Expert Systems With Applications 102 (2018) 207–217

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Predicting mortgage default using convolutional neural networks

Håvard Kvamme[a,*], Nikolai Sellereite[b], Kjersti Aas[b], Steffen Sjursen[c]

[a] Department of Mathematics, University of Oslo, Niels Henrik Abels hus Moltke Moes vei 35, Oslo 0851, Norway
[b] Statistical Analysis, Machine Learning and Image Analysis, Norwegian Computing Center, Gaustadalleen 23a, Oslo 0373, Norway
[c] Group Risk Modelling, DNB ASA, Dronning Eufemias gate 30, Oslo 0191, Norway

ARTICLE INFO

Article history:
Received 15 August 2017
Revised 17 February 2018
Accepted 18 February 2018
Available online 19 February 2018

Keywords:
Consumer credit risk
Machine learning
Deep learning
Mortgage default model
Time series

ABSTRACT

We predict mortgage default by applying convolutional neural networks to consumer transaction data. For each consumer we have the balances of the checking account, savings account, and the credit card, in addition to the daily number of transactions on the checking account, and amount transferred into the checking account. With no other information about each consumer we are able to achieve a ROC AUC of 0.918 for the networks, and 0.926 for the networks in combination with a random forests classifier.

1. Introduction

The ability to discriminate bad customers from good ones is important for banks and other lending companies. A small improvement in prediction accuracy may result in a large gain in profitability. Early identification of high risk consumers may aid the prevention of loan defaults and help the consumers to better manage their personal economy.

In credit scoring, one builds a model for the correspondence between default and various loan obligor characteristics based on a relevant sample of people, and use this model to predict the probability that a person will repay his debts.

There is an extensive literature on credit scoring, both for assessing private loans (Butaru et al., 2016; Chi & Hsu, 2012; Khandani, Kim, & Lo, 2010; Sousa, Gama, & Brandão, 2016) and corporate loans (Jones, Johnstone, & Wilson, 2015; Ravi Kumar & Ravi, 2007). Some recent work include Abellán and Castellano (2017); Chen, Zhou, Wang, and Li (2017); Xia, Liu, Li, and Liu (2017), and Barboza, Kimura, and Altman (2017). For an overview and comparison of papers, see García, Marqués, and Sánchez (2014) and Lessmann, Baesens, Seow, and Thomas (2015).

All the papers above attempt to model delinquencies and defaults by applying machine learning algorithms to a set of explanatory variables. While there are variations in the information that is available to researchers (see e.g. Lessmann et al., 2015), the constructed explanatory variables tend to be quite similar. Papers typically use information from credit bureaus, such as number of outstanding accounts, delinquent accounts, and balance on other loans; individual account characteristics, such as current balance of the individuals accounts and monthly income; and demographic data, such as age and marital status. Butaru et al. (2016) also include macroeconomic variables, such as interest rates and unemployment statistics, as an attempt to make the delinquency model generalize better over longer periods of time.

As all these papers use similar explanatory variables, the researchers commonly explore differences between scoring models rather than the benefit of adding new explanatory variables. There are however some exceptions: Khandani et al. (2010) explore the benefit of adding information from detailed purchase volumes to their models. This includes travel expenses, gas station expenses, bar expenses, etc. Chi and Hsu (2012) also introduce consumer transaction data through an aggregated measure called average utilization ratio of credit.

In this paper we further investigate how transaction data can be used for credit scoring. In a joint research with Norway's largest financial service group, DNB, we use transaction data to predict mortgage defaults. In 2012, the average Norwegian made 323 card transactions, where 71% of the value transferred was through debit payments (Norges-Bank, 2012). Hence, transactional data may provide a useful description of user behavior, and subsequently consumer credit risk.

The transaction information consists of the daily balances on consumers' credit, checking, and savings accounts, in addition to

* Corresponding author.
E-mail addresses: haavakva@math.uio.no (H. Kvamme), nikolai.sellereite@nr.no (N. Sellereite), kjersti.aas@nr.no (K. Aas), sasjurse@gmail.com (S. Sjursen).

https://doi.org/10.1016/j.eswa.2018.02.029
0957-4174/

the daily number of transactions on the checking account, and the amount transferred into the checking account. Our dataset is thus a collection of time series for each customer. We do not use any of the common covariates mentioned above, as our goal is to investigate the value of the information available in the time series data. In a production setting, our model can be combined with existing risk models to improve the overall performance.

The idea behind this paper is to use deep learning, or deep neural networks (LeCun, Bengio, & Hinton, 2015), to predict mortgage defaults. Deep learning have had a dramatic impact in fields like image classification, text mining, and speech recognition. Common to these three fields is that the data is unstructured. Thus, the original data (pixels, letters, words, frequencies) needs to be transformed into meaningful covariates before they can be passed to a classical algorithm. Until recently, such tasks have been solved by the manual creation of informative covariates from the data. Deep neural nets, on the other hand, process the data in a sequential manner, learning multiple levels of abstraction. Hence, a deep net use data to "learn" how to create good explanatory variables for the problem at hand.

In this paper we apply a type of deep neural networks denoted convolutional neural networks (CNNs). To the extent of our knowledge, we are the first to apply CNNs to consumers' account balances to predict mortgage defaults. Transaction data has previously been used for credit scoring, see e.g. Khandani et al. (2010). However, most existing work relies on heuristic hand-crafted feature design. It is often hard for us humans to figure out appropriate features or covariates for credit scoring. Hence, the purpose of this study is to use a convolutional neural network to automate feature engineering from the raw time series, in a systematic way. The learned features resulting from such a model may be viewed as the higher level abstract representation of the low level raw time series signals.

There have been some successful attempts applying deep neural nets to time series data in other application areas. In the field of human activity recognition, Ordóñez and Roggen (2016); Yang, Nguyen, San, Li, and Krishnaswamy (2015), and Hammerla, Halloran, and Ploetz (2016) use sensor data to recognize activities improving the state-of-the-art. Cui, Chen, and Chen (2016); Prasad and Prasad (2014); Zheng, Liu, Chen, Ge, and Zhao (2014), and Le Guennec, Malinowski, and Tavenard (2016) present network structures for time series classification across multiple domains. Also, Sirignano, Sadhwani, and Giesecke (2016) assess mortgage risk using a deep net on 294 explanatory variables. However, on the contrary to our work, they use multilayer perceptrons instead of CNNs, and the majority of the variables are loan-level feature and performance variables. The deep net is mainly used to identify complex interactions between the input variables.

Much work in credit scoring is related to regulatory frameworks such as the Basel Accord. The Basel II regulations require transparency in the loan-granting process. Due to their non-linear structure, CNNs are usually considered as black boxes. That is, it is usually not obvious what exactly makes them arrive at their predictions. However, since this lack of transparency in many applications is considered a major drawback, the development of methods for explaining and interpreting deep learning models has recently attracted increased attention, see e.g. Lundberg and Lee (2016); Samek, Wiegand, and Müller (2017); Shrikumar, Greenside, Shcherbina, and Kundaje (2016), and Ribeiro, Singh, and Guestrin (2016). Hence, it is not obvious that CNNs will remain inappropriate for building regulatory models in the future. Meanwhile, a credit scoring system based on a CNN may be useful in many other applications. Financial institutions may e.g. use them in their own internal risk estimation or as part of a validation exercise (Pillar 2 of the Basel framework).

A lender commonly makes two types of credit decisions: first, whether to grant credit to a new applicant, and second, how to deal with existing applicants, including whether to increase their credit limits. The first problem is denoted application scoring and the latter behavioral scoring. The majority of previous work focus on application scoring, while prior work on behavioral scoring is much less developed, see e.g. Thomas (2000) and Kennedy, Mac Namee, Delany, O'Sullivan, and Watson (2013). The methodology proposed in this paper may be used both for application and behavioral scoring. Consumer transaction histories contain implicit repayment behavior, making them suitable for behavior scoring. However, as the new Revised Payment Service Directive (PSD2) becomes implemented across the EU and the European Economic Area during 2018, the banks (and other lending institutions) will have access to transaction data even for new customers. Hence, this kind of data may also be used for application scoring.

The remainder of the paper is organized as follows. In Section 2 we describe our dataset and how it was processed before fitting the neural networks. In Section 3 we introduce convolutional neural networks, and show how they were applied to our problem. In Section 4 we evaluate the performance of the proposed algorithms. Finally, in Section 5, we summarize our findings.

2. Data

This study is a collaboration with the largest Norwegian financial service group, DNB, using data from their banking services. The dataset consists of a sample of 20,989 customers who either previously had a mortgage, or got approved for a mortgage at some point between January 2012 and April 2016. For every customer we have the daily balances on their checking account, savings account, and credit card, in addition to the daily number of transactions on the checking account. We also know the daily amounts that are transferred into the checking account (e.g. from salary, payments from friends, etc.). Hence, we have a set of five time series for each customer. Finally, adding the sum of checking account, savings account, and credit card as a new time series, we end up with a total of six series. A summary of the six series can be found in Table 1.

For a customer to be granted a mortgage by DNB, he or she is required to open a checking account at the bank, given that this criterion is not already satisfied. There are however no requirements as far as savings accounts and credit cards are concerned. As a result, there are many missing series in the dataset.

Our definition of default is the one used in Basel II, i.e. that the obligor is past due for more than 90 days on the obligation to the bank. Housing prices in Norway have generally increased steadily since 2003, and thus, the mortgage market has seen few defaults (Finanstilsynet, 2016). The lack of defaults poses a problem due to the fact that our algorithms need large datasets to generalize well. In addition to this, the large class imbalance (ratio between non-defaults and defaults) is commonly regarded as a problem for classification algorithms. Both issues are addressed in later sections.

2.1. Training and test set

We divide our data into a training set and a test set. For training, we extract transaction histories from the time period December 31, 2012 to December 31, 2013 and use the outcome (default / non-default) in the period from January 1, 2014 to January 1, 2015 as response variables. A subset of the training set was held out from training and used as a validation set for tuning of our algorithms. For testing, we extract transaction histories from the time period February 28, 2014 to February 28, 2015, and response variables from the period March 1, 2015 to March 1, 2016. There are no customers who appear in both the training and test set, meaning that our study is both "out-of-time" and "out-of-sample". Cus-

Table 1
Time series.

Series	Abbrev.	Explanation
Checking account	ch	Balance on the checking account.
Savings account	sa	Balance on the savings account.
Credit card	cc	Balance on the bank issued credit card.
Checking transactions	trch	Number of transaction on the checking account.
Into checking	in	Sum of transactions into the checking account.
Sum	sum	Sum of series ch, sa, and cc.

The time series used in this paper. Each time series consists of values aggregated to a daily level.

Table 2
Data proportions for training, validation, and test sets.

Dataset	Defaults	Non-defaults	Total	Default type
Training	1298	11,398	12,696	90 or 60 days past
Training augmented	95,647	841,247	936,894	90 or 60 days past
Validation	329	6043	6372	90 or 60 days past
Test	96	1825	1921	90 days past

tomers for whom we did not have a full year of transaction data, were removed both from the training and the test set. However, we evaluate the performance of our methods on customers with missing data in Section 4.5. For a further specification of the datasets, see Table 2.

2.2. Preprocessing time series

Convolutional neural networks require the input to be scaled. Hence, for images it is common to normalize the pixels to lie in the range [0, 1] (Goodfellow, Bengio, & Courville, 2016). Pixels however have the benefit of all being in [0, 255] and somewhat evenly distributed in this interval. On the contrary, there are great differences in the magnitude of the accounts. If we scale all series based on the most extreme account values, most of the series will have very small variations, making it hard for our neural net to learn from the data. Therefore, the time series were scaled such that each individual series is in the range [0, 1] through

$$\mathbf{x} = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}, \tag{1}$$

where $\mathbf{x}$ denotes *one* time series. With this scaling, the neural net receives inputs of similar magnitude. However, this comes at the cost of removing the magnitude of the individual series. Hence, the network can only find information in the relative patterns of the series, and has no knowledge about the actual size of the different accounts. Missing series were treated as accounts without any movements.

3. Methods

While there is an extensive literature on time series classification, we have chosen to focus only on the subset of algorithms that falls under deep learning. For a review of other state-of-the-art methods, see e.g. Bagnall, Lines, Bostrom, Large, and Keogh (2016).

In the following, we will first, in Section 3.1, introduce convolutional neural networks and how they were applied to our credit scoring problem. Then, in Section 3.2, we will discuss methods used to avoid overfitting and how we approached the class imbalance problem.

3.1. Convolutional neural networks

Deep learning arguably owes much of its success to its extent of modularity. The framework is based on combining differentiable transformations, where each such transformation is commonly referred to as a *layer*. The way one organizes the layers is called the *network architecture*. When fitting such a network to a dataset, or performing the "training" as it is denoted in machine learning, it is necessary to define a loss function that can be optimized. The loss function does not necessarily need to be the objective we actually want to optimize, but rather a function that indirectly optimizes our true goal (Goodfellow et al., 2016). For binary classification it is common to use the binary cross entropy

$$\text{Loss} = -\sum_i \{y_i \log p_i + (1 - y_i) \log(1 - p_i)\}, \tag{2}$$

where y_i denotes the true class label as {0, 1}, and p_i denotes our prediction for customer i in [0, 1]. Eq. (2) is the same as the negative log likelihood used in logistic regression. Note that the loss decreases as the p_i's move closer to the y_i's, hence reducing the loss should improve our objective.

During training the data is passed through the network, the loss is calculated, the gradients are computed with respect to all the parameters in the network, and the parameters are optimized through some version of gradient descent. The gradients can be obtained in a nice way using backpropagation (LeCun, Bottou, Orr, & Müller, 1998), which basically is application of the chain rule for differentiation. Backpropagation calculates the gradients in a sequential manner, such that the gradients of each transformation can be derived solely from the error passed back from the succeeding layer. The main takeaway here is that layers can be implemented independent of each other, simplifying the process of combining them in an architecture.

Convolutional neural networks (CNNs) constitute the subset of neural networks that contain convolutional layers. However, they typically contain other layers as well. The following sections describe the different types of layers that have been used in our neural networks.

3.1.1. Convolutional layers

The first two layers of our network are presented in Fig. 1. The left block shows an input time series that is 365 days long. For now, assume that the dataset only consist of a single time series per customer (instead of six).

A convolutional layer consists of multiple *filters* that are applied to the input series. A filter is simply a vector (or matrix) consisting of weights that need to be optimized. The first convolutional layer has filters of length 9, as shown in the figure. The application of such a filter to the input series is just a sum over the element-wise product of the filter weights and the time series, resulting in

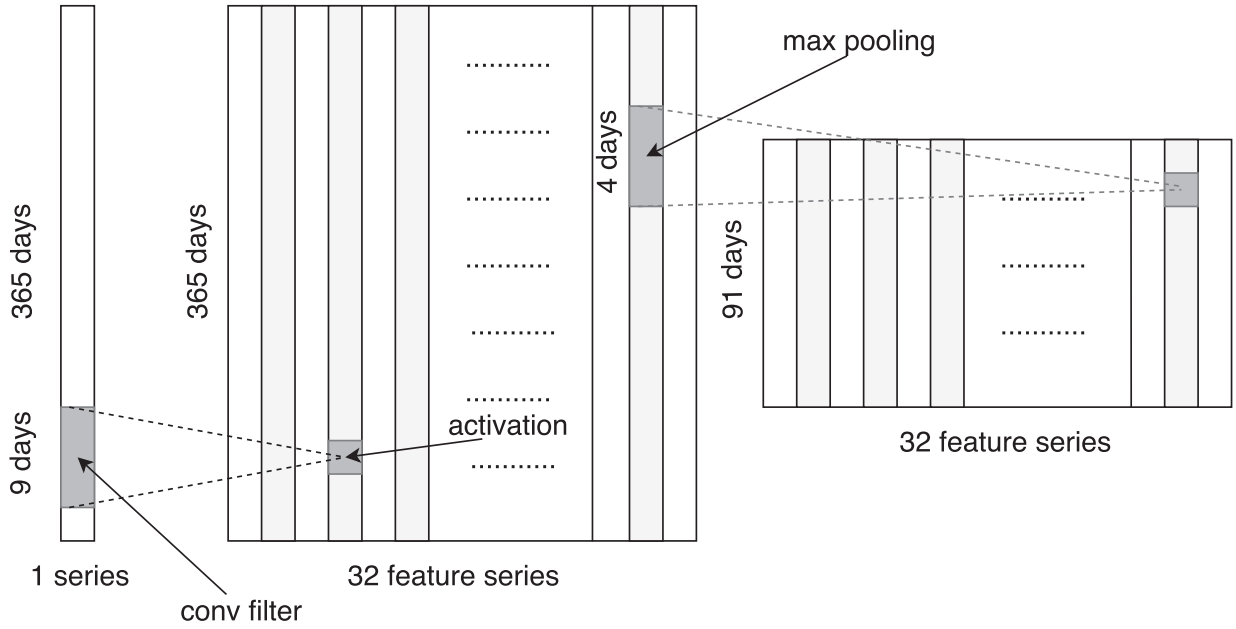

Fig. 1. The first convolutional layer and first pooling layer of our model. The left block is the input series. The gray area marks the part of the series for which a convolutional filter is currently being applied. The middle block shows the convolved features (or activations) resulting from the 32 different convolutional filters. The right block shows the result of applying max pooling to the middle block.

a convolved feature (or activation). A convolved feature is therefore a measure of the correlation between the filter and the relevant part of the input. The middle block of Fig. 1 displays all the convolved features from the input, using 32 different filters. Each filter is slided across the input to generate as many convolved features as the length of the time series[1]. Due to the fact that the 32 filters are randomly initialized, they end up extracting different information from the input.

After applying the filters to a time series, the convolved features are passed through a ReLU transform, which is simply $\max\{0, x\}$. Such *activation functions* are applied to make the transformation non-linear, and they are commonly considered a part of the convolutional layer rather than a separate layer. See Gu et al. (2017) for a discussion of other activation functions.

3.1.2. Max pooling layers

A convolutional layer preserves the resolution in the temporal dimension. There can however be some benefits of reducing the temporal resolution, such as reducing the number of parameters in the next layer, and introducing some shift-invariance (Gu et al., 2017). As a result, many CNNs include so-called max pooling layers, which simply replace a set of values with their maximum. Note that this layer does not introduce any additional parameters.

We use max pooling after each convolutional layer. An example is shown to the right in Fig. 1, where four values of each column of the convolved features are pooled into one. As shown in the figure, pooling is applied separately for each feature series. Thus, the pooling layer preserves the number of feature series, while reducing the temporal resolution. See Gu et al. (2017) for a discussion of other pooling layers.

3.1.3. Fully connected layers

To perform the actual classification, it is necessary to combine the convolved features into a binary classifier. This is typically done through one or more *fully connected* (or *dense*) layers, which form a multilayer perceptron (a classical neural net). A dense layer requires a one-dimensional input, so the columns of the last layer are concatenated into a single vector x. The transformation is then simply the inner product with a weight matrix, $z = x^T W$, with some non-linear activation function (typically ReLU). The number of columns in W determines the dimension of the output. Our final layer has only two outputs, one for each class, and therefore a softmax activation function is applied to ensure that the predictions are in the interval [0, 1].

$$\text{Softmax}(z)_c = \frac{e^{z_c}}{\sum_j e^{z_j}}, \tag{3}$$

where c and j refers to classes. Note that the softmax can be applied for an arbitrary number of classes, but for binary classification it it equivalent to the logit function. As a result, in combination with our choice of loss function, this final layer is equivalent to a logistic regression.

3.1.4. Network architecture and training

In our network architecture we alternate between a convolutional layer and a max pooling layer two times, followed by two fully connected layers. For an input of length 365 days, this results in 199,234 weights that need to be optimized. The whole structure is displayed in Fig. 2, were we also show the length of the filters and number of filters for the convolutions, the size of the pooling kernels, and the number of output nodes in the dense layers. This architecture was developed manually by evaluating on a validation set[2], and the hyper parameter search was therefore not as rigorous as with e.g. a grid search. However, it was found that a variety of model configurations resulted in quite similar performance. In particular, further increasing the number of nodes, filters, or layers did not really seem to affect the performance.

We train one network for each of our six time series, completely independently, using the Adam optimization algorithm (Kingma & Ba, 2014) with default parameters and batch size of 512. The resulting six predictions are averaged to give a final prediction

[1] Zero-padding is used to preserve the temporal dimension (Goodfellow et al., 2016).

[2] Note that the validation set is not the same as the test set.

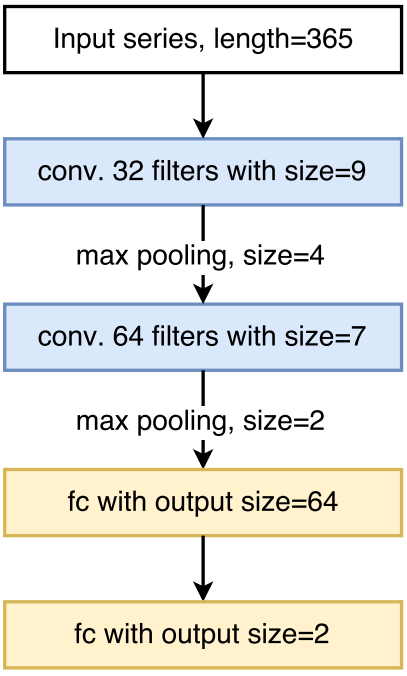

Fig. 2. Our convolutional neural net. The blue layers are convolutional layers and the yellow layers are fully connected layers. All activations are ReLU, with the exception of softmax in the final layer. Dropout, with rate 0.5, is performed between the last two layers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

for each customer. When computing the averages, predictions corresponding to missing series are discarded. We also experimented with averaging the log-odds instead of the predictions, which resulted in a slight performance decrease.

We also tried an alternative model where we train a network on all six series simultaneously. Thus, the input is 365×6 rather than 365×1, meaning that the first convolutional filter will have size 9×6, rather than 9×1. Apart from this, the architecture is identical to the architecture for the individual series. By combining all series in the input, the network has the potential to learn interactions between a customer's time series. However, as this model is more complex, it will require more data than the previous model.

In addition to the two models presented here, we have also tried deeper and more shallow models, as we will get back to in Section 4.2. In early stages of the project, we even experimented with recurrent neural networks in the form of Long Short-Term Memory networks (LSTM). They were found to be outperformed by the CNNs, in addition to being much more computationally expensive. Hence, they were dropped from further investigation.

3.2. Overfitting and class imbalance

In the following sections we explain the techniques we have used to prevent the models from overfitting and how we handled the large class imbalance between defaulting and non-defaulting customers.

3.2.1. Regularization

Deep neural networks are commonly overparameterized, making them prone to overfitting (Gu et al., 2017). Subsequently, regularizing techniques have been introduced to cope with these problems. Among the more common is the use of a validation set for monitoring the training process. We used this set to stop the gradient descent iterations when the net starts to overfit. This technique is commonly referred to as *early stopping*.

Another common form of regularization is *dropout* (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014), which simply sets activations to zero with a given probability during training. This prevents units in the network from co-adapting too much. At test time, the dropout layer scales the activations according to the dropout rate. We use dropout with rate 0.5 between the two last layers, but it can in practice be applied between any two layers.

See Goodfellow et al. (2016) and Gu et al. (2017) for a further review of regularization.

3.2.2. Data augmentation

The problem of predicting mortgage defaults is highly imbalanced in that the number of defaults is very small compared to the number of non-defaults. As a classifier's performance commonly deteriorates under imbalance, we under-sample the non-defaulting customers to make the two classes more similar in size. Moreover, to increase the number of defaults in the training set, we use delinquencies of 60 days in addition to those of 90 days. Hence, we ended up with a training set consisting of 12,696 customers, 10% of whom where characterized with default.

Deep neural nets commonly require more data than classical machine learning algorithms to perform well. As a result, data augmentation schemes have been explored in the literature (Goodfellow et al., 2016; Krizhevsky, Sutskever, & Hinton, 2012). Most of this work has been on images, and does not necessarily generalize well to other data sources. However, recently some attempts of performing data augmentation on time series have been proposed. Le Guennec et al. (2016) explore the possibilities of training a CNN in a semi-supervised manner using time series from other datasets. Cui et al. (2016) propose a sliding window technique where they split the time series in many overlapping slices, and train a classifier using each slice as an individual series. This is in some sense similar to other time series prediction tasks where the dataset originally consists of many overlapping series of data (e.g. Ordóñez & Roggen (2016)).

In this paper, we use a slight variation of the sliding windows approach of Cui et al. (2016). Our approach can be described as follows. For each defaulting customer we first define a training period which ends one month before the default, and starts two years earlier. If the customer did not exist in the dataset at the start of this period, the start was set to the earliest date the customer existed. We then extract the time series corresponding to the first 365 days of the training period and use that as an observation. By jumping a fixed number of days, called *stride* in the machine learning literature, we can get a new observation that is partially overlapping with the previous observation. This process is repeated until the end of our defined training period is reached. In this way, we get a training set consisting of multiple overlapping series from each defaulting customer. This augmentation scheme is illustrated in Fig. 3. The gray area shows the defined training periods for three different customers, and the red dots mark the time of their defaults. For id: 2 we show three of the observations (windows) that are added to the training set. Note that the first of the corresponding periods ends one year before the default occurs, while the last ends one month before. Thus both these observations, and all in between, fit our objective of predicting a default in the following 365 days.

We also need to perform a similar augmentation for the non-defaulting individuals. Our first experiments showed that sampling non-defaulting and defaulting customers from different time periods made our net learn different characteristics of these periods instead of the actual objective. Hence, to avoid this sampling bias, we had to sample the start and end dates for the non-defaulting customers from the ones of the defaulting customers.

Ideally one would use strides of 1 as this would give most diversity in the training data. However, our preliminary analysis

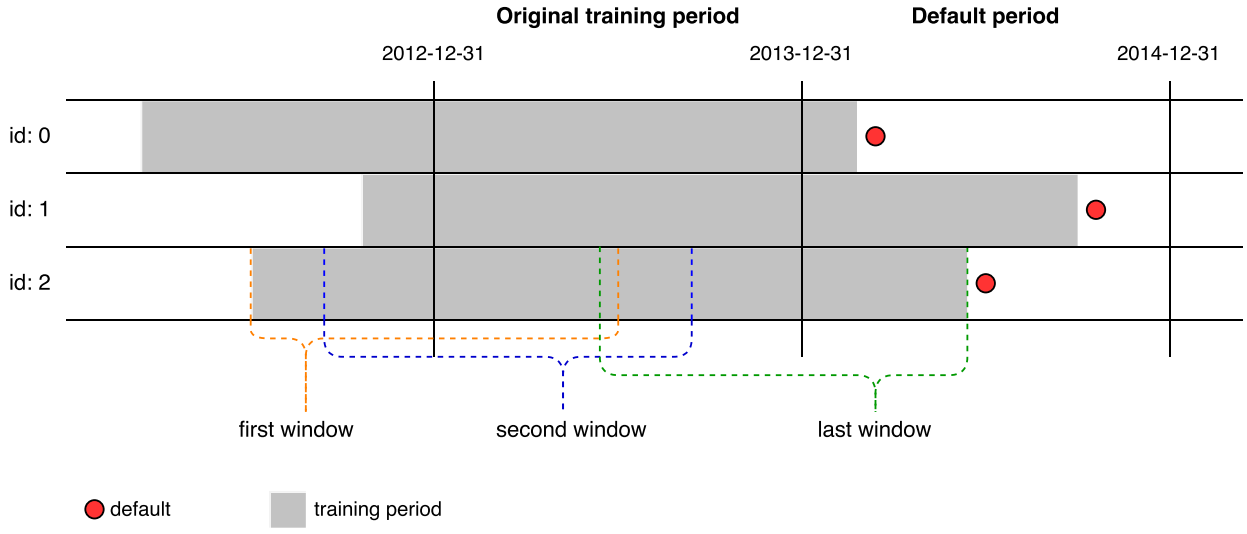

Fig. 3. Illustration of our data augmentation scheme. The gray area identifies the time period from which we extract data, and the red dots show the time of default. The sliding window approach is illustrated for id: 2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

showed that strides of 1 did not give better results than strides of 5. Hence, since the memory usage of the latter is only a fifth of the first, we decided to use strides of 5 in our final analysis.

In Table 2 we have shown the size of our training set, the training set augmented with the sliding windows, the validation set, and the test set. The financial group DNB does not want to disclose its true default ratio. Hence, to generate the test set, 2000 customers were randomly sampled in such a way that the fraction of defaulting customers was 5%. After removing customers with missing history, we ended up with a test set of 1921 customers. The size of the test set is quite small, which results in larger variance in our results. Note however that we have experimented with different test sets from different time periods, and the results seem to be quite consistent.

Table 3
AUC for different CNN architectures.

Model	Mean AUC	Std AUC	AUC for average pred.
Full, 1 conv layer	0.8884	0.0055	0.8979
Full, 2 conv layers	0.8879	0.0041	0.8982
Full, 3 conv layers	0.8897	0.0048	0.9015
Indiv, 1 conv layer	0.9022	0.0022	0.9043
Indiv, 2 conv layers	**0.9146**	**0.0025**	**0.9184**
Indiv, 3 conv layers	0.9106	0.0033	0.9148

AUC mean and standard deviation for 10 experiments for each of six different CNNs. "Full" refers to the approach with all time series as input, and "Indiv" to the approach where we compute the average of the predictions from six individually trained models. The rightmost column gives the AUC of the predictions averaged over the 10 trained models.

4. Experiments

In the following, we first present the measures used to evaluate the performance of our models. We then describe our experiments and their results.

4.1. Evaluation measures

The Receiver Operating Characteristic (ROC) curve is a common measure for evaluating a classifier's ability to separate two classes. The ROC curve is a plot of the true positive rate (for default) against the false positive rate, for all thresholds. A threshold is the chosen cut-off in the estimated scores from the neural net.

Since the ROC curve is based on the true positive rates and the false positive rates, it is invariant to class proportions. Moreover, it is not dependent on the quality of the predictions (probability estimates), only on the classifier's ranking capabilities.

ROC curves are often summarized through the area under the curve, or AUC. With perfect classification, the AUC will be 1, while random predictions result in an AUC of 0.5 on average.

AUC is a general measure of binary classification, and is widely used across disciplines. There are however more specialized measures for evaluating a credit scoring model. One such measure is the Expected Maximum Profit (EMP) (Verbraken, Bravo, Weber, & Baesens, 2014), which is a profit based performance measure accounting for the benefits generated by healthy loans and the costs caused by loan defaults. We have chosen not to use EMP in this paper. The main reason is that this measure is heavily dependent on the class proportions. As stated earlier, we have used a fictive default rate, since the bank did not want to disclose its true ratio. Hence, it does not make any sense to use a measure that significantly changes when the class proportions are altered.

Our convolutional neural network is initialized with random weights. Moreover, we use dropout by multiplying random Bernoulli variables with the input to the last layer, and the net is trained with stochastic gradient descent. As a result, the network will be slightly different every time it is trained. Hence, to control how our results are influenced by randomness, we repeat all experiments 10 times and report the average and standard deviation of the 10 resulting AUC values.

4.2. Competing architectures

In the following section, the two CNNs introduced in Section 3.1.4 are compared and evaluated. The architectures of the two models are identical, except that the first takes individual time series as input (*Indiv model*), while all six time series are input to the latter (*Full model*). Note that all training sets are augmented as described in Section 3.2.2. Summary statistics for the AUCs are shown in Table 3 under the names *Indiv, 2 conv layes* and *Full, 2 conv layers*. We see that the individual model is clearly better than the full model. This might indicate that the full model is not able to take advantage of interactions between the series, or at least that the added benefit is smaller than the downside of the added complexity of the model.

Table 4
AUC for individual series.

Series	Mean AUC	Std	NM Mean AUC	NM Std	Prop Missing
ch	0.8632	0.0040	0.8646	0.0036	0.0068
in	0.7116	0.0108	0.7140	0.0107	0.0068
cc	0.7636	0.0080	0.8099	0.0101	0.2405
sa	0.7902	0.0056	0.8178	0.0039	0.1463
sum	0.8752	0.0078	0.8752	0.0078	0
trch	0.8472	0.0064	0.8480	0.0063	0.0068
All	0.9146	0.0025	0.9146	0.0025	0

In the NM columns (Not Missing) we have removed customers having missing series or series without any activity. "Prop missing" gives the proportion of series removed from the test set in NM. The AUC is averaged over 10 experiments. The "All" series gives the score of the averaged predictions (Indiv model).

Table 5
AUC for different subsets of the training data.

Proportion	Mean AUC	Std AUC
1.00	0.9146	0.0026
0.90	0.9109	0.0048
0.75	0.9086	0.0054
0.50	0.9028	0.0033
0.25	0.8954	0.0082
0.10	0.8827	0.0062
Non-augmented	0.8880	0.0042

For each subset, we sample the customers from the training set, and use sliding window for data augmentation. The leftmost column shows the proportion of customers sampled from the training set, while the remaining columns show the average and standard deviation of AUC for the 10 repetitions. 1.00 gives the results for the full training set (Indiv model), while "Non-augmented" refers to using all customers, but with no data augmentation.

Table 6
AUC for different lengths of time series.

Model	Mean AUC	Std AUC	AUC for average pred.
365 days	0.9146	0.0026	0.9184
180 days	0.8945	0.0051	0.8981
91 days	0.8744	0.0051	0.8791
31 days	0.8425	0.0043	0.8460

The "Mean AUC" is averaged over 10 repetitions. The "AUC for average pred." gives the AUCs for the averaged predictions of the 10 repetitions of the same model. All models have the same architecture as the 365 days CNN.

Table 3 also includes four other models. The only difference between the models is the number of convolutional layers (and pooling layers), and whether the input consist of individual series or all six series simultaneously. For the full specification of the models, see Fig. A.1 in the Appendix. From the table we see that all the individual models have higher average AUC than the full models. Additionally, it seems that there is no reason to have more than two convolutional layers in the networks.

Averaging the predictions from six individually trained models has a regularizing effect. To check whether the differences in performance between the full models and the individual models are caused by this averaging, we also, for each model, compute the AUC for the average of the 10 predictions for each customer. The results are shown in the rightmost column of Table 3 ("AUC for average pred.") We see that this improves the AUC for all models. The individual models are still better than the full models, but the difference seems to be smaller.

In the remainder of this section we concentrate on the *Indiv model, 2 conv layers*, since this is the one that shows the best performance in Table 3.

Table 7
AUC for different length of series.

Model	Mean AUC	Std AUC	AUC for average pred.
RF 365 days	0.9130	–	–
RF 31 days	0.8907	–	–
Combined 365 days	0.9254	0.0011	0.9260
Combined 31 days	0.8941	0.0006	0.8946

RF is a random forests classifier with covariates extracted from the unscaled time series. Combined gives the averaged prediction of the CNN and the random forests classifier. The results for the combined classifier are averaged over 10 repetitions of the CNN. The "AUC for average pred." gives the AUC for the averaged predictions of the 10 repetitions of the CNN.

4.3. Importance of different series

Our model creates predictions based on six different time series: checking account balance (ch), amount transferred into checking account (in), credit card balance (cc), savings account balance (sa), the sum of checking, saving and credit card balances (sum), and number of transactions on the checking account (trch). To investigate how much information there is in each series, we report their individual AUC values in Table 4. As before, we report the results averaged over 10 experiments. The checking account (ch and trch) and the sum seem to be most informative, while the amount transferred into the checking account (in) provides less information.

In Section 2 it was stated that missing savings accounts and credit card accounts are regarded as accounts without any movements. This means that the comparison between the different series in the two leftmost columns of Table 4 might not be completely fair. Hence, we did a new experiment in which we removed all customers missing the relevant time series from the test set. The results are shown in columns 3 and 4 of the table (NM Mean and NM Std), while the proportions of removed customers is shown in column 5. From the table we see that the AUC for credit card accounts (cc) and savings accounts (sa) significantly increase, while there are small differences for the other series. This is reasonable, as the proportion of missing data is largest for the credit cards and savings accounts.

4.4. Training data

In this section we investigate how the size of the training set affects the performance of our classifier. This can help us understand to what extent more training data would improve our discriminator. We also evaluate the effect of our augmentation scheme.

Table 5 shows AUC for models trained on different subsets of the training data. The subsets were created by randomly sampling customers. For a given proportion of the full training set we train 10 models in the same way as previously for the full training set.

Investigating the table, the performance seems to be improved with the size of the training data. This suggests that with more data we might be able to obtain even better results than those presented in this paper.

The last row of the leftmost column in Table 5 shows the results of fitting the same model to the original (not augmented) training set. We see that using 10% to 25% of the original customers with augmented data results in the same performance as using all customers with non-augmented data.

4.5. Length of time series

To investigate to what extent lack of information affects the performance of our model, we have also fitted our CNN to series of different lengths, More specifically, we used the last month (31

Table 8
Evaluation metrics.

Model	Threshold	Accuracy	Sensitivity	Specificity	AUC	Brier Score	H-measure
CNN	0.3	0.954	0.374	0.985	0.915	0.0381	0.564
Combined	0.3	0.947	0.527	0.970	0.925	0.0364	0.592
LR	0.3	0.910	0.490	0.935	0.864	0.0458	0.455
MLP (512)	0.3	0.899	0.590	0.918	0.875	0.0550	0.484
RF	0.3	0.920	0.602	0.940	0.913	0.0386	0.557
CNN	0.5	0.953	0.064	0.999	0.915	0.0381	0.564
Combined	0.5	0.956	0.177	0.997	0.925	0.0364	0.592
LR	0.5	0.943	0.219	0.981	0.864	0.0458	0.455
MLP (512)	0.5	0.927	0.455	0.953	0.875	0.0550	0.484
RF	0.5	0.957	0.332	0.991	0.913	0.0386	0.557

Different evaluation metrics for two different threshold values. All results are based on the time series with length 365 days. For each model we have averaged the metrics over 10 trained models. CNN is the best-performing CNN from Table 3, LR is a logistic regression model, MLP is a Multilayer perceptron with one hidden layer and 512 nodes, RF is the random forest model, and Combined is the combined model in which we averaged the predictions from the CNN and the RF.

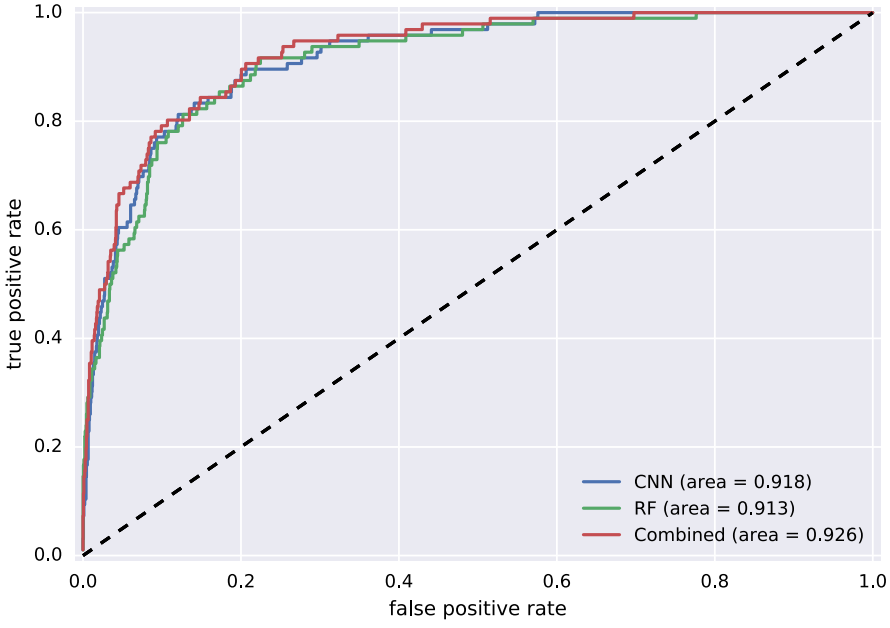

Fig. 4. ROC curves for the CNN, the RF, and their averaged predictions model. The time series are the full 365 day long series. The CNN comes from the averaged predictions over the 10 experiments.

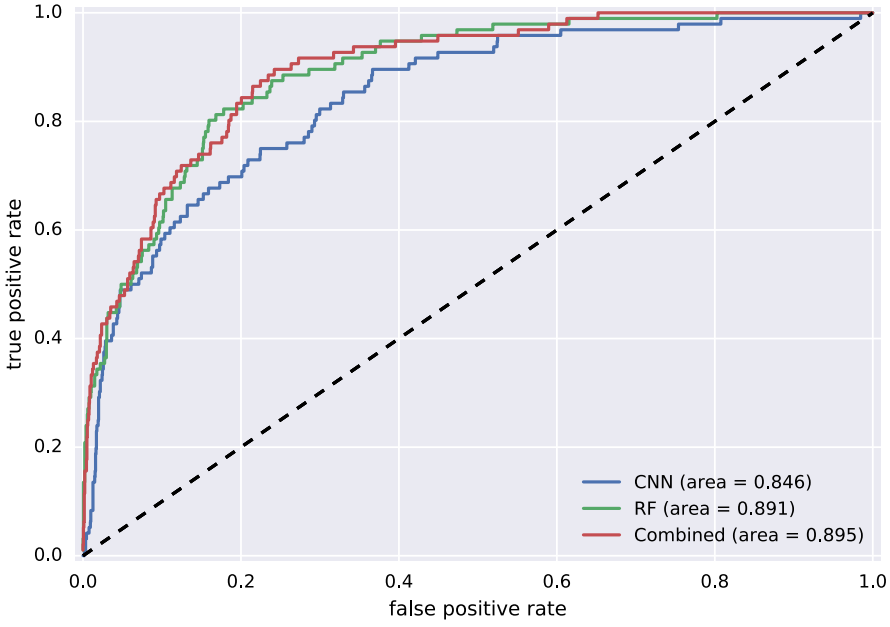

Fig. 5. ROC curves for the CNN, the RF, and their averaged predictions model. The time series used are contain only the last 31 days of data. The CNN comes from the averaged predictions over the 10 experiments.

days), the last quarter (91 days), and the two last quarters (182 days) of the original dataset to train the same model as previously.

The results are shown in Table 6. As can be seen, there is a clear drop in performance when the time series get shorter. On the downside, this makes it harder to evaluate customers with a short relationship to the bank. However, the results also suggest that it might be possible to increase the performance of our models by applying them to time series longer than 365 days.

4.6. Classical covariates

As described in Section 2.2, all time series are scaled to be in [0, 1] before fitting the CNN. This kind of scaling means that information about the magnitude of the series is lost. To assess the potential decrease in results connected to this removal of relevant information, we fit a random forests classifer (RF), consisting of 800 trees, to explanatory variables created from the original, unscaled time series. RF is often credited as a very strong classifier outperforming several alternative methods (including Support Vector Machines and Neural networks) when applied to credit scoring (Brown & Mues, 2012; Kruppa, Schwarz, Arminger, & Ziegler, 2013; Lessmann et al., 2015). For each of the six time series, we computed the mean, max, minimum, standard deviation, and the standard deviation scaled by the mean. These covariates were computed both for the full series and for the last month (31 days). In addition, we divided each of the covariates for the full series by the equivalent covariate calculated for the last month, producing a total of 15 features for each series. The resulting AUC is shown in the upper row of Table 7. We also fitted a logistic regression model and a Multilayer perceptron (with one hidden layer and 512 nodes) using the same explanatory variables. The AUC values for these models were significantly worse (0.86 and 0.88 see Table 8) than that for the RF, confirming the observations from previous studies. Hence, we decided not to include these models in the further comparisons.

Comparing the numbers in the upper rows of Tables 6 and 7 we see that for the 365 days period, the AUC for the CNN is slightly better than that for the RF. We also used a combined model in which we averaged the predictions from the CNN and the RF. From the third row of Table 6 we see that there is a small increase in performance compared to that obtained using one of the models only. The ROC curves for the CNN, the RF and the combined model are shown in Fig. 4. Running the diagnostic test for comparing ROC curves described in DeLong, DeLong, and Clarke-Pearson (1988) verifies that there are no significant differences between the AUCs for any pair of ROC-curves.

Table 8 gives the values of 6 evaluation metrics for each of the 5 models. In addition to Accuray, Sensitivity, Specificity and AUC, the Brier Score (Brier, 1950) and the H-measure (Hand, 2009; 2010) are reported. The AUC is sometimes criticised for treating the relative severities of misclassifications differently when different classifiers are used. The H-measure, on the other hand, may accommodate expert knowledge regarding misclassification costs, whenever that is available. We want to treat misclassifications of the smaller class as more serious than those of the larger class, since otherwise very little loss would be made by assigning everything to the larger class. Hence, we set the cost of misclassifying a bad customer as healthy to 0.95 and the cost of misclassifying a good customer as defaulter to 0.05.

We have also compared the three methods using data only from the last 31 days. The AUC-values are given in the last row of Table 6 and rows two and four in Table 7, respectively, while the corresponding ROC-curves are shown in Fig. 5. As can be seen from the tables and the figure, the performance of the RF does not degrade as much as that of the CNN when decreasing the length of the time series to 31 days. The p-value of the test for comparing the two ROC-curves is 0.005, meaning that the AUC for the RF is significantly higher than that for the CNN in this case.

5. Discussion

The ability to discriminate bad customers from good ones is very important for banks and other lending companies. There is a large literature on methods used to predict defaults of consumer loans. Status quo in both the industry and the academic literature is to use models with handcrafted explanatory variables. In contrast, we use a highly nonlinear convolutional neural network (CNN), where the input is raw account transactional data. More specifically, we have used daily balances of customers' checking accounts, savings accounts, and credit card accounts, in addition to daily number of transaction on the checking accounts, and amount transferred into the checking accounts to predict mortgage delinquency.

A CNN has a number of hyperparameters which need to be chosen, such as the number of layers, type of layers, and the type of regularization. Our best performing CNN with two convolutional and two fully connected layers achieved an AUC of 0.915. Considering the fact that our training set only consists of 12,696 customers, and that we do not use any other information than the account data, the results are quite promising.

In our analysis, we found that the AUC for our CNN was increasing with the size of the training set, suggesting that a larger dataset might result in even higher performance. In addition, the performance increased with the length of the time series, indicating that using a time series that is longer than 365 days may give better results. Comparing the predictions from a single CNN with ensemble predictions from 10 randomly initialized models showed that the increase in AUC using several model was limited.

By using transaction data only, one throws away a lot of other informative data that is typically used for credit scoring, e.g. the loan balance, socioeconomic data, payment history, and credit bureau data. Further, all time series are scaled to be in the interval [0,1] before fitting the CNN, meaning that information about the magnitude of the time series is lost. To account for the scaling, a random forests classifier was fitted to covariates extracted from the unscaled series, resulting in an AUC of 0.913. By combining the predictions of the CNN and the random forests, we achieved an AUC of 0.925.

Future research directions could include the combination of our model with other credit scoring models that use more classical credit information; applying our model on a larger dataset, as CNNs tend to perform better with more data; and using more granular data in the form of labeled transactions (e.g. groceries, gas, rent, vacation).

Acknowledgments

This work was supported by The Norwegian Research Council 237718 through the Big Insight Center for research-driven innovation. We thank Ørnulf Borgan for comments and suggestions on the paper. We also thank Sven Haadem for his help on creating a dataset from DNB's mortgage portfolio.

Appendix A

Fig. A.1.

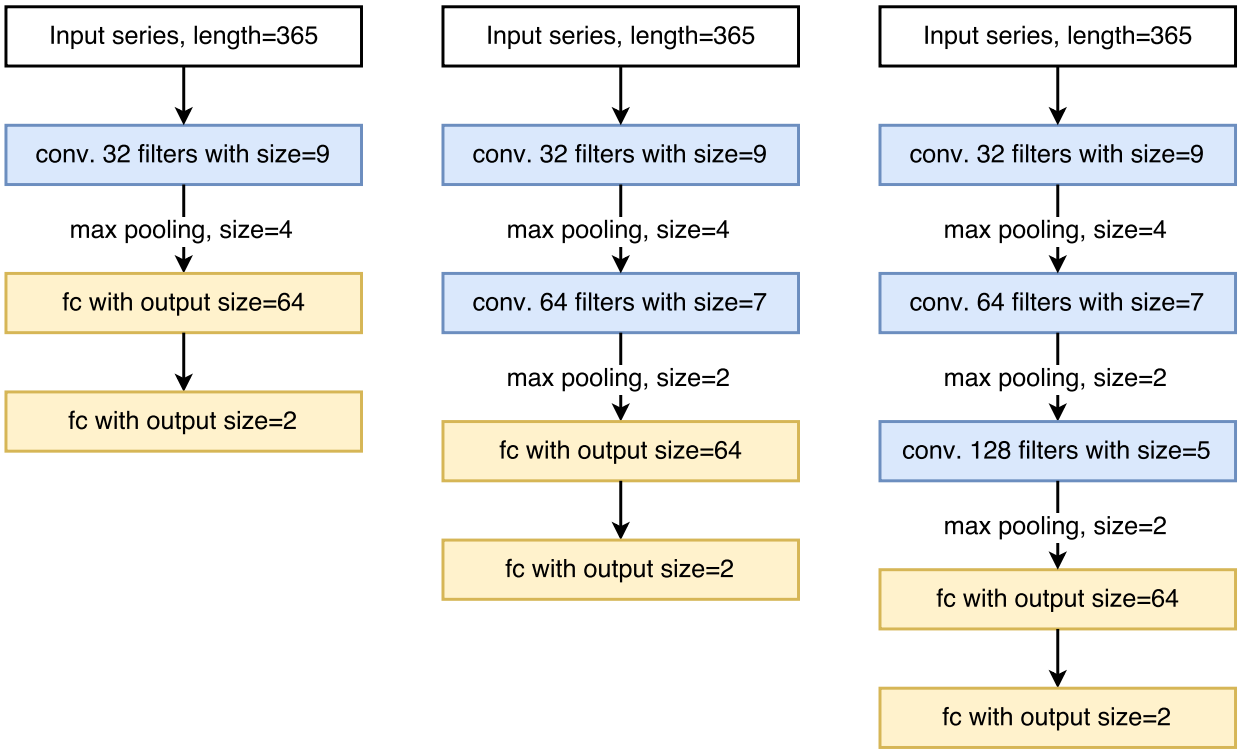

Fig. A.1. Our three convolutional neural net architectures for single time series. We also have a version of each model with 365×6 input. The blue layers are convolutional layers and the yellow layers are fully connected layers. All activations are ReLU, with the exception of softmax in the final layer. Dropout, with rate 0.5, is performed between the last two layers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

References

Abellán, J., & Castellano, J. G. (2017). A comparative study on base classifiers in ensemble methods for credit scoring. *Expert Systems with Applications, 73*, 1–10. doi:10.1016/j.eswa.2016.12.020.

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2016). The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*. doi:10.1007/s10618-016-0483-9.

Barboza, F., Kimura, H., & Altman, E. (2017). Machine learning models and bankruptcy prediction. *Expert Systems with Applications, 83*, 405–417. doi:10.1016/j.eswa.2017.04.006.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review, 78*(1), 1–3.

Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications, 39*(3), 3446–3453.

Butaru, F., Chen, Q., Clark, B., Das, S., Lo, A. W., & Siddique, A. (2016). Risk and risk management in the credit card industry. *Journal of Banking & Finance, 72*, 218–239. doi:10.1016/j.jbankfin.2016.07.015.

Chen, X., Zhou, C., Wang, X., & Li, Y. (2017). The credit scoring model based on logistic-bp-adaboost algorithm and its application in p2p credit platform. In X. Li, & X. Xu (Eds.), *Proceedings of the fourth international forum on decision sciences* (pp. 119–130). Singapore: Springer Singapore.

Chi, B.-W., & Hsu, C.-C. (2012). A hybrid approach to integrate genetic algorithm into dual scoring model in enhancing the performance of credit scoring model. *Expert Systems with Applications, 39*(3), 2650–2661. doi:10.1016/j.eswa.2011.08.120.

Cui, Z., Chen, W., & Chen, Y. (2016). Multi-Scale Convolutional Neural Networks for Time Series Classification. *CoRR* http://arxiv.org/abs/1603.06995.

DeLong, E. R., DeLong, D. M., & Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics, 44*(3), 837–845.

Finanstilsynet (2016). Finansielt utsyn 2016. http://www.finanstilsynet.no/Global/Venstremeny/Rapport/2016/Finansielt_utsyn_2016.pdf. Accessed: 2017-02-06.

García, V., Marqués, A. I., & Sánchez, J. S. (2014). An insight into the experimental design for credit risk and corporate bankruptcy prediction systems. *Journal of Intelligent Information Systems, 44*(1), 159–189. doi:10.1007/s10844-014-0333-4.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. http://www.deeplearningbook.org.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., & Wang, G. (2017). Recent Advances in Convolutional Neural Networks. *CoRR* http://arxiv.org/abs/1512.07108.

Hammerla, N. Y., Halloran, S., & Ploetz, T. (2016). Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (pp. 1533–1540). New York, New York, USA: AAAI Press. http://dl.acm.org/citation.cfm?id=3060832.3060835.

Hand, D. J. (2009). Measuring classifier performance: A coherent alternative to the area under the ROC curve. *Machine Learning, 77*(1), 103–123. doi:10.1007/s10994-009-5119-5.

Hand, D. J. (2010). Evaluating diagnostic tests: The area under the ROC curve and the balance of errors. *Statistics in Medicine, 29*(14), 1502–1510. doi:10.1002/sim.3859.

Jones, S., Johnstone, D., & Wilson, R. (2015). An empirical evaluation of the performance of binary classifiers in the prediction of credit ratings changes. *Journal of Banking & Finance, 56*, 72–85. doi:10.1016/j.jbankfin.2015.02.006.

Kennedy, K., Mac Namee, B., Delany, S. J., O'Sullivan, M., & Watson, N. (2013). A window of opportunity: Assessing behavioural scoring. *Expert Systems with Applications, 40*(4), 1372–1380.

Khandani, A. E., Kim, A. J., & Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance, 34*(11), 2767–2787. doi:10.1016/j.jbankfin.2010.06.001.

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *CoRR* http://arxiv.org/abs/1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 1097–1105). Curran Associates, Inc. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Kruppa, J., Schwarz, A., Arminger, G., & Ziegler, A. (2013). Consumer credit risk: Individual probability estimates using machine learning. *Expert Systems with Applications, 40*(13), 5125–5131.

Le Guennec, A., Malinowski, S., & Tavenard, R. (2016). Data augmentation for time series classification using convolutional neural networks. In *Proceedings of the ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data ,Riva Del Garda, Italy*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. doi:10.1038/nature14539.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (1998). Efficient backprop. *Neural Networks: Tricks of the Trade*, 9–50. doi:10.1007/3-540-49430-8_2.

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136. doi:10.1016/j.ejor.2015.05.030.

Lundberg, S., & Lee, S. (2016). An unexpected unity among methods for interpreting model predictions CoRR abs/1611.07478.

Norges-Bank (2012). Årsrapport om betalingssystem 2012. http://static.norges-bank.no/pages/94894/Betalingssystem_2012_o.pdf?v=27052013101201&ft=.pdf. Accessed: 2017-02-09.

Ordóñez, F., & Roggen, D. (2016). Deep convolutional and LStm recurrent neural networks for multimodal wearable activity recognition. *Sensors, 16*(1), 115. doi:10.3390/s16010115.

Prasad, S. C., & Prasad, P. (2014). Deep Recurrent Neural Networks for Time Series Prediction. *CoRR* http://arxiv.org/abs/1407.5949.

Ravi Kumar, P., & Ravi, V. (2007). Bankruptcy prediction in banks and firms via statistical and intelligent techniques – A review. *European Journal of Operational Research, 180*(1), 1–28. doi:10.1016/j.ejor.2006.08.043.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier CoRR abs/1602.04938.

Samek, W., Wiegand, T., & Müller, K. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models CoRR abs/1708.08296.

Shrikumar, A., Greenside, P., Shcherbina, A., & Kundaje, A. (2016). Not just a black box: Learning important features through propagating activation differences CoRR abs/1605.01713.

Sirignano, J., Sadhwani, A., & Giesecke, K. (2016). Deep Learning for Mortgage Risk. *ArXiv e-prints* http://adsabs.harvard.edu/abs/2016arXiv160702470S.

Sousa, M. R., Gama, J., & Brandão, E. (2016). A new dynamic modeling framework for credit risk assessment. *Expert Systems with Applications, 45*, 341–351. doi:10.1016/j.eswa.2015.09.055.

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

Thomas, L. C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International journal of forecasting, 16*(2), 149–172.

Verbraken, T., Bravo, C., Weber, R., & Baesens, B. (2014). Development and application of consumer credit scoring models using profit-based classification measures. *European Journal of Operational Research, 238*(2), 505–513. doi:10.1016/j.ejor.2014.04.001.

Xia, Y., Liu, C., Li, Y., & Liu, N. (2017). A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. *Expert Systems with Applications, 78*, 225–241. doi:10.1016/j.eswa.2017.02.017.

Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L., & Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the twenty-fourth international conference on artificial intelligence*. In *IJCAI'15* (pp. 3995–4001). AAAI Press.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. L. (2014). Time series classification using multi-channels deep convolutional neural networks. *Lecture Notes in Computer Science*, 298–310. doi:10.1007/978-3-319-08010-9_33.