on the idea that if we remove or corrupt a feature from a dataset, the performance will change. By analyzing these changes, it is possible to determine if a feature is valuable or not. The obvious drawback of that idea is that it is computation-intensive, and in order to check n features, one need to train a DL model at least n times.

Another similar strategy was proposed in [1], where the dropout layer is exploited for feature ranking [12] for feature ranking. To analyze which features are important, an artificial NN model with a dropout layer must achieve minimum loss, and the dropout layer should learn small dropout rates for features that are important, while increasing the dropout rate for the rest of unimportant features. In this case, a model can be run once.

An interesting approach for quantifying the influence of individual input signals on the output computed by a deep neural network was proposed in the paper [6]. This method is based on the estimation of local linear models for each neuron in the network and the propagation and aggregation of these models into a net wide model.

The work in [9] introduces a similar idea to the linear models with elastic net regularization, but it employs a NN with multiple layers. This method regularizes input weights in a loss function using $l1$ and $l2$ norms together; without these terms the method is unstable.

Furthermore, many articles investigate an interpretation of a decision of the ANN for a single data sample [13]. However, this approach is hard to apply for feature ranking of a whole dataset.

The paper is divided into four sections. In Sect. 3, the proposed approach for feature ranking is introduced. Section 4 presents the implementation and result of the study. Finally, Sect. 5 contains a summary of the work.

3 CancelOut

In this Section, we present a new layer for deep neural networks - CancelOut, a method that helps identify a subset of relevant input features (variables) in a dataset. Also, the proposed method can be applied for feature sensitivity analysis.

CancelOut is an artificial neural network (ANN) layer, which is comparable to a fully connected (FC) layer with one distinction: neurons in the FC layer have connections to every input, whereas neurons in the CancelOut layer have only one connection to one particular input (Fig. 1).

The primary idea behind CancelOut layer is to update its weights (W_{CO}) during the training stage so that irrelevant features will be *canceled out* with a negative weight (Eq. 1). Otherwise, the best variables, which contribute more to a learning process, are going to be passed through with a positive weight.

$$CancelOut(\boldsymbol{X}) = \boldsymbol{X} \odot g(W_{CO}) \tag{1}$$

where $\odot$ indicates an element-wise multiplication, $\boldsymbol{X}$ is an input vector $\boldsymbol{X} \in \mathbb{R}_v^N$, W_{CO} is a weight vector $W_{CO} \in \mathbb{R}_v^N$, N_v is the feature size, and g is an activation function. Note, $g(x)$ denotes here element-wise application, e.g. $\boldsymbol{X} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$, then $g(\boldsymbol{X}) = g\left(\begin{bmatrix} a \\ b \\ c \end{bmatrix}\right) = \left(\begin{bmatrix} g(a) \\ g(b) \\ g(c) \end{bmatrix}\right)$.

3.1 Theoretical Justification of the CancelOut Layer

For simplicity, consider a three layers artificial neural network (Fig. 1) with *linear activation function* after layer $^{(2)}$ and $^{(3)}$, where the CancelOut layer with the sigmoid activation function σ is utilized as input layer (superscript $^{(1)}$). Note that bias terms in FC layers are omitted for simplicity reasons. Then the output of the network is given by:

$$\hat{y} = \boldsymbol{X} \odot \sigma(W_{CO}) \cdot W^{(2)} w_1^{(3)} \tag{2}$$

where $\boldsymbol{X} = [x_1, x_2, x_3]$, $W_{CO} \in \mathbb{R}^3, W^{(2)} \in \mathbb{R}^3$, and $w_1^{(3)} \in \mathbb{R}$. Note, $\cdot$ is the dot product between two vectors.

The Eq. 2 can be seen as a linear regression model:

$$\hat{y} = X\theta_1 + \theta_0 \tag{3}$$

where θ_1:

$$\theta_1 = \sigma(W_{CO}) \cdot W^{(2)} w_1^{(3)} \tag{4}$$

In case of multiple artificial neurons in the hidden layer$^{(2)}$ (Fig. 2), the output is given by:

$$\hat{y} = \boldsymbol{X} \odot \underbrace{\sigma(W_{CO}) \cdot W_1^{(2)} w_1^{(3)}}_{\theta_1} + \boldsymbol{X} \odot \underbrace{\sigma(W_{CO}) \cdot W_2^{(2)} w_2^{(3)}}_{\theta_2} \tag{5}$$

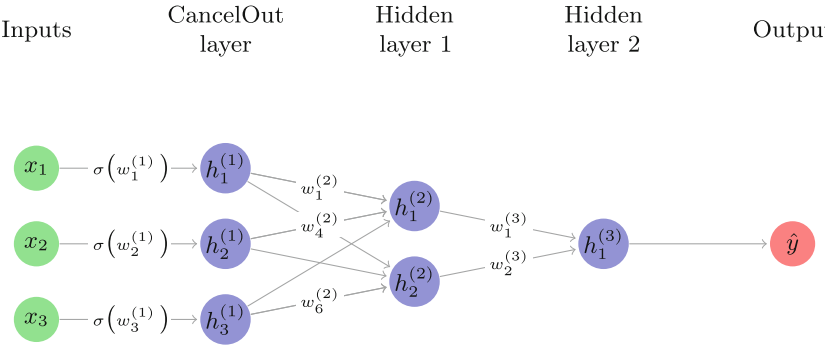


Fig. 2. A deep neural network with a CancelOut layer and two artificial neurons in the third layer.

From the Eq. 2 it can be seen that if the any value after the activation function $g(w_{CO}^{(1)}))$ in the CancelOut layer equals 0, than the corresponding input value x_n to $g(w_{CO}^{(1)}))$ is also 0. The following lemma summarizes this idea.

Lemma 1. *If a value after an activation function from the CancelOut layer is 0, a corresponding input variable does not affect the output of an ANN.*

As an illustration: let a value after the activation function from the CancelOut layer be 0.

$$\sigma(w_1^{(1)}) = 0 \tag{6}$$

Then $\hat{y}$ from Eq. 5 from the ANN 5, can be seen as:

$$\begin{aligned}\hat{y} &= \boldsymbol{X} \odot \sigma(W_{CO}) \cdot W_1^{(2)} w_1^{(3)} + \boldsymbol{X} \odot \sigma(W_{CO}) \cdot W_2^{(2)} w_2^{(3)} \\ &= \cancel{x_1 \sigma(w_1^{(1)}) w_1^{(2)} w_1^{(3)}} + x_2 \sigma(w_2^{(1)}) w_2^{(2)} w_1^{(3)} + x_3 \sigma(w_3^{(1)}) w_3^{(2)} w_1^{(3)} \\ &\quad + \cancel{x_1 \sigma(w_1^{(1)}) w_4^{(2)} w_2^{(3)}} + x_2 \sigma(w_2^{(1)}) w_5^{(2)} w_2^{(3)} + x_3 \sigma(w_3^{(1)}) w_6^{(2)} w_2^{(3)}\end{aligned} \tag{7}$$

Remark 1. Clearly, if $w_1^{(2)} = w_4^{(2)} = 0$ in the Eq. 7 and the bias[1] term is also zero, then a CancelOut weight $w_1^{(1)}$ does not represent beneficial information. In order to avoid this outcome, we suggest to consider these recommendations:

- a proper choice of activation function in a layer after the CancelOut layer helps to bypass that issue;
- regularization terms can be used in a loss function;
- finally, it is advisable to have numerous artificial neurons in the layer after the CancelOut layer, since it diminishes the chance that all neurons weights in the layer after $W^{(2)}$ the CancelOut layer be zeros.

[1] The bias term is omitted here, see Subsect. 3.1.

Moreover, if a weight w_i in the CancelOut layer is 0, then the gradient with respect to w_i is 0. The following lemma summarizes this idea.

Lemma 2. *If a value after an activation function from the CancelOut layer is approximately 0, then the gradient of the weight is also approximately 0.*

Combining Lemmas 1 and 2, we get the following theorem:

Theorem 1. *Values after the activation function in the CancelOut layer indicate contributions to the output of a corresponding variable.*

Consequently, the CancelOut layer ranks features in a similar way linear models do, e.g. the large the absolute values in the CancelOut layer the more a corresponding input variable contributes to the output. Also, compared to linear models, the CancelOut method takes into account linear and non-linear combinations of input data.

Remark 2. A zero coefficient in CancelOut values $\sigma(W_{CO}^{(1)})$ leads to fewer optimization parameters, hence, a model learns faster. This also helps to reduce, it helps to reduce the number of features, therefore mitigating overfitting. Moreover, feature selection with the CancelOut layer can be adopted in two scenarios; a user can either specify the number of features or extract features using a chosen threshold.

Our FR approach is similar to [9], but in our work, the input scalar $\sigma(W_{CO}^{(1)})$ is bound in the chosen interval (e.g. for the sigmoid activation function is $\big(0, 1\big)$). Therefore our approach is more stable, and it is simpler to rank features since a user selects only a threshold. Besides, the CancelOut FR method does not require a penalty coefficient in a loss function.

3.2 CancelOut Layer Weights Initialization

A random weight initialization is not desired for the CancelOut layer, since it may give an advantage for one subset of features over another. Therefore, weights are initialized with uniformly distribution [5] with additional β coefficient:

$$W_{CO} \sim \mathcal{U}(-\frac{1}{\sqrt{n_{in}}} + \beta, \frac{1}{\sqrt{n_{in}}} + \beta) \tag{8}$$

where n_{in} is the size of the previous layer, and β is the coefficient which depends on the choice of an activation function.

We introduced β coefficient into Eq. 8 in order to control the initial output values after an activation function $g(W_{CO})$, it needs to be $g(W_{CO}) \neq 0$, because we assume that every feature is equally important e.g. for the logistic activation function $\beta \in [-3, \text{inf})$.

3.3 Loss Function

In order to accelerate the feature ranking process in the CancelOut layer, we introduce two regularization terms in a loss function (Eq. 9):

$$\mathbb{L}(X,Y) = \mathcal{L}(X,Y) - \lambda_1 \, var(\frac{W_{CO}}{N_v}) + \lambda_2 \left\| \frac{W_{CO}}{N_v} \right\|_1 \tag{9}$$

where $\mathcal{L}$ is a selected loss function, for the classification task it can be seen as:

$$\mathcal{L}_{CE}(X,Y) = -\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} \ln \psi(x^{(i)}) + \left(1 - y^{(i)}\right) \ln \left(1 - \psi(x^{(i)})\right)) \\ -\lambda_1 \, var(\frac{W_{CO}}{N_v}) + \lambda_2 \left\| \frac{W_{CO}}{N_v} \right\|_1 \tag{10}$$

where $X = \left\{x^{(1)}, \ldots, x^{(n)}\right\}$ is the set of input examples in the training dataset, and $Y = \left\{y^{(1)}, \ldots, y^{(n)}\right\}$ is the corresponding set of labels. The $\psi(x)$ represents the output of the neural network given input x, λ_1 and λ_2 are user-specified parameter coefficients $\lambda_1 \in [0,1], \lambda_2 \in [0,1]$, N_v is a number of variables in a dataset, and W_{CO} are CancelOut weights.

The mean square error (MSE) loss can be utilized for regression tasks:

$$\mathcal{L}_{MSE}(X,Y) = \frac{1}{2n}\sum_{i=1}^{n}(y^{(i)} - \psi(x^{(i)}))^2 - \lambda_1 \, var(\frac{W_{CO}}{N_v}) + \lambda_2 \left\| \frac{W_{CO}}{N_v} \right\|_1 \tag{11}$$

The variance of the weights from the CancelOut layer $var(\frac{W_{CO}}{N_v})$ helps to stimulate diversity in the CancelOut layer, there $l1$ norm is used to introduce sparsity in W_{CO} weights and to constrain the variation to small weights. Also, $l1$ penalty restricts the model from selecting correlated features. Lastly, our feature selection approach supports all losses and does not require the realization terms (Table 1).

Table 1. Datasets

Dataset	Samples	Features	Target
Statlog (Australian Credit Approval)	690	14 (continuous, nominal)	Binary
Diabetes	442	10 (continuous, nominal)	Regression
MNIST	70.000	784 (continuous)	Multiclass

4 Experimental Results

In this section, we perform several experiments to evaluate different aspects of our CancelOut layer. In a first experiment, we examine the performance of our

algorithm for classification and regression tasks, using the Statlog (Australian Credit Approval) and Diabetes dataset [3] (Sect. 4.1). We choose these datasets, because they contain different feature types such as continuous and nominal. We compare the proposed features from a CancelOut NN with a Random Forest and a Gradient Boosting algorithm using k-fold cross-validations (stratified for the classification experiment).

In a second experiment, we add a dummy variable $(Y + \epsilon_1) \in X$ with normally distributed noise ϵ to the Australian Credit Approval dataset, in order to see if the proposed method is able to detect a feature that is highly correlated with the target feature (Sect. 4.2). Additionally, we introduce a "noisy" variable $X_{random} \sim N(0, 1) + \epsilon_2$ to assess, whether CancelOut discards irrelevant features. Note, $\epsilon_1 \neq \epsilon_2$.

Next, we compare feature importance characteristics from LASSO, SHAP [10], and CancelOut (Sect. 4.3). In a final experiment, we evaluate our model for the unsupervised scenario using a convolutional autoencoder (Sect. 4.4).

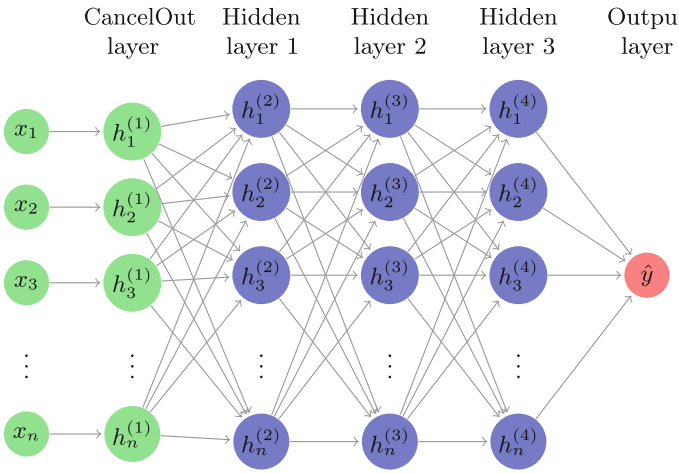


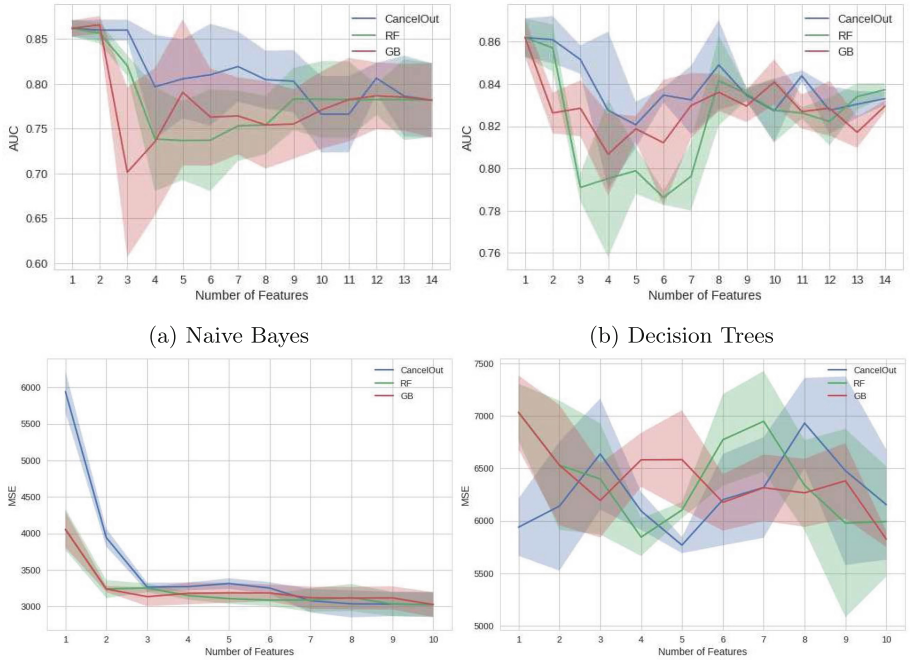
Fig. 3. A deep neural network architecture used for the experiments, where n is a number of variables.

In all experiments, we use a five layers DL model (Fig. 3) where the input layer is the CancelOut layer with the sigmoid activation function after each FC layer the ReLU activation function [5] was applied. Further, we use the optimization algorithm Adam [7] with learning rate 0.003, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-9}$. We utilize the early stopping technique to control overfitting of our model.

4.1 Feature Ranking

Classification Example. We illustrate the AUC scores for different sizes of feature subsets [3] in Fig. 4. The results are obtained by five-fold stratified cross-validation on the Australian Credit Approval dataset using Naive Bayes (Fig. 4a) and decision trees (Fig. 4b). Our algorithm achieves consistently good predictions for both classifiers and all feature set sizes. Moreover, CancelOut obtains superior predictions for small feature subsets. The variability in AUC is similar for all algorithms.

Regression Example. To evaluate CancelOut in context of a regression problem, we apply linear regression (Fig. 4c) and decision trees regression (Fig. 4d) on the diabetes dataset. We illustrate the MSE for different sizes of the reduced feature set in Fig. 4. We obtain the MSE scores again by five-fold cross-validation. CancelOut has disadvantages for linear regression if the number of features is smaller than three. However, our algorithm obtains competitive results for the



(a) Naive Bayes

(b) Decision Trees

(c) Linear Regression

(d) Regression Decision Trees

Fig. 4. A comparison of FS methods using Naive Bayes classifier (a) and decision trees (b) for the classification problem, and using linear regression (c) and decision trees regression (d) algorithms for the regression problem.

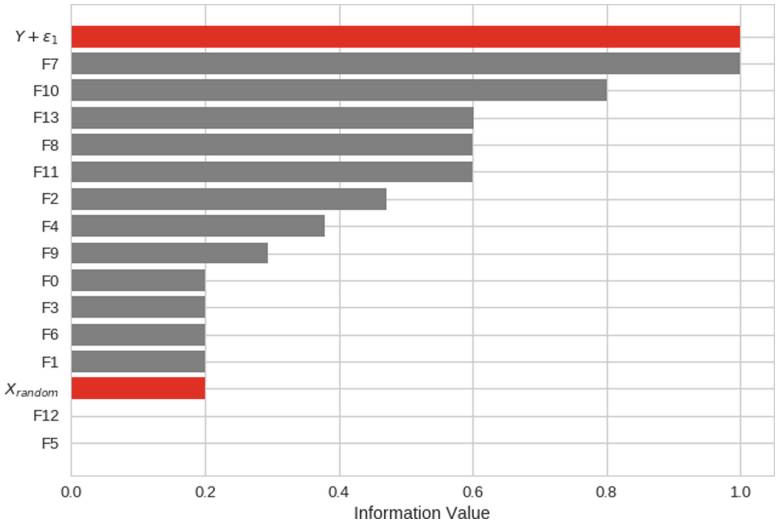


Fig. 5. A feature importance analysis for the Australian credit approval dataset [3] with two new features.

mid- and end-range of the reduced feature set size. The error measures for regression tasks with decision trees highly fluctuate with the number of selected features. Yet, our algorithm obtains best results for a small reduced feature set. These observations suggest that CancelOut can generally obtain feature sets that perform well in regression tasks.

4.2 Identifying Target and Noisy Features

In this experiment, we introduce two new features into the Australian Credit Approval dataset [3]. The first variable $Y + \epsilon_1$ is highly correlated to the target feature and the second is a random noise feature generated from the normal distribution $X_{random} \sim N(0, 1) + \epsilon_2$. The idea of the experiment is to show the ability of the proposed FR method to detect key and noisy features in the dataset.

In Fig. 5, we present a feature importance analysis for the augmented Australian Credit Approval dataset. The depicted values are the average of ten runs of an ANN obtained using the CancelOut layer. The analysis indicates that our method can successfully detect variables that are highly correlated to the target by evaluating them as the most important variable. Moreover, CancelOut mitigates the influence of noisy features by giving them low weights. This is shown exemplary by the low rank of X_{random}.

4.3 Evaluating Individual Feature Importance

We investigated several feature analysis methods for the diabetes dataset and summarized it into Fig. 6. The purpose of this comparison is to show that

CancelOut behaves comparable to other algorithms. Although there are differences in feature importance for the single features *age*, *sex*, *s2* and *s3*, the overall distribution of CancelOut weights is comparable to that of the SHAP and LASSO models.

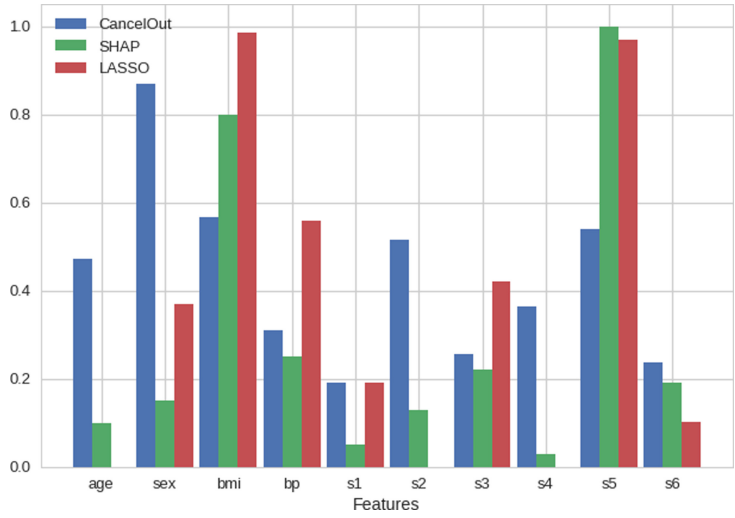
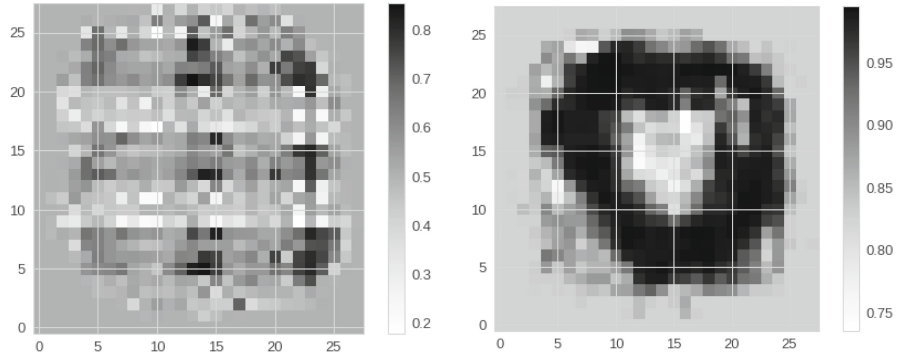


Fig. 6. A feature importance analysis for the diabetes dataset [3] using the proposed method (CancelOut), LASSO, and SHAP.

4.4 Unsupervised Feature Ranking Using Autoencoder

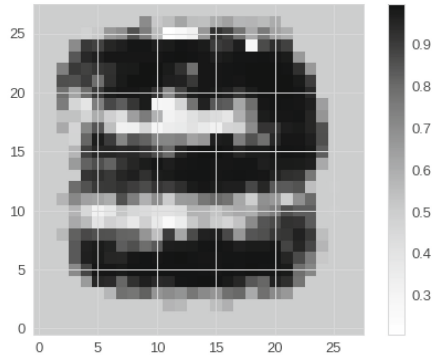
In this subsection, we demonstrate how the CancelOut layer can be utilized for unsupervised learning tasks with a convolutional autoencoder [11]. The architecture of the autoencoder consists of three convolutional neural network (CNN) layers in encoder and decoder parts, and the CancelOut as an input layer for the encoder. For this experiment, the MNIST dataset [8] is used.

Figure 7 shows CancelOut variable weights after training the convolutional autoencoder on the whole dataset (a), only on digit 0 (b), only on digit 3 (c), and only on digit 8 (c). CancelOut captures the most relevant regions of the picture for all four training sets. The information provided by CancelOut layer weights can help in model understanding, debugging, and adjustment, e.g. by introducing a "focus" on relevant features if a model performs poorly.

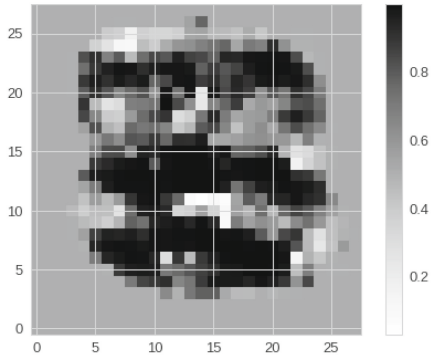


(a) Trained on whole MNIST dataset.

(b) Trained on digit 0



(c) Trained on digit 3



(d) Trained on digit 8

Fig. 7. $\sigma(W_{CO})$ values from the CancelOut layer after training using MNIST dataset: (a) the whole dataset, (b) images from 0 class, (c) images from 3 class, (c) images from 8 class.

5 Conclusion

In this paper, we introduced a novel feature ranking method using deep neural networks for various machine learning problems. The proposed method is extremely easy to implement, it can be done using all modern DL frameworks, and this method can be simply scaled. Due to the power of the neural networks, the presented approach learns linear and non-linear data dependencies. Also, the CancelOut layer can be applied for any data type and machine learning tasks, such as classification and regression problems or even for unsupervised problems as an input layer for an autoencoder. Finally, the proposed layer helps understand the data and its influence on the performance of DL models.

References

1. Chang, C.H., Rampasek, L., Goldenberg, A.: Dropout feature ranking for deep learning models. arXiv e-prints (2017)
2. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control, Signals Syst. **2**(4), 303–314 (1989). https://doi.org/10.1007/BF02551274
3. Dua, D., Graff, C.: UCI machine learning repository (2017). http://archive.ics.uci.edu/ml
4. Gilpin, L.H., Bau, D., Yuan, B.Z., Bajwa, A., Specter, M., Kagal, L.: Explaining explanations: an overview of interpretability of machine learning. arXiv e-prints (2018)
5. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016). http://www.deeplearningbook.org
6. Kasneci, G., Gottron, T.: LICON: a linear weighting scheme for the contribution ofInput variables in deep artificial neural networks. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, pp. 45–54. ACM, New York, NY, USA (2016). https://doi.org/10.1145/2983323.2983746
7. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv e-prints (2014)
8. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010). http://yann.lecun.com/exdb/mnist/
9. Li, Y., Chen, C.Y., Wasserman, W.W.: Deep feature selection: theory and application to identify enhancers and promoters. J. Comput. Biol. **23**(5), 322–336 (2016). https://doi.org/10.1089/cmb.2015.0189. PMID: 26799292
10. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 4765–4774. Curran Associates, Inc. (2017). http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf
11. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, J.L., (eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, pp. 318–362. MIT Press, Cambridge (1986). http://dl.acm.org/citation.cfm?id=104279.104293
12. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014). http://jmlr.org/papers/v15/srivastava14a.html
13. Zhang, Q., Nian Wu, Y., Zhu, S.C.: Interpretable convolutional neural networks. arXiv e-prints (2017)