The comparison of autoencoder architectures in improving of prediction models

A. V. Prosvetov
CleverDATA
Space Research Institute, Russian Academy of Science
prosvetov@gmail.com

Abstract. In our day many prediction models require to encode the series of events in a way that will allow to train the model and obtain the highest quality of predictions. The encoding of events depends on data domain and applied methods, however one can use neural network to encode the series of actions and obtain informative features for predictive models. We compared several architectures of neural networks in a task of feature extraction for predictive models. The comparison of architectures of neural networks was obtained on the field of sequence modeling, where the popularity of LSTM networks is dominant. We found, that the usage of appropriate event encoding allows to improve the quality of CNN based networks without using the modification of architectures.

1. Introduction
Rise of interest from business to the machine learning is largely due to the fact that the models are able to give a significant increase in profits in the areas related to the prediction of the behavior of complex systems. In particular, the complex system, whose behavior is beneficial to predict, is the man. Fraud detection at an early stage, churn tendency of customers – these tasks occur regularly and have already become classic in practice of Data Science. Of course, they can be solved by various methods, depending on the preferences of a particular expert and the business requirements.

We had the opportunity to use a neural network to solve the problem of the prediction of the behavior of people on the basis of email newsletters data. Our approach is aimed on generalizing the use of neural networks in the analysis of user actions (proposed, for example, in [1]) by applying autoencoders for features extraction. In our work we are comparing autoencoders, based on Convolution Neural Networks and LSTM Neural Networks, that has recurrent architecture and highly popular for sequence modelling. Recently many researches applied attention on comparison of convolution and recurrent architectures (for example [2]), and surprisingly found that convolution networks can reach state-of-the-art accuracy in audio synthesis, word-level language modeling, and machine translation [3–7], however and additional improvements of CNN required to obtain the significantly high results with sequence data, for example Temporal Convolution Networks. In our work we are trying to show, that even without improvement of CNN architecture one can reach high accuracy by using an appropriate way of sequence encoding. The proposed approach can be used in various practical problems, connected with sequence of actions independent from data domain.
2. Data and instruments

Our experiments based on data of newsletter opening during several years. The number of the newsletter subscribers is usually much larger than the number of customers of ordinary e-commerce site. Naturally, there is also a group of addresses, in which the probability of the first purchase is increased. These people can target additional advertising campaigns. Also, there is a group of recipients with high probability of unsubscript. There is no additional information about these potential customers, the brand knows only events of email opening and on what links the recipients passed.

| email receiving | 1 1 1 0 1 0 0 |
|-----------------|--------------|
| email opening   | 0 0 0 1 0 1 0 |
| link click      | 0 0 0 0 0 0 1 |
|                 | ...          |
| time delta      | 0.3 0.3 0.1 0.2 0.1 0.1 0.01 |

**Figure 1.** The encoding of action events of email recipient. The recipient got 3 emails, than opened an email, than got 1 more email, opened an email and clicked on the link in the email.

Recurrent neural networks (RNN) are well suited for processing a series of events [8]. RNN has gained tremendous popularity in language modeling and machine translation. Recurrent neural networks in their classical implementation have a number of fundamental problems, for example, vanishing gradients.

Today, LSTM neural networks are widely used, which managed to overcome several problems of classical RNN [9, 10]. Units in LSTM architecture have multiple switch gates in a memory cell thus are able to remember for longer time steps than RNN units can. A memory cell is composed of four main elements: an input gate, a neuron with a self-recurrent connection (a connection to itself), a forget gate and an output gate. The following equations describe how a layer of memory cells is updated at every timestep $t$.

At first step, one computes the values for $i_t$ (the input gate), and $\tilde{c}_t$ (the candidate value for the states of the memory cells at time $t$):

$$ i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) $$
$$ \tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) $$

On the second step, one computes the value for $f_t$ (the activation of the memory cells’ forget gates at time $t$):

$$ f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) $$

Given the value of the input gate activation $i_t$, the forget gate activation $f_t$ and the candidate state value $\tilde{c}_t$, one can compute $C_t$, the memory cells’ new state at time $t$:

$$ C_t = i_t \tilde{c}_t + f_t C_{t-1} $$

With the new state of the memory cells, one can compute the value of their output gates and, subsequently, their outputs:

$$ o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) $$
$$ h_t = o_t \tanh(C_t) $$
In these equations:

\( x_t \) is the input to the memory cell layer at time \( t \)

\( W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o \) and \( V_o \) are weight matrices

\( b_i, b_f, b_c \) and \( b_o \) are bias vectors

To compare results of LSTM neural network, we use Convolution Neural networks [11] and a neural network with a combination of layers of both types. CNNs are basically consist of several layers of convolutions with nonlinear activation functions applied. During the training phase, a CNN automatically fits the parameters for filters based on target variable. Each filter composes a local patch of lower-level features into higher-level representation. Also, the location invariance is reached by using filters in CNN.

We trained neural networks to obtain features that can be used in predictive models on the next step. Therefore, for subsequent use the features we decided to train on events without knowing their result. If we would train the model on a specific target action, then this model would have information about the future – whether the person will make a target action in the future. Thus, this kind of model will be able to work well with only one type of behavior, and the features obtained from such a model will be difficult to use in other models. If we plan to use the result of the model in other predictive models, it is necessary to carefully avoid data leakage: if the information about the future passes unintentionally, the new model will be overfitted. One of the way to reduce the sequence of events into numerical features and avoid all mentioned above problems is in training of autoencoder [12], which then could be used in other predictive models.

The autoencoders are constructed in a way, that they have the same dimensions at the input and output, but the dimension in the middle is much smaller. This restriction forces the neural network to search of generalizations and correlations in the sequence of events, and therefore autoencoders are used to generalize the incoming data.

In the training of autoencoders, the principle of back propagation of the error is used, as in supervised learning, but one can require that the signal at the input and the signal at the output of the network be with minimal distinction. In the end, one can get unsupervised training without using the information about the target. In our case we used cross-entropy loss function:

\[
L = -y \log(p) + (1 - y) \log(1 - p)
\]

To prepare training set we used the following one-hot encoding of recipient’s actions (fig. 1): every action is presented as a binary vector, thus the sequence of actions will be sparse binary matrix. In addition we added information about time delay between actions in the following way:

- we found the time delay from last event for every sorted events of time series in the correspondent time scale;
- we used natural logarithm on values from the obtained array;
- we divided all values of the obtained array on maximum.

3. Results

We tried several architectures of neural networks: LSTM, CNN and a combination of CNN and LSTM. For the first case we used several LSTM layers as the encoder and combination of LSTM layers and layers with repetitions of input tensors as decoder, and as a "bottleneck" we used several vanilla fully-connected layers of neurons with Dropout and Batch Normalization. For the second case we used several sequence of CNN layers and Max Pooling layers in the encoder and sequence of CNN layers and Upsampling layers in the decoder with the vanilla "bottleneck". For the combined case we used at first several CNN and Max Pooling layers, than several LSTM layers as encoder and as decoder we used combination of LSTM layers and layers with repetitions of input tensors at first and a sequence of CNN layers and Upsampling layers at the end. Every neural network was tuned independently to obtain the highest results on training data set.

The comparison of obtained metrics is presented in table 1.
Table 1. The comparison of training metrics of autoencoders based on different architectures of neuron networks.

|                                      | LSTM    | CNN      | CNN+LSTM |
|--------------------------------------|---------|----------|----------|
| Binary cross-entropy loss            | 0.1398  | 0.0968   | 0.0957   |
| Roc Auc of predictive model, trained on extracted features | 0.81–0.84 | 0.83–0.86 | 0.82–0.87 |

After training, the encoder translates the sequence of actions for each recipient into a feature vector, and one is able to obtain a Word2vec analog for a series of events for each recipient. The resulting vector can be used in other predictive models, clustering, searching for Look alike recipients, and for anomaly detection.

![Figure 2](image_url)  

**Figure 2.** Projection onto a two-dimensional space (t-SNE) of behavior of recipients obtained with the CNN autocoder. Red points are the recipients who made a purchase in the near future, the blue points are the recipients who did not make the purchase.

Projection onto a two-dimensional space (t-SNE) of behavior of recipients obtained with the CNN autocoder can be found on the fig. 2. It can be seen that there are areas in which the inert recipients are predominant, and there are areas with a high concentration of future buyers. In comparison with the presented picture, the projection of LSTM features on two dimensions gives a form of a white noise.

In our case, we built a model that predicts the probability of recipient’s buying based on features form autoencoders. The model trained on a number of main features shown roc-auc metric in the range 0.74–0.77. The use of an autoencoder based on CNN network for the described problem produced a better result, but the LSTM-based autoencoder provided a vector of smaller dimension and using both types of features we reached the improvement of Roc-auc metric: with the added vectors responsible for human behavior obtained using both autoencoders, roc-auc metric reached range 0.84–0.88 (fig. 4). In the case of training on the features based on the autoencoder with combination of LSTM and CNN layers, Roc-auc metric is reached the range of 0.82–0.87.
In the list of the most significant features, the dominant position is occupied by the features from two autoencoders: based on LSTM and CNN.

The comparison of ROC curves of two models: roc-auc metric of a model trained on the main features is reached 0.74–0.77, roc-auc metric of a model trained on the main features and the features obtained by the autoencoder is reached 0.84–0.88.

4. Conclusions
In our study we found that autoencoders based on CNN architecture are able to extract more sufficient features from the sequence of people's behavior, that autoencoders based on LSTM architecture can. In our case, the sequence of the recipients' behavior was encoded with the help of auto-encoders on LSTM and on CNN-architectures and the obtained loss for CNN autoencoder was significantly better than loss of LSTM autoencoder. Roc-auc metric of the predictive model, trained on features, extracted from the mentioned autoencoders, shown the higher results in cases, when CNN layers were
used. The feature importance of trained predictive model shown the domination of features, obtained using CNN autoencoder. The fact that the CNN-based autoencoder received better results in our case than LSTM autoencoder, allows to suggest that the behavior patterns are used to extract more informative features for this task than the features based on temporal relationships between events. Nevertheless, on the prepared data set, it is possible to use both approaches, and the total effect of sharing the features of both autoencoders gives an increase to roc-auc metric.

The comparison of architectures of neural networks was obtained on the field of sequence modeling, where the popularity of LSTM networks is dominant. The usage of appropriate event encoding allows to improve the quality of CNN based networks without using the modification of architectures.

The proposed way of encoding of objects behavior can be used in the models that require features to describe the actions of complex systems: the anti-fraud detection, the churn prediction, the anomaly detection, etc.

References
[1] Zołna K and Romanski B 2016 Proceedings of the 33rd International Conference on Machine Learning (New York, NY, USA, JMLR: W&CP) 48
[2] Shaojie Bai, J. Zico Kolter and Vladlen Koltun 2018 An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling arXiv:1803.01271v2
[3] van den Oord A, Dieleman S et al 2016 WaveNet: A generative model for raw audio. arXiv:1609.03499
[4] Kalchbrenner N, Espeholt et al 2016 Neural machine translation in linear time. arXiv:1610.10099
[5] Dauphin Y N, Fan A, Auli M and Grangier D 2017 Language modeling with gated convolutional networks In ICML
[6] Gehring J, Auli M, Grangier D and Dauphin Y 2017 A convolutional encoder model for neural machine translation In ACL
[7] Gehring J, Auli M, Grangie, D, Yarats D and Dauphin Y 2017 Convolutional sequence to sequence learning In ICML
[8] Elman J L 1990 Finding structure in time Cognitive science 14.2 179–211
[9] Hochreiter S and Jürgen S 1997 Long short-term memory Neural computation 9.8 1735–80
[10] Gers F A, Schmidhuber J and Cummins F 2000 Learning to forget: Continual prediction with LSTM Neural computation 12(10) 2451–71
[11] LeCun Y et al 1998 Gradient-based learning applied to document recognition Proceedings of the IEEE 86.11 2278–2324
[12] Bourlard H and Yves K 1988 Auto-association by multilayer perceptrons and singular value decomposition Biological cybernetics 59.4-5 291–94