Central European Economic Journal

sciendo

**ISSN:** 2543-6821 (online)

Journal homepage: http://ceej.wne.uw.edu.pl

Aneta Dzik-Walczak,
Maciej Odziemczyk

# Modelling cross-sectional tabular data using convolutional neural networks: Prediction of corporate bankruptcy in Poland

**To cite this article**

Dzik-Walczak A., Odziemczyk M. (2021). Modelling cross-sectional tabular data using convolutional neural networks: Prediction of corporate bankruptcy in Poland. Central European Economic Journal, 8(55), 352-377.

**DOI:** 10.2478/ceej-2021-0024

To link to this article: https://doi.org/10.2478/ceej-2021-0024

Central European Economic Journal

**Aneta Dzik-Walczak**
Faculty of Economic Sciences, University of Warsaw, Poland
corresponding author: adzik@wne.uw.edu.pl

**Maciej Odziemczyk**
Faculty of Economic Sciences, University of Warsaw, Poland

# Modelling cross-sectional tabular data using convolutional neural networks: Prediction of corporate bankruptcy in Poland

**Abstract**
The paper deals with the topic of modelling the probability of bankruptcy of Polish enterprises using convolutional neural networks. Convolutional networks take images as input, so it was thus necessary to apply the method of converting the observation vector to a matrix. Benchmarks for convolutional networks were logit models, random forests, XGBoost, and dense neural networks. Hyperparameters and model architecture were selected based on a random search and analysis of learning curves and experiments in folded, stratified cross-validation. In addition, the sensitivity of the results to data preprocessing was investigated. It was found that convolutional neural networks can be used to analyze cross-sectional tabular data, especially for the problem of modelling the probability of corporate bankruptcy. In order to achieve good results with models based on parameters updated by a gradient (neural networks and logit), it is necessary to use appropriate preprocessing techniques. Models based on decision trees have been shown to be insensitive to the data transformations used.

**Keywords**
convolutional neural networks | machine learning | simulation | bankruptcy prediction | financial indicators

**JEL Codes**
C45, C02, G33

## 1. Introduction

The paper considers the problem of modelling the probability of bankruptcy on the example of Polish companies from the manufacturing sector in the years 2007–2013. The problem described, due to its specificity, ie, the lack of causal effect between the vector of characteristics and the label, is usually considered a predictive task. Therefore, the aim is not to infer the process of generating the data, because such a process does not exist in the form considered. The predictive character of the task makes it a constantly topical issue, research is focused on finding better methods of analysis. Such is also the purpose of this article: the presentation and application of a non-standard method of analysis and a proposal for solving the problems that are an integral part of it.

The main research objective was to test whether convolutional neural networks can achieve results comparable to models commonly used for bankruptcy forecasting based on cross-sectional tabular data. It was expected that data preprocessing involving the transformation of predictor distributions would be a key aspect of the analysis. We supposed that the transformations would prove useful in models based on gradient-updated parameters, as opposed to models based on decision trees.

The paper is structured as follows. The first chapter presents the literature review. The second chapter describes the methodology. Chapter three presents the results. Experiments performed in cross-validation concerning both the conversion of observation vectors to images as well as the models themselves, selection of hyperparameters, and architecture are also described.

The analytical part of the study was performed using the Python language.

## 2. Review of the literature

Modelling the probability of corporate bankruptcy is a topic that researchers have been dealing with continuously for years, applying more and more effective methods. As a rule, these analyses are based on financial ratios, which is a highly flexible approach, and we can easily imagine its application in business. One of the first models based on financial ratios is the Z-score discriminant function (Altman, 1968). The logit model for bankruptcy prediction was first used by Ohlson (1980). The bias on the estimators of the Maximum Likelihood Method in the case of artificial balancing of the sample and its non-random selection was first described in the literature by Zmijewski (1984).

### 2.1. Current methods and challenges

In order for standard statistical and econometric models to produce the expected results, it is often required to meet a number of assumptions, such as the normality of the distribution, linear dependence, the constancy of variance in time, or independence of predictors in relation to each other, which, as it turns out, is difficult to achieve in reality. Moreover, in order to draw correct statistical inferences, it is extremely important to know the functional form of the process generating the data *a priori*, which is often a problem that cannot be solved. The existing limitations have led researchers to develop research methodologies resulting in a number of machine learning algorithms. As it turns out, modern methods often show better predictive capabilities than typical statistical models without requiring compliance with restrictive assumptions. However, this is very often at the expense of interpretability.

Zhang, Hu, Patuwo & Indro (1999) were among the first to propose a feedforward multilayer perceptron (MLP) neural network model to model the probability of corporate bankruptcy. In terms of metric accuracy, the neural network was found to be superior to the logit model. Kim & Kang (2010) proposed two novel approaches to modelling with MLP: bagging and boosting, methods belonging to the ensembling class. Both of these methods are based on training multiple small and weak models (weak classifiers) and combining them to produce more accurate and robust predictions. AdaBoost, or decision trees with only one stump trained in the boost system, also find application in bankruptcy prediction (Heo & Yang, 2014). The results reported allow us to conclude that the discriminatory model inspired by Altman's Z-score model has better predictive quality for small companies in terms of both Type I and Type II errors. The AdaBoost model was better in terms of Type I errors for medium and large companies. Issues such as imbalance of the training set (Veganzones & Séverin, 2018; Carmon, Climent & Momparler, 2019) and the role of preprocessing (Wyrobek & Kluza, 2018; Vellamcheti & Singh, 2020) were also emphasised in previous studies. Son, Hyun, Phan & Hwang (2019) reported significant gains in success metrics after applying the Box-Cox power transformation (Box & Cox, 1964). Pawełek (2019) examined the performance of boosted decision trees in the context of outlier observations. The solution to the outlier problem proposed by Pawełek was to constrain the set with appropriate quantiles from above and below but only for observations that were assigned the label of non-bankruptcy. The approach seems controversial because in this way the model can build a decision rule that assigns the bankrupt label based on the determination of whether an observation is an outlier. Given that the outlier reduction process took place only on the training set, it can be concluded that the results from the test set are still unbiased.

### 2.2. Articles of inspiration

Zięba, Tomczak & Tomczak (2016) addressed the automation of feature generation in a boosted decision tree algorithm (using XGBoost). The authors, Carmon et al. (2019), took advantage of the ability to assign importance to individual features using XGBoost. From the features that exceeded the desired level of importance, a discrete probability distribution was created, and this importance was the probability of a particular event occurring. Two features were then drawn at random from the distribution created and one from the basic arithmetic operations to be performed on them. Obviously, the more trees built within the algorithm, the more synthetic and less interpretable the attributes became; however, the goal was prediction, not inference.

The method presented by Hosaka (2019) consisted in converting a vector of parameters to a greyscale image using Monte Carlo simulation, which allowed the arrangement of pixels, ie, representations of each of the financial ratios disposed, in a two-dimensional space taking into account their correlation with each other, and then the training of a convolutional neural network. The model proposed was compared with the CART decision tree, discriminant function, SVM, MLP, and AdaBoost. The highest ROC-AUC scores were obtained for the convolutional neural network.

Szegedy et al. (2015) proposed a new order for building convolutional neural networks – the inception module. Normally, convolutional models are built in a fairly uniform way: several convolutional layers are followed by a pooling operation applied resulting in a three-dimensional tensor, which at the end is flattened to a one-dimensional vector to be connected to a dense layer. The standard approach to achieving better results when using neural networks is to build a wider and deeper architecture, which is directly related to a much higher computational cost; moreover, it often results in over-fitting. The proposed inception blocks allow a significant increase in the architecture while reducing the computational cost.

When we review the literature, it is possible to notice the recurrence of certain problems: a strong imbalance, high correlation coefficient of predictors and skewness of their distributions, the occurrence of outliers, the curse of multidimensionality, the lack of a defined framework of a good model and data generating process. Researchers try to propose solutions to the problems mentioned above, usually bringing better results.

## 3. Methodology

The ignorance of the process generating the data – and thus the inability to determine the causal influence – force analysts to try to find the best possible approximation of reality. Such an approximation, ie a de facto model according to econometric procedures, usually consists in determining the nearest truth specification of a phenomenon according to beliefs (or theories). These beliefs not only include explanatory variables that have a potential influence on the variable explained, an additional aspect is the functional form of the phenomenon itself. Then, with the use of statistical tools, the accuracy of beliefs is checked by, among others, testing the significance of the explanatory variables, correctness of functional form or assumptions required for non-bias, consistency, or the effectiveness of a given model. It can be argued that model building involves approximation of a real data generating process. However, the available tools require a great deal of expertise and have a limited spectrum of functional dependencies to be tested. In response to the problems presented the solutions discussed below have been developed.

### 3.1. Convolutional Neural Networks

A single input to the MLP model is a vector whose components undergo transformations within each neuron of the next layer. The values stored in the hidden layer neurons are thus the features of the given vector that the model learns during the training process. If, on the other hand, the description of a given observation has more than one axis, then it is a matrix, and it cannot be directly reprocessed by the MLP model. One solution is to flatten the matrix into a vector. This solution has two major drawbacks, the first being the high computational cost, since each component must be concatenated with each neuron of the neighboring layer. The second disadvantage is the omission of the issue of the neighbourhood of components in the model structure, which occurs during the flattening operation. The drawbacks mentioned become even more significant if the representation of the observation is an image.

By convolving the image representation, ie tensor $A \in [0,255] \in Z^{n_h \times n_w \times n_c}$, where $n_h \times n_w \times n_c$ represents the height, width, and number of channels of the image through the filter, respectively; $F \in R^{f_h \times f_w \times n_c}$ is understood to be the conduct of operations:

$$conv(A,F)_{x,y} = A * F_{x,y} = \sum_{i=1}^{f_h} \sum_{j=1}^{f_w} \sum_{k=1}^{n_c} f_{i,j,k}\, a_{x+i-1,y+j-1,k}\,,$$

where $x$, $y$ are the indices of the result of the operation, ie the matrix $A*F$, $n_c$ is the number of channels that varies depending on the color palette of the image, for RGB $n_c$=3, for greyscale image $n_c$=1. The operation is possible only if the third dimension of tensor $A$ is equal to the third dimension of filter $F$ and if the first two dimensions of tensor $A$ are not less than the first two dimensions of filter $F$. The dimensions of the result of this operation are $(n_h - f_h + 1, n_w - f_w + 1)$. In the convolutional layer, the parameters are the elements of the filter $F$, and they are optimised analogously to the dense network.

By default, the more convolutional layers the network contains the smaller the height and width of the returned volume and the larger the depth. Reducing the first two dimensions is not always a desirable effect; then the input tensor is complemented with some constant, usually zeros, and this operation is called padding. The order of padding is determined by the *p* parameter; for *p*=1, a row is added from the top and bottom, and a column from the top and bottom, to make a 'frame'. When *p*=1 is applied, the size of the tensor $A \in R^{n_h \times n_w \times n_c}$ is changed to $(n_h+2, n_w+2, n_c)$.

The last of the convolution layer hyperparameters considered in this paper is the so-called stride (*s*) or filter step. The stride significantly affects the size of the output tensor, the final formula for the size of the output feature map (total) is:

$$\left(\left\lfloor \frac{n_h + 2p - f_h}{s} + 1 \right\rfloor, \left\lfloor \frac{n_w + 2p - f_w}{s} + 1 \right\rfloor, \#F\right),$$

where $\lfloor . \rfloor$ means a rounding down operation.

The convolutional layer is thus used for direct extraction of low-level features. A family of operations is defined to aggregate these features (ie pooling). Standard types of pooling are max pooling and average pooling. Both versions rely on a mechanism similar to convolution, ie, moving the filter along the input tensor, with the difference that the filter does not have weights updated in the learning process.

### 3.1.1. Inception module

A convolutional neural network has been used with success in tasks in the area of image analysis (Simonyan & Zisserman, 2014; Krizhevsky, Sutskever & Hinton, 2012; Lin, Chen & Yan, 2013; LeCun et al., 1998). These models typically use 3 × 3, 5 × 5 or 7 × 7 filters. An interesting approach is to use an inception module. The inception module uses several variants of layers simultaneously; originally, in the paper by Szegedy et al. these were 1 × 1, 3 × 3, 5 × 5 and max pooling. The individual feature maps are concatenated along a specific axis, so as mentioned in the literature review, a correspondence between two of the three dimensions of the concatenated tensors is required. In order to achieve this correspondence, the layers themselves (convolution and pooling) are used. However, such a module scheme is computationally very expensive; the remedy turns out to be the addition of 1 × 1 convolutions before 1 × 1 and 5 × 5, and after pooling, it is then possible to reduce the *z*-axis of the output tensors, as reported by the authors without significantly reducing the information held.

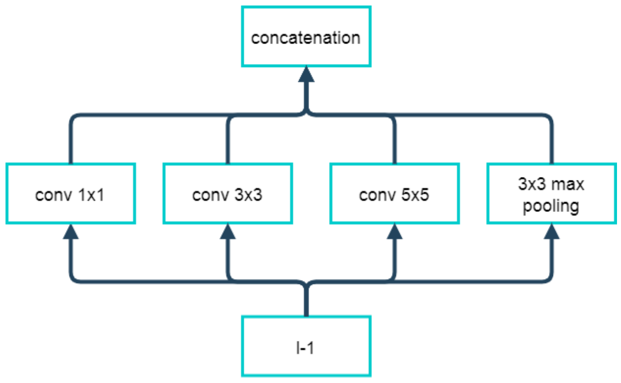


**Figure 1.** Naive Inception
Source: Own elaboration

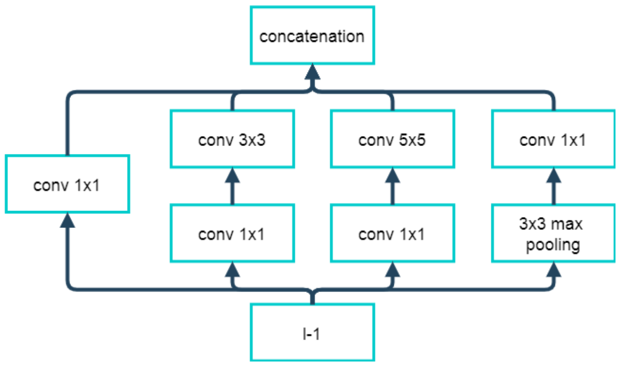


**Figure 2.** Optimum Inception
Source: Own elaboration.

This study uses an optimised variant of the inception module (Figure 2).

## 3.2. Validation methods

A key aspect of predictive modelling is the selection of an appropriate model of validation method. The basis of the division into training and test sets is the consistency of the distributions of variables. Particularly important in unbalanced classification problems is to keep the label distributions as close as possible during the division, in order to maintain the desired properties instead of using random division a layered division is used and this was done in this study. A common division of data is 70% for training set, 20% for validation set and 10% for test set. The problem arises when few data are in the dataset, in which case the validation and test sets may not be an adequate representation of the population, especially in highly unbalanced problems where only a few positive labels – or in extreme cases none – may end up in the test set. Furthermore, testing many combinations of hyperparameters involves a risk of overfitting to the validation set, in which case the results obtained may be too optimistic. The solution to both problems is K-fold cross-validation, or in the case of this study, K-fold stratified cross-validation.

K-fold cross-validation, involves dividing the training set into K subsets and training the model K times. During one iteration, the *k*th set serves as the validation set while the rest are used to train the algorithm, thus obtaining K results, which are finally averaged to estimate the generalisation ability of the model. The premise of K-fold cross-validation is to use each observation as a validation observation exactly once. During the validation process, the separation rules between training, validation, and test sets must be respected, ie data transformations can take place only on the basis of the training set characteristics and their distributions must be as similar as possible to each other. Layering is applied analogously.

In this study, it was decided to divide the dataset into a training set (90%) and a test set (10%). A 6-fold stratified cross-validation procedure was then performed on the training set, ie, in each iteration the validation set was separated from the training set. In the literature K is often stated to be 5–10. In general, the higher the number of folds, the more robust the results. Choosing K equal to 6 we gain a bit of the robustness of results in relation to five folds not increasing the cost of calculations too much (compared with 10 folds) at the same time. All data transformations were performed inside the validation, except for the removal of attributes with too many missing values (completion was already done in the current validation iteration, separately). The hyperparameters of the random forest and boosted decision tree (XGBoost) models were tuned using random search after examining the sensitivity of a given algorithm to a particular hyperparameter and narrowing the range considered. During random search, hyperparameters are randomised, then validation metrics are measured and stored in the validation process, and finally a configuration is selected as preferred. As the main metric in this study is AUC-PR, it was the value of this metric that was used to select the best model, in addition, attention was also paid to overfitting, ie the difference in performance between the training and validation sets.

For neural network models, the hyperparameter space is too large to rely on random search; in addition, training and validating these models are much more computationally expensive. The architecture and hyperparameters of the neural networks were selected based on experiments and analysis of the results obtained (metrics, learning curves, early stopping) in a 6-fold stratified cross-validation.

## 3.3. Specification of tested models

The sensitivity of each model type (logit, random forest, XGBoost, MLP neural network, and CNN) to preprocessing variant (raw data, normalised data, data without outliers and normalised, data without outliers after power transformation and normalised) and hyperparameters was tested. If justified (appropriate space of considered hyperparameters) a random search was performed to find a quasi-optimal set of them. If it was necessary (neural networks), learning curves were analysed, both in terms of the best results obtained with the use of early stopping, as well as the shape of curves (desirable possibly smooth, monotonic course and as little overtraining as possible).

The logistic regression model was considered in four preprocessing variants: raw data, normalised data, data without outliers and normalised, data without outliers after power transformation and normalised. The only hyperparameter tested was regularisation by the $L_1$ norm ($R_{L_1}(W) = \lambda \sum_{k=1}^{K} |w_k|$).

The random forest model proved to be resistant to a variant of the data preprocessing, so it was decided to treat it as a hyperparameter of the learning process during random search. The sensitivity of the model to changes in individual hyperparameters – maximum tree depth, maximum number of features used in the tree, minimum number of observations required to perform division, minimum number of observations in a leaf, the minimum number of observations required to perform the division – was examined.

The method of data preprocessing was treated as a hyperparameter also for the XGBoost model. The optimisation of the hyperparameters followed principles analogous to the procedure used for random forests. The set of hyperparameters was as follows: maximum tree depth, learning rate $\eta$, a fraction of the rows used to train the tree, a fraction of the columns used to train the tree, minimum loss function reduction, a regularisation of $L_1$, regularisation $L_2$ ($R_{L_2}(W) = \lambda \sum_{k=1}^{K} (w_k)^2$).The number of estimators was calculated as the early stopping median of all validation subsets.

Due to the similarity in the training process between the MLP and CNN models, the procedure for selecting the preprocessing variant, architecture, and hyperparameters was analogous for the latter.

## 4. Results

### 4.1. Data

The dataset used in this study was created for the article 'Ensemble boosted trees with synthetic feature generation in application to bankruptcy prediction' (Zięba et al., 2016). The set was divided into five subsets: bankruptcies at one-year, two-year, three-year, four-year, and five-year horizons were analysed. The set of explanatory variables includes 64 financial ratios describing a company, while the objective variable is the state of that company after the *t*th period, bankrupt or not bankrupt, respectively. The data come from the EMIS database, which contains, among others, financial statements. The objects analysed are Polish enterprises from the manufacturing sector, and the time range is 2000–2013.

### 4.2. Analysis and preparation of data for modelling

The main problems that had to be solved during data analysis and modelling were a strong imbalance, the presence of many data gaps and outliers, and the skewness of the predictor distributions.[1]

Strong unbalancing compounds the problem of insufficient set size. The number of bankrupts in the validation and test sets would be too small to perform a valid procedure. The solution is to use the area under the precision-recall curve as the main metric while monitoring the area under the ROC curve. The problem of validation set counts was solved by using a 6-fold cross-validation, and the disbalance required stratified randomisation.

The procedure for dealing with missing data was designed to preserve as many observations as possible. A gap-filling algorithm was implemented based on random forests.

The problem of outliers was solved by constraining the distributions of the predictors from above and below with quantiles of order $Q_{0.005}$ and $Q_{0.995}$. The skewness problem was also detected. Owing to the fact that the variables analysed in this study are financial ratios, ie, the condition of applicability of the Box-Cox transformation, $x>0$, is not met, it was decided to use its generalisation, the Yeo-Johnson transformation (Yeo & Johnson, 2000).

Limiting distributions, power transformations, and normalisation were the techniques that were used to transform the input data for standard algorithms that process vectors, as convolutional neural networks require data in the form of matrices or tensors. A method proposed by Hosaka (2019) to convert a vector to an image was used. First, the vector had to be converted to a matrix (greyscale), which would eventually become an image. Then, an energy function was defined in the form $E=\Sigma_{(i,j)\in\xi}|\phi[R(i),R(j)]|\times d(i,j)$, where $i$, $j$ are indexes of matrix elements – image pixels, $d(i, j)$ is a measure of distance between pixels, $R(i)$ is a given attribute, $\xi$ is a set of all possible combinations of pairs of $R(i)$ and $R(j)$, and $\phi(\cdot)$ is a correlation coefficient. The result of minimising this function is a matrix (conversion formula) in which the more correlated variables are closer together. The solution to this task was obtained using Monte Carlo simulation.

Square images were assumed to be generated, so if the number of features did not have an equal square root, a complement with a neutral shade of grey (128) was applied. The algorithm was carried out only on companies that were not labeled bankrupt; this approach was also used by Hosaka (2019) arguing that if a company is bankrupt, it is very possible that some anomalies have been observed in it. In the creation of a picture, the goal is to create a pattern that is as structured as possible, which should be easier if thriving units are considered. It is then assumed that if the companies are from the same sector and there are no anomalies, then the relationships between their balance sheet items will be similar.

Owing to the fact that color saturation is coded using integers in the range 0–255, the normalisation method $f:x\in R\rightarrow Z\cap[0,255]$ was required. The function proposed by Hosaka (2019) was implemented:

$$x_{norm} = 0 < int\left(\frac{x-\mu}{\sigma} \times 100 + 128\right) < 255 ,$$

1 Owing to the similarity of the analysis and preprocessing process, it was decided to show details only for the one-year prediction.

where *int*(·) is a function that returns the integer part of a decimal number, 0<⋯<255 means the operation of limiting to boundary values of 0 and 255.

If the observed value was higher than the mean, then the corresponding pixel was brighter, while if the value was lower than the mean, then the pixel was darker. Convolutional neural networks perform better for small floating point numbers; for this reason, before they were passed to the model, the images were scaled to the range [0,1] by dividing each element of the matrix by 255.

The observations are described by 64-dimensional vectors, which means that under the assumption of generating square images, the conversion results were images with a resolution of 8 × 8 pixels. To solve the problem of small images, it was decided to implement a procedure to automatically generate new attributes. It allowed the resolution of the images to be increased from 8 × 8 to 20 × 20 pixels.

## 4.3. Experiments in K-fold stratified cross-validation

It was decided to run the experiments in a 6-fold stratified cross-validation to select the best model in its class (best logit, random forest, XGBoost, MLP, CNN).

### 4.3.1. Conversion of the observation vector to image

This section presents the results of converting the observation vector to an image, one bankrupt and non-bankrupt case each for the quasi-optimised and initialised variants, the 8 × 8 and 20 × 20 sizes for each cross-validation fold in the one-year prediction. In addition, results are presented depending on the data preprocessing.

From Figure A1 in the Appendix, it is hard to see the differences between bankrupt and non-bankrupt. In fact, in most examples it is hard to distinguish between the image before and after the optimisation simulation, which was also reflected in the model results. As it turned out, a technique that reduces the tails of the distributions (see the results in Figure A2 in the Appendix) had a huge effect also during image generation. The reduction of outliers has significantly improved the quality of the generated images, and it is now easy to see the differences between the initialised image and the quasi-optimal image, especially in the case of non-bankrupts (white spot in the upper right corner for folds 1–4). Moreover, it is possible to observe the difference between bankrupt and non-bankrupt: the latter seems to be kept in a lighter tone, which means higher than average values of the ratios.

Figure A3 in the Appendix shows the images generated for data without outliers previously transformed using the Yeo-Johnson transformation. The difference is not as large as when the tails of the distribution are truncated compared with the raw data, but it is noticeable. After the power transformation, the images become sharper, there is less shading around 128, and as the experiments have shown, models trained on such prepared data perform better. It is worth pointing out a common feature of all the 8 × 8 images presented, a grey bar built of five pixels in the lower right corner of the images. This is the previously mentioned effect of complementing the incomplete square root of the number of features with a neutral shade of grey (128).

Figure A4 in Appendix shows images created from only normalised data of size 20 × 20, ie, after generating artificial variables. Similar to the 8 × 8 images, this variant of preprocessing results in low information images with the difference that it is easier to see the optimisation results in the enlarged image.

Similar to the 8 × 8 images, the application of the outlier reduction technique significantly improved the quality of the generated 20 × 20 images. Bankrupts and non-bankrupts can be easily distinguished in the examples presented. Non-bankrupts are usually characterised by a white spot in the upper right corner and lighter shades of the whole image. Bankrupts, on the other hand, do not present a visible pattern even after optimisation, which is consistent with the hypothesis that financial anomalies occur in poorly performing companies.

After applying the power transformation, much sharper 20 × 20 images were obtained, as in the case of 8 × 8, for which there are fewer neutral shades close to saturation 128. The difference between bankrupt and non-bankrupt in this case is even more apparent, similarly with the effect of the optimisation algorithm. The results are shown in Figure A6 in the Appendix.

Thus, it can be concluded that the proposed data transformations significantly affected the quality of the generated images: the more elaborate the procedure the better the results, and the better the images the better the scores of the CNN model metrics, as it turns out.

### 4.3.2. Models

The sensitivity of investigated models to preprocessing variants and hyperparameters was tested.

In the logistic regression model normalisation did not affect the results (the exception was model convergence each time). After we applied the outlier reduction method and normalisation, an increase in AUC-PR was detected for the model without regularisation. When the power transformation was added, a significant improvement in AUC-PR and AUC-ROC was observed. For the random forest model the best results were obtained using the method of reducing the tails of the distribution and normalisation. In case of the XGBoost the number of estimators was calculated as the early stopping median of all validation subsets. The best results were achieved on normalised data without outliers.

Figure A7 in Appendix shows the learning process of a feedforward neural network example. The graphs presented confirm the conclusions drawn by Son et al. (2019) about the improved results after the application of the power transformation. Furthermore, one can see the huge impact of applying normalisation – smoothing and increasing learning curves and much higher results (+ 0.18 on the AUC-PR metric). Subsequent components of the data preprocessing procedure turned out to be accurate: outlier reduction is a + 0.09 gain and + 0.05 after applying the power transformation. The results shown in the legend of each graph are the effect of early stopping, indicated by the red dot. From the results of this experiment, it was decided to build MLP models on the fully preprocessed data. The final MLP model is a two-layer model with 60 neurons, a dropout of 0.4, and regularisation of $L_1 = 0.00005$ in each hidden layer. The activation functions are the hyperbolic tangent and sigmoid, respectively. It is worth mentioning that different functions were tested, including tanh, sigmoid, linear, ReLU, SELU, and PReLU, on both layers. For ReLU and PReLU, for instance, the lowest values of AUC-ROC and AUC-PR metrices were achieved. When the linear function was applied, the learning curves were non-monotonic. Given the tests, we choose tanh, ensuring high results, and the sigmoid function, ensuring smoothed curves.

The model was trained in batches of 394 observations each for 5,000 epochs using the Adam algorithm with standard hyperparameters. The results obtained by the MLP model are simply spectacular – AUC-PR = 0.712 (0.238 overfitting), AUC-ROC = 0.93 (0.07 overfitting).

Due to the similarity in the training process between the MLP and CNN models, the procedure for selecting the preprocessing variant, architecture, and hyperparameters was analogous for the latter. The learning curves presented in Figure A8 a–c in the Appendix also illustrate the positive relationship between the complexity of the data preprocessing procedure and the results (AUC-PR). In this case, however, it should be emphasised that the preliminary experiments were conducted on an architecture that turned out to be fundamentally different from the optimal one. In addition, the CNN model presented was trained with only 100 epochs while the MLP was trained with 400. The model whose learning process is illustrated in graphs a–c of Figure A8 in Appendix consists of two consecutive convolutional layers (same) 3 × 3; #$F$ are 32 and 64 respectively, the next layers are average pooling 2 × 2, $p = 0$, and convolution (same) 1 × 1 with 32 filters, all flattened and passed to a dense two-layer network (512,256 neurons). The convolutional layer activations are ReLU functions and dense sigmoid, and the optimiser is RMSprop (Hinton, Srivastava & Swersky, 2012) with standard hyperparameters. The input data are 8 × 8 images. For comparison, Figure A8 d in the Appendix shows the learning curves for the same validation division with full preprocessing and the best architecture found (inception blocks, see Table 2 for details). The model in Figure A8 d was trained at 450 epochs, so the comparison with graphs a–c is the progress of graph d up to 100 epochs. Model d reached lower metrics after 100 epochs, but the observed learning curves are almost perfectly smooth and increasing, which justifies training the model on a larger number of epochs. Ultimately, model d achieved noticeably higher metrics, much lower overtraining, and its training process was not objectionable.

In most of the prediction horizons, the architecture based on inceptive blocks turned out to be the best. As a rule, we searched for the best possible network for 8 × 8 images; then new variables were generated, and the process of searching for architecture for models trained on 20×20 images was started by checking the best architecture for models trained on 8 × 8 images. During the experiments in the one-year horizon, it turned out that there was no sense testing a separate architecture for models taking into account a larger number of features – the process was computationally very time-expensive and did not bring any noticeable results. Thus, the 20 × 20 models in the sense of architecture from the 8 × 8 models differ only in the dimensions of the output tensors of the individual layers and the differences between the dimensions of the input tensors. For the one-year horizon, the application of the image magnification method did not improve the results; the results obtained were AUC-PR 0.541 (0.233 overfitting) and AUC-ROC 0.881 (0.074 overfitting). However, fewer epochs were generally needed to achieve similar metric values (Table 2), and the learning curves were even smoother.

Table 1 shows the results of the cross-validation performed for the logistic regression for each of the predictive horizons analysed. The numbers presented are the average of all validation splits, and the models were trained each time on data without outliers, after power transformation, and normalised. In the context of the AUC-PR metric, it should be emphasised that the naive classifier (ie assigning a label of 1 to each observation) achieves a result equal to the sample balance. An exception to the decreasing relationship between the results and the prediction horizon is the last, five-year horizon. However, in the analysis of the results, it is worth noting not only the absolute values of the metrics, but also the difference between them and the metrics of the naive classifier, ie, the sample balance in the case of AUC-PR and the value of 0.5 in the case of AUC-ROC. The longer the prediction horizon, the more difficult the task, not only because of the less informative predictors, but also because of the decreasing balance. When a proper data preprocessing method is applied, logistic regression achieves really good results. What is extremely noticeable, however, is the increased difficulty when the time horizon is extended: one more year equals almost twice the AUC-PR results. On the AUC-ROC metric, admittedly there is also a noticeable decrease, but it is much lower relative to the naive classifier (model added value) than in the case of AUC-PR, which confirms the validity of using the latter metric as the main one. In addition, the very low overtraining of the models (especially compared with the more comprehensive ones) should be noted.

The results of the random forest validation are presented in Table 1. Increasing the prediction horizon significantly decreased the metrics, the exception to their negative relationship being the four-year horizon. The values of the hyperparameters found using the random search are worth noting; they are very similar between horizons. Random forests proved to be a much better model than logit in terms of both AUC-PR and AUC-ROC but much more overfitted. As mentioned earlier, the data preprocessing variant for decision tree-based models was treated as a hyperparameter, and so *q* stands for outlier reduction (limiting distributions with quantiles of order $Q_{0.005}$ and $\boldsymbol{Q}_{0.995}$), *p* stands for power transformation, and *n* is normalisation. The use of more than one method is denoted by +, and the absence of any method is denoted by -.

XGBoost is the model that is often indicated as the best in the literature, and its cross-validation results are presented in Table 1. XGBoost performed much better than logit and random forest. In the context of the area under the ROC curve, noticeably worse results were obtained compared to Zięba et al. (2016). The hyperparameters found by a random search, as with random forest, are very similar between horizons. However, the models presented differ in the parameters responsible for regularisation ($\gamma$, $L_1$, $L_2$). There is no relationship between the lengthening of the time horizon and the optimal number of estimators (early stopping, line median of estimators), the fastest peak was reached for the one-year horizon. XGBoost also generally exhibits less overtraining than random forests, which is certainly a result of the model's inclusion of regularisation components. In addition, the relatively high optimal fraction of columns, or attributes in each case, suggests the possibility of using variables that are even strongly correlated. XGBoost is the model that most often performed best on raw data.

Very good results were obtained for the MLP model in particular in view of the results reported by Zięba et al. (2016) for models of this class – 0.514–0.699 AUC-ROC, which means that in some cases the models built differed little in performance from the naive classifiers. The models presented in this study were found to be comparable with the best XGBoost models from the reference study in terms of AUC-ROC (ie, the best models from that study). Extremely high results (the highest of all models) were obtained primarily in the main metric: AUC-PR, for the one-year horizon, the model on average (in the context of the cut-off threshold) correctly classified significantly more than 2/3 of bankrupts while noting very high AUC-ROC. The huge impact of using the full preprocessing procedure should be emphasised here, as on normalised data the best MLP model obtained an AUC-PR of 0.457, which is an even larger gain than in the reports of Pawełek (2019) and Son et al. (2019). The results presented in Table 2 were achieved after applying the full preprocessing procedure, ie, $q + p + n$. Moreover, the dense neural network has much lower overfitting than decision tree-based models. The models built for different horizons are very similar,

**Table 1.** Hyperparameters and results for the best models, all horizons

| Model | Horizon (balance) | 1 year (6.94%) | 2 years (5.25%) | 3 years (4.71%) | 4 years (3.93%) | 5 years (3.86%) |
|---|---|---|---|---|---|---|
| Logit | λ | 1 | 0.08 | 0 | 0.05 | 0.03 |
| | Results | | | | | |
| | AUC-PR | 0.497 | 0.28 | 0.251 | 0.221 | 0.27 |
| | overfitting PR | 0.05 | 0.067 | 0.035 | 0.07 | 0.075 |
| | AUC-ROC | 0.852 | 0.782 | 0.762 | 0.726 | 0.764 |
| | overfitting ROC | 0.003 | 0.022 | 0.03 | 0.04 | 0.036 |
| Random forests | max depth | 12 | 12 | 13 | 12 | 8 |
| | max number of characteristics | 52 | 58 | 54 | 55 | 33 |
| | min observations to divide | 4 | 18 | 8 | 10 | 5 |
| | min observations in the leaf | 4 | 17 | 6 | 9 | 4 |
| | number of trees | 200 | 200 | 200 | 600 | 100 |
| | preprocessing | *q+n* | *p+n* | - | *q+p+n* | *p+n* |
| | Results | | | | | |
| | AUC-PR | 0.535 | 0.337 | 0.284 | 0.292 | 0.264 |
| | overfitting PR | 0.416 | 0.412 | 0.67 | 0.603 | 0.564 |
| | AUC-ROC | 0.89 | 0.84 | 0.829 | 0.817 | 0.795 |
| | overfitting ROC | 0.1 | 0.1367 | 0.168 | 0.175 | 0.192 |
| XGBoost | max depth | 6 | 7 | 5 | 5 | 4 |
| | learning rate η | 0.08 | 0.03 | 0.07 | 0.1 | 0.07 |
| | column fraction | 0.8 | 0.8 | 1 | 0.8 | 0.9 |
| | raw fraction | 0.9 | 0.9 | 0.9 | 0.9 | 0.8 |
| | loss reduction γ | 3 | 5 | 0.08 | 4 | 2 |
| | regularisation $L_1$ | 0.4 | 1 | 1 | 0.4 | 0.1 |
| | regularisation $L_2$ | 0.4 | 0.2 | 0 | 1 | 0.3 |
| | preprocessing | *q+n* | - | *q+n* | - | *n* |
| | estimators median | 122 | 271 | 209 | 246 | 164 |
| | Results | | | | | |
| | AUC-PR | 0.596 | 0.411 | 0.39 | 0.392 | 0.355 |
| | overfitting PR | 0.392 | 0.52 | 0.598 | 0.571 | 0.54 |
| | AUC-ROC | 0.909 | 0.869 | 0.864 | 0.86 | 0.835 |
| | overfitting ROC | 0.09 | 0.124 | 0.136 | 0.137 | 0.145 |

Source: Authors' calculations

and in three cases the only difference is the number of epochs (1, 2, 5 years). The convention for presenting the neural network architecture is as follows: + denotes the presence of a layer in a given model, and early stopping denotes the median of the epochs in which the best AUC-PR results were achieved in cross-validation. Optimisers were run with the default hyperparameters of the keras library.

The results and architecture of convolutional neural networks built for 8 × 8 images, ie, without artificial variable generation, are presented in Table 2. CNN models are more different from MLP models;

in the case of the three-year horizon the architecture used is fundamentally different from the rest. The concept of this model is based, among others, on the models of AlexNet (2012), VGG (2014) or the article 'Network in network' (2013), while the rest is built on inception modules and the GoogLeNet model (Szegedy et al., 2015). The first submodule of the inception module is always formed by a 1 × 1 convolution layer, the second one is a 3 × 3 convolution preceded by a 1 × 1 convolution, the third one is a 5 × 5 convolution preceded by a 1 × 1 convolution, and the last, the fourth one, is a max pooling layer followed by a 1 × 1 convolution (optimised inception module, Figure 2). The structure of the modules is shown as follows: F1 denotes the number of filters of the first submodule, the subscripts denote the subsequent submodules (digit), and in and out denote the number of filters of the 1 × 1 convolution and the following 'main' convolution, ie 3 × 3 or 5 × 5, respectively. The padding from the definition of the inception module is always the same, the stride is 1 × 1, and the activation function is always the same. The number of filters is not given in pooling layers because it results from the number of channels of the input tensor. Relative to the MLP models, the CNNs performed much worse but much faster in terms of the number of epochs (in the time dimension, the MLP models trained faster than the CNN models – no GPU). The 8 × 8 model performed satisfactorily, generally ranking behind MLP and XGBoost and ahead of random forests and logit (AUC-PR). For the three-year horizon, the CNN model built for the 8 × 8 data even performed slightly better than XGBoost (AUC-PR). As the prediction horizon increased, the overfitting also increased dramatically.

The results of the convolutional neural networks trained on 20 × 20 data, ie, after generating artificial variables, are reported in Table 2. Since these models have an architecture analogous to the 8 × 8 models, the presentation has been simplified by considering only the relevant details. It can be concluded that the proposed method of image enlargement turned out to be successful; beyond the one-year horizon, an increment of results was recorded, and for the two-, four- and 5-year horizons, the increment turned out to be statistically significant at the 10% level (test for equality of median AUC-PR validation splits with the alternative of superiority of the 20 × 20 model). In terms of the AUC-PR metric, the CNN models after the variable generating technique were even superior to the XGBoost models for 2-, 3-, and 4-year horizons and also recording lower overtraining. Moreover, at the expense of higher overfitting, the maximum results in the context of epochs were achieved even faster than in models with fewer features. On the test sets it was decided to test only one type of convolutional neural networks, the decision whether it was an 8 × 8 or 20 × 20 variant was made on the basis of Wilcoxon tests: if the increment of metrics caused by the generation of new variables was statistically significant at the 10% level then the more comprehensive model was chosen.

## 4.4. Results on the test set

We present the results of the best-in-class models (best logit, random forest, XGBoost, MLP, and CNN) trained on the entire training sample, the data used to train and validate the individual models in the previous subsections. The hyperparameters and architecture of the models are presented in Tables 1 and 2. The test data were processed separately for each model considering the components by which the results presented in the previously mentioned tables were obtained, using the characteristics of the training data.

The graphs presented in Figure 3, in addition to the results of the trained models – the curves and the areas below them (legend) – also contain information about the results that would be obtained from a naive classifier, ie one that assigns each observation a bankrupt label. This makes it possible to quantify the number of dependencies that the model actually learned. It turns out that the results obtained in some cases differ from those obtained in cross-validation, with generally higher results obtained on the test set. The differences between the validation and test results may suggest that too few data were used as a test set. However, owing to the huge imbalance and the number of observations, it was considered better that more data were used for training and validation. It can be surmised that generally higher test metrics are the result of the strategy adopted, and by doing the opposite there would be a risk of undertraining the models. The biggest surprise is the positive deviation of the PR curves for the four-year horizon. As expected, the MLP models outclassed all others in terms of the area under the PR curve at almost every prediction horizon, recording excellent precision for horizons 1–3 for the longest time. The excellent precision of the logit at the 5-year horizon, which is maintained for the longest time, is also a surprise. The convolutional neural network model in the 20 × 20 variant maintained excellent precision the longest

**Table 2.** MLP and convolutional neural networks: hyperparameters and results for best networks, all horizons

| Network | Horizon (balance) | 1 year (6.94%) | 2 years (5.25%) | 3 years (4.71%) | 4 years (3.93%) | 5 years (3.86%) |
|---|---|---|---|---|---|---|
| MLP neural networks | Architecture | | | | | |
| | dense layer | + | + | + | + | + |
| | neurons | 60 | 60 | 60 | 60 | 60 |
| | activation | tanh | tanh | tanh | tanh | tanh |
| | dropout | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 |
| | regularisation | $L_1$ 0.00005 | $L_1$ 0.00005 | | $L_2$ 0.00005 | $L_1$ 0.00005 |
| | dense layer | + | + | + | + | + |
| | neurons | 60 | 60 | 60 | 60 | 60 |
| | activation | sigmoid | sigmoid | sigmoid | sigmoid | sigmoid |
| | dropout | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 |
| | regularisation | $L_1$ 0.00005 | $L_1$ 0.00005 | | $L_2$ 0.00005 | $L_1$ 0.00005 |
| | dense layer | + | + | + | + | + |
| | Neurons | 1 | 1 | 1 | 1 | 1 |
| | activation | sigmoid | sigmoid | sigmoid | sigmoid | sigmoid |
| | early stopping | 4,873 | 3,972 | 4,892 | 6,843 | 7,243 |
| | batch size | 394 | 394 | 394 | 394 | 394 |
| | Optimiser | Adam | Adam | Adam | Adam | Adam |
| | Results | | | | | |
| | AUC-PR | 0.717 | 0.611 | 0.632 | 0.629 | 0.519 |
| | overfitting PR | 0.238 | 0.214 | 0.35 | 0.282 | 0.34 |
| | AUC-ROC | 0.929 | 0.895 | 0.9 | 0.909 | 0.883 |
| | overfitting ROC | 0.065 | 0.079 | 0.098 | 0.085 | 0.106 |
| 8×8 convolutional neural networks | Architecture | | | | | |
| | inception module | + | + | - | + | + |
| | $\#F_1$ | 4 | 4 | | 4 | 4 |
| | $\#F_{2in}/\#F_{2out}$ | 1/4 | 1/4 | | 1/4 | 1/4 |
| | $\#F_{3in}/\#F_{3out}$ | 1/4 | 1/4 | | 1/4 | 1/4 |
| | $\#F_{4out}$ | 4 | 4 | | 4 | 4 |
| | Activation | ReLU | ReLU | | ReLU | ReLU |
| | inception module | + | + | - | + | + |
| | $\#F_1$ | 8 | 8 | | 8 | 8 |
| | $\#F_{2in}/\#F_{2out}$ | 4/8 | 4/8 | | 4/8 | 4/8 |
| | $\#F_{3in}/\#F_{3out}$ | 4/8 | 4/8 | | 4/8 | 4/8 |
| | $\#F_{4out}$ | 8 | 8 | | 8 | 8 |
| | Activation | ReLU | ReLU | | ReLU | ReLU |
| | Convolution | - | - | + | - | - |
| | Size | | | 3 × 3 | | |

Continued **Table 2.** MLP and convolutional neural networks: hyperparameters and results for best networks, all horizons

| Network | Horizon (balance) | 1 year (6.94%) | 2 years (5.25%) | 3 years (4.71%) | 4 years (3.93%) | 5 years (3.86%) |
|---|---|---|---|---|---|---|
| 8×8 convolutional neural networks | *#F* | | | 32 | | |
| | Padding | | | *same* | | |
| | Stride | | | 1 × 1 | | |
| | Activation | | | ReLU | | |
| | Convolution | - | - | + | - | - |
| | Size | | | 3 × 3 | | |
| | *#F* | | | 64 | | |
| | Padding | | | *same* | | |
| | Stride | | | 1 × 1 | | |
| | Activation | | | ReLU | | |
| | Max pooling | - | - | + | - | - |
| | Size | | | 2 × 2 | | |
| | Padding | | | *Valid* | | |
| | Stride | | | 2 × 2 | | |
| | Convolution | - | - | + | - | - |
| | Size | | | 1 × 1 | | |
| | *#F* | | | 32 | | |
| | Stride | | | 1 × 1 | | |
| | Activation | | | ReLU | | |
| | Average pooling | + | + | - | + | + |
| | Size | 2 × 2 | 2 × 2 | 2 × 2 | 2 × 2 | 2 × 2 |
| | Padding | *valid* | *valid* | | *valid* | *valid* |
| | Stride | 2 × 2 | 2 × 2 | 2 × 2 | 2 × 2 | 2 × 2 |
| | flatten layer | + | + | + | + | + |
| | dense layer | + | + | + | + | + |
| | Neurons | 256 | 256 | 512 | 256 | 256 |
| | Activation | ReLU | ReLU | sigmoid | ReLU | ReLU |
| | Dropout | 0.4 | 0.4 | | 0.4 | 0.4 |
| | regularisation | $L_1$ 0.000025 | $L_2$ 0.00005 | $L_2$ 0.00005 | $L_2$ 0.0001 | $L_2$ 0.00005 |
| | dense layer | - | - | + | + | - |
| | neurons | | | 256 | 128 | |
| | activation | | | sigmoid | ReLU | |
| | dropout | | | | | |
| | regularisation | | | $L_2$ 0.00005 | $L_2$ 0.00005 | |
| | dense layer | + | + | + | + | + |
| | neurons | 1 | 1 | 1 | 1 | 1 |
| | activation | sigmoid | sigmoid | sigmoid | sigmoid | sigmoid |

Continued **Table 2.** MLP and convolutional neural networks: hyperparameters and results for best networks, all horizons

| **Network** | **Horizon (balance)** | **1 year (6.94%)** | **2 years (5.25%)** | **3 years (4.71%)** | **4 years (3.93%)** | **5 years (3.86%)** |
|---|---|---|---|---|---|---|
| 8×8 convolutional neural networks | early stopping | 418 | 350 | 112 | 220 | 326 |
| | batch size | 394 | 394 | 394 | 394 | 394 |
| | optimiser | Adam | Adam | RMSprop | Adam | Adam |
| | Results | | | | | |
| | AUC-PR | 0.547 | 0.393 | 0.392 | 0.342 | 0.306 |
| | overfitting PR | 0.219 | 0.404 | 0.43 | 0.414 | 0.392 |
| | AUC-ROC | 0.885 | 0.828 | 0.84 | 0.814 | 0.793 |
| | overfitting ROC | 0.07 | 0.147 | 0.136 | 0.156 | 0.169 |
| 20 × 20 convolutional neural networks | early stopping | 189 | 157 | 137 | 155 | 239 |
| | Results | | | | | |
| | AUC-PR | 0.542 | 0.456 | 0.401 | 0.394 | 0.345 |
| | overfitting PR | 0.233 | 0.447 | 0.489 | 0.576 | 0.553 |
| | AUC-ROC | 0.881 | 0.846 | 0.837 | 0.847 | 0.811 |
| | overfitting ROC | 0.074 | 0.143 | 0.152 | 0.151 | 0.173 |

Source: Authors' calculations

for the four-year horizon. The MLP model (best results of AUC-PR for each horizon and AUC-ROC for horizons 1–3) should undoubtedly be considered the best model for the set analysed for the purpose of this paper, preprocessing proved to be crucial. Thanks to outlier reduction technique, power transformation and normalization it was possible to train the model for many epochs, which with smooth and increasing learning curves affected the results very positively. With the exception of the five-year horizon, logistic regression turned out to be the worst model (the lowest AUC-PR results from the two–four year horizon and AUC-ROC every time). It is worth noting that decision tree-based models proved to be resistant to the preprocessing variant used, which resulted in second-to-last results for random forests compared with significant improvements for all other models. XGBoost ranked near the top despite its resistance to transformations of the predictor distributions. While feedforward neural networks are undoubtedly the best model, identifying the second-best algorithm is a difficult task, because depending on the time horizon, XGBoost and convolutional neural networks were alternately better.

In response to the main research question, convolutional neural networks, despite the fact that they were not created for modelling cross-sectional tabular data, do really well, recording some of the leading results. The undoubted advantage of convolutional models is the possibility to look at the analysis and presentation of data in a different way, because it is much easier for humans to analyse images than tables. The disadvantage of convolutional models in the context of this study is the need to spend a lot of time on data processing, development of vector-to-image conversion methods and their computation, and the selection of appropriate architecture and hyperparameters. The presented vector-to-image conversion method, along with the algorithm generating new predictors, seems to be suitable and perhaps would also find application in fields other than corporate bankruptcy prediction. The XGBoost model is definitely advantageous, owing to its ease of application, construction, and optimisation. Of course, during the development of the models presented, it was not possible to use GPUs, which would significantly speed up the process of training convolutional networks, perhaps making them competitive in this field. Before we started this work, it was expected that by extending the prediction horizon, the results would be weaker and weaker. In the context of the results obtained, this hypothesis cannot be confirmed, because it turns out that the amount of information

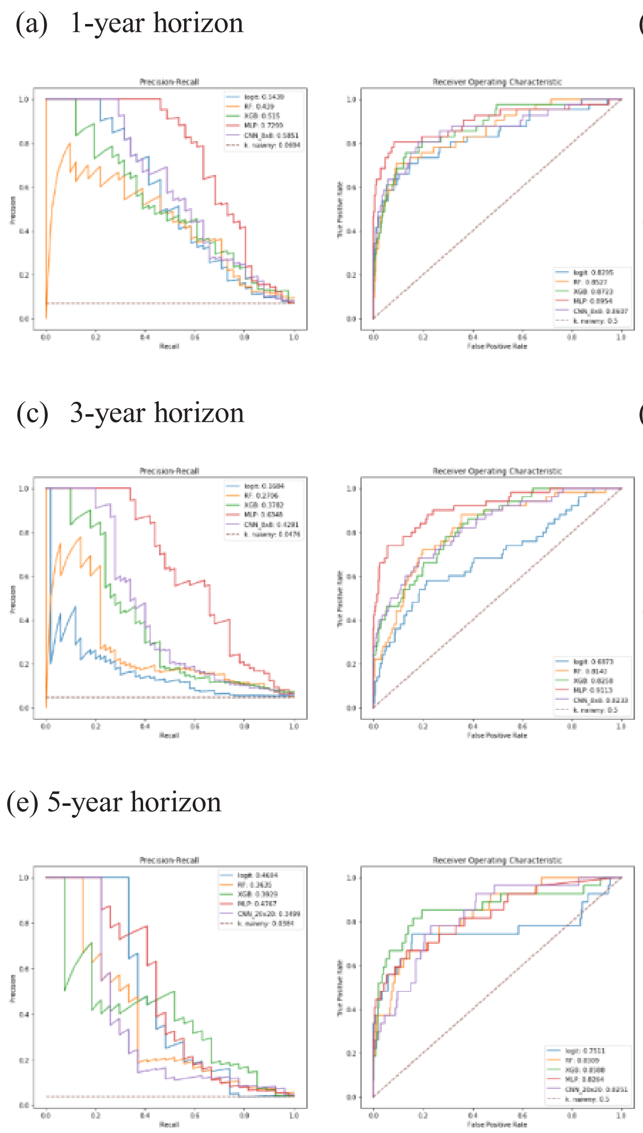


**Figure 3.** PR and ROC curves for all prediction horizons, test data
Source: Authors' calculations

that can be extracted from the financial indicators on the probability of bankruptcy of the company is much greater than assumed and goes beyond a linear analysis in relation to the parameters.

## 5. Conclusion

Modelling the likelihood of corporate bankruptcy using financial indicators is definitely a difficult topic but one worthy of attention. Research in this area may be of interest to state decision makers, economists, investors, entrepreneurs, and due to the specificity of the phenomenon, also to specialists in data analysis. Early detection of a potential bankrupt can save many fatal situations such as increased unemployment or loss of capital; on the other hand such a possibility creates opportunities for market speculators and allows effective control by the state by stimulating appropriate economic sectors. The lack of a real process generating data excludes the possibility of correct causal inference, which increases interest in models treated in other fields as 'black boxes', because all the attention is focused on prediction. This state of affairs focuses researchers on methodology, which contributes to the development of novel methods of analysis. Although there is no data generating process between financial indicators and bankruptcy probability, the correlation between these variables is evident. Some of the indicators fit directly into the definition of bankruptcy (for example, debt or liquidity ratios).

The right approach seems to be to treat the problem as a binary classification, distinguishing only the state of formal bankruptcy and the entity constantly prospering.

The fundamental problems indicated in the literature and detected in this study include the strong skewness of the label distribution, the presence of many missing data, outliers, the insufficient number of observations, and the skewness of the predictor distributions. The class imbalance was resolved by using appropriate success metrics (precision-recall curve and the area under it) and stratified randomisation to training, validation, and test sets. The problem of the insufficient number of observations was solved by using a 6-fold stratified cross-validation and simulating a real production process (separating the test set and excluding it from the analyses). Missing data were supplemented with the author's algorithm based on random forests, which seems to be a slightly more sophisticated method than median supplementation, and due to the small number of observations their removal should not be considered at all. Addressing the problems of outliers and skewness of the predictor distributions proved crucial. Outliers were reduced using a method that consisted of limiting the top and bottom distributions (0.5%); this limitation did not consist of removing observations, but of assigning a boundary value to a given attribute if it exceeded the boundary value. Owing to the skewness of distributions, a generalisation of the Box-Cox transformation, ie the Yeo-Johnson transformation was applied. The result of applying data preprocessing methods was a significant improvement of the obtained results, both in terms of models and generated images.

The main conclusion of the study is that convolutional neural networks, after appropriate data preparation, can be successfully used to analyse cross-sectional tabular data, especially in the problem of modeling the probability of corporate bankruptcy. The second, very important conclusion is the necessity of using appropriate preprocessing techniques in order to achieve good results with models based on parameters updated with a gradient – neural networks and logit. At the same time, models based on decision trees proved to be insensitive to the applied data transformations.

## References

Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance*, *23*(4), 589–609

Box, G. E., Cox, D.R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, *26*(2), 211–243

Chen, M. Y. (2011). Predicting corporate financial distress based on integration of decision tree classification and logistic regression. *Expert Systems with Applications*, *38*(9), 11,261–11,272

Heo, J., & Yang, J. Y. (2014). AdaBoost based bankruptcy forecasting of Korean construction companies. *Applied Soft Computing*, *24*, 494–499

Hinton, G., Nitish S. & Swersky K. (2012). Divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning*. Technical Report.

Hosaka, T. (2019). Bankruptcy prediction using imaged financial ratios and convolutional neural networks. *Expert Systems with Applications*, *117*, 287–299

Kim, M. J., & Kang, D. K. (2010). Ensemble with neural networks for bankruptcy prediction. *Expert Systems with Applications*, *37*(4), 3,373–3,379

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in *Neural Information Processing Systems*, *25*, 1,097–1,105

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. arXiv preprint arXiv:1312.4400. 57

Ohlson, J. A. (1980). Financial ratios and the probabilistic prediction of bankruptcy. *Journal of Accounting Research*, 109–131

Pawełek, B. (2019). Extreme Gradient Boosting Method in the Prediction of Company Bankruptcy. *Statistics in Transition. New Series*, *20*(2), 155–171

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

Son, H., Hyun, C., Phan, D., & Hwang, H. J. (2019). Data analytic approach for bankruptcy prediction. *Expert Systems with Applications*, *138*, 112,816

Szegedy, C. et al. (2015). Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9

Tomczak, S. (2014). Comparative analysis of liquidity ratios of bankrupt manufacturing companies. *Business and Economic Horizons, 10*(3), 151–164

Tomczak, S. (2014). Comparative analysis of the bankrupt companies of the sector of animal slaughtering and processing. *Equilibrium. Quarterly Journal of Economics and Economic Policy, 9*(3), 59–86

Tomczak, S. (2014). The early warning system. *Journal of Management and Financial Sciences, 7*(16), 51–74

Veganzones, D., & Séverin, E. (2018). An investigation of bankruptcy prediction in imbalanced datasets. *Decision Support Systems*, *112*, 111–124

Vellamcheti, S., & Singh, P. (2020). Class Imbalance Deep Learning for Bankruptcy Prediction. In *2020 First International Conference on Power, Control and Computing Technologies (ICPC2T)*, 421–425. IEEE

Wyrobek, J., & Kluza, K. (2018). Efficiency of gradient boosting decision trees technique in Polish companies' bankruptcy prediction. In *International Conference on Information Systems Architecture and Technology*, 24–35

Yeo, I. K., & Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, *87*(4), 954–959

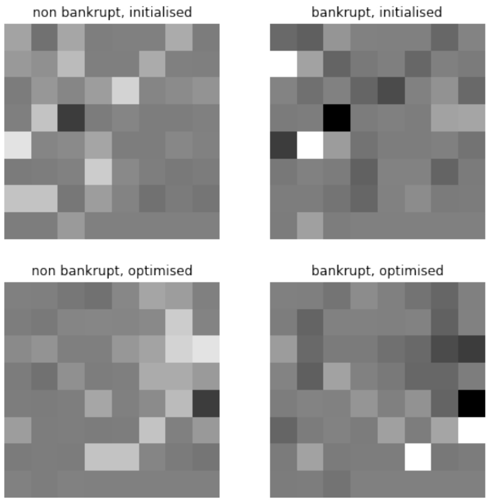
Zhang, G., Hu, M. Y., Patuwo, B. E., & Indro, D. C. (1999). Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis. *European Journal of Operational Research, 116*(1), 16–32

Zięba, M., Tomczak, S. K., & Tomczak, J. M. (2016). Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. *Expert Systems with Applications, 58*, 93–101

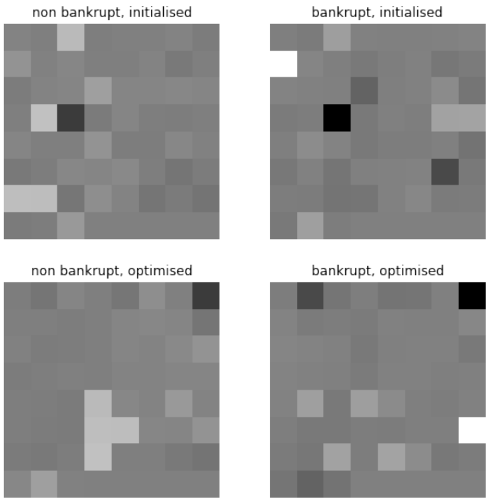
Zmijewski, M. E. (1984). Methodological issues related to the estimation of financial distress prediction models. *Journal of Accounting Research*, 59–82

## Appendix

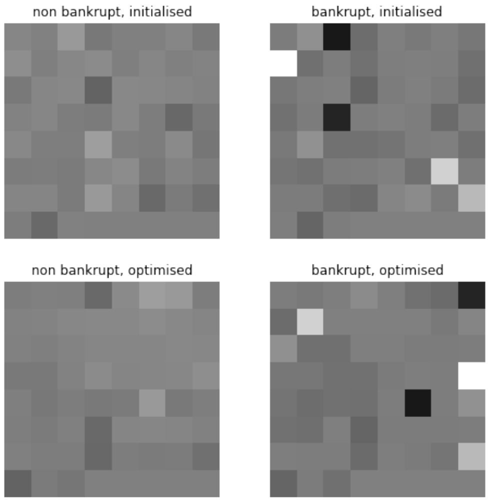
(a) fold 1



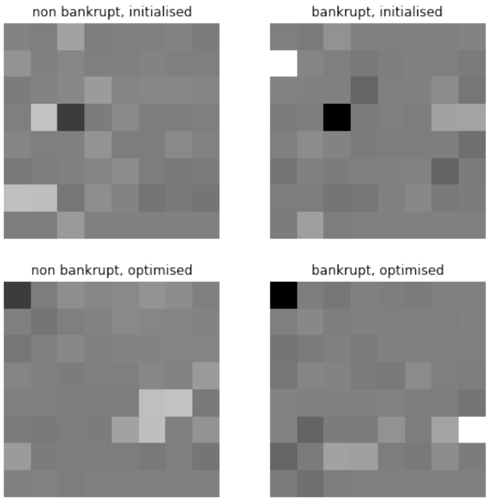
(b) fold 2



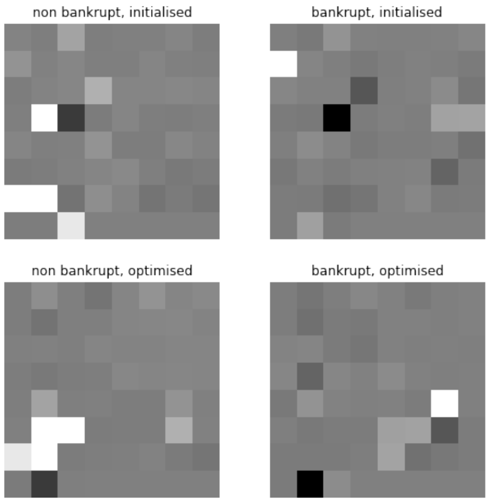
(c) fold 3



(d) fold 4



(e) fold 5



(f) fold 6

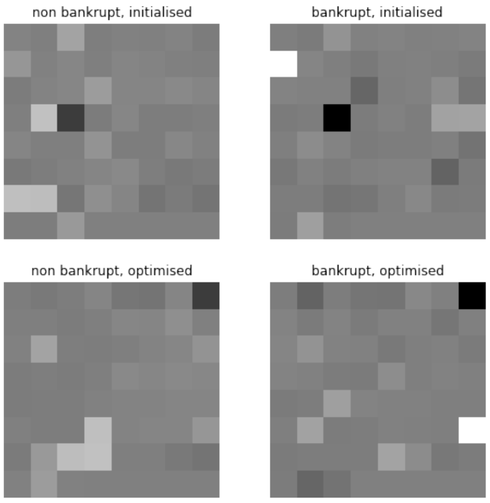


**Figure A1:** 8 × 8 images, raw data, normalised, validation, one-year horizon
Source: Authors' calculations

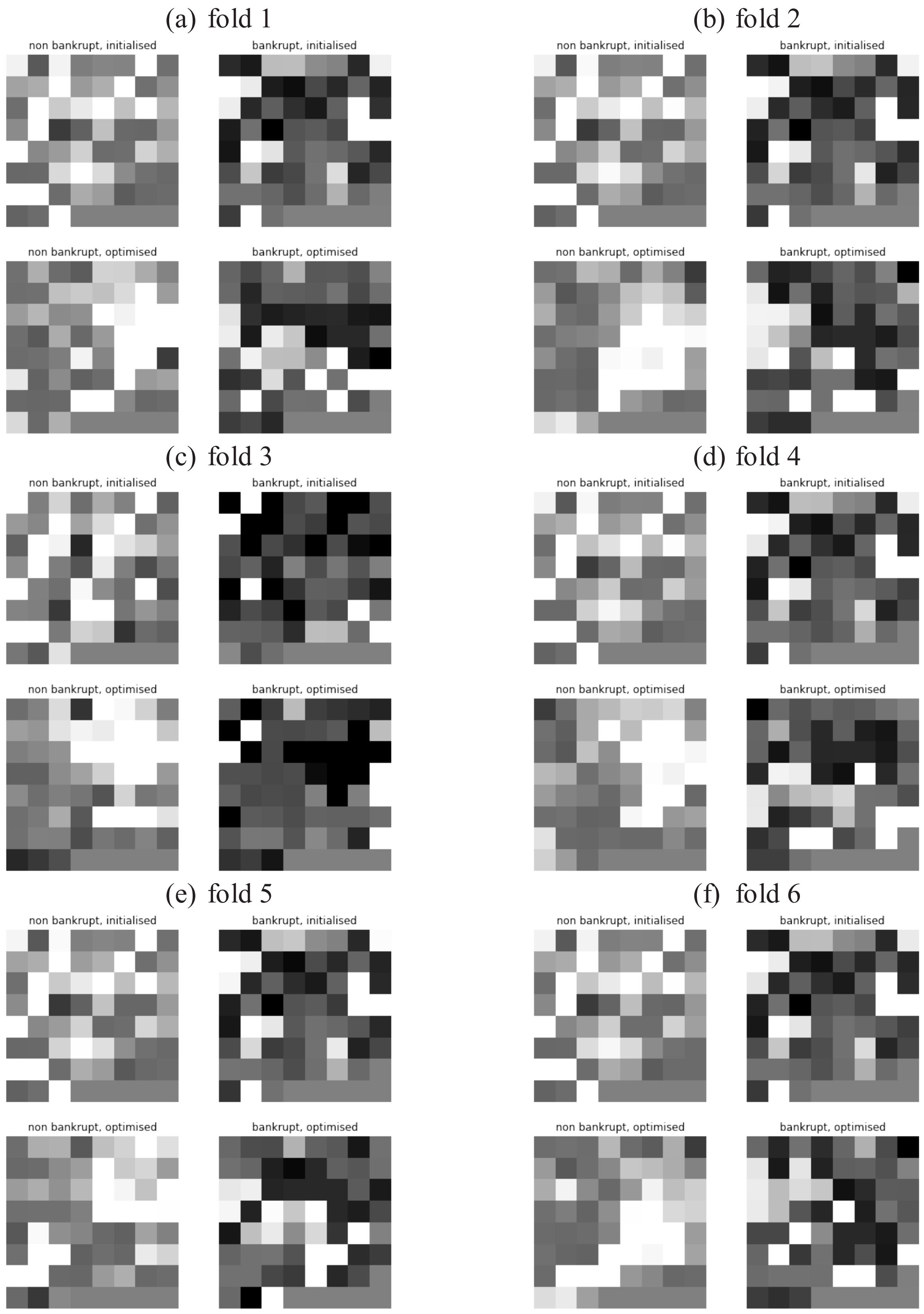


**Figure A2:** 8 × 8 images, outlier-free, normalised, validation, one-year horizon
Source: Authors' calculations

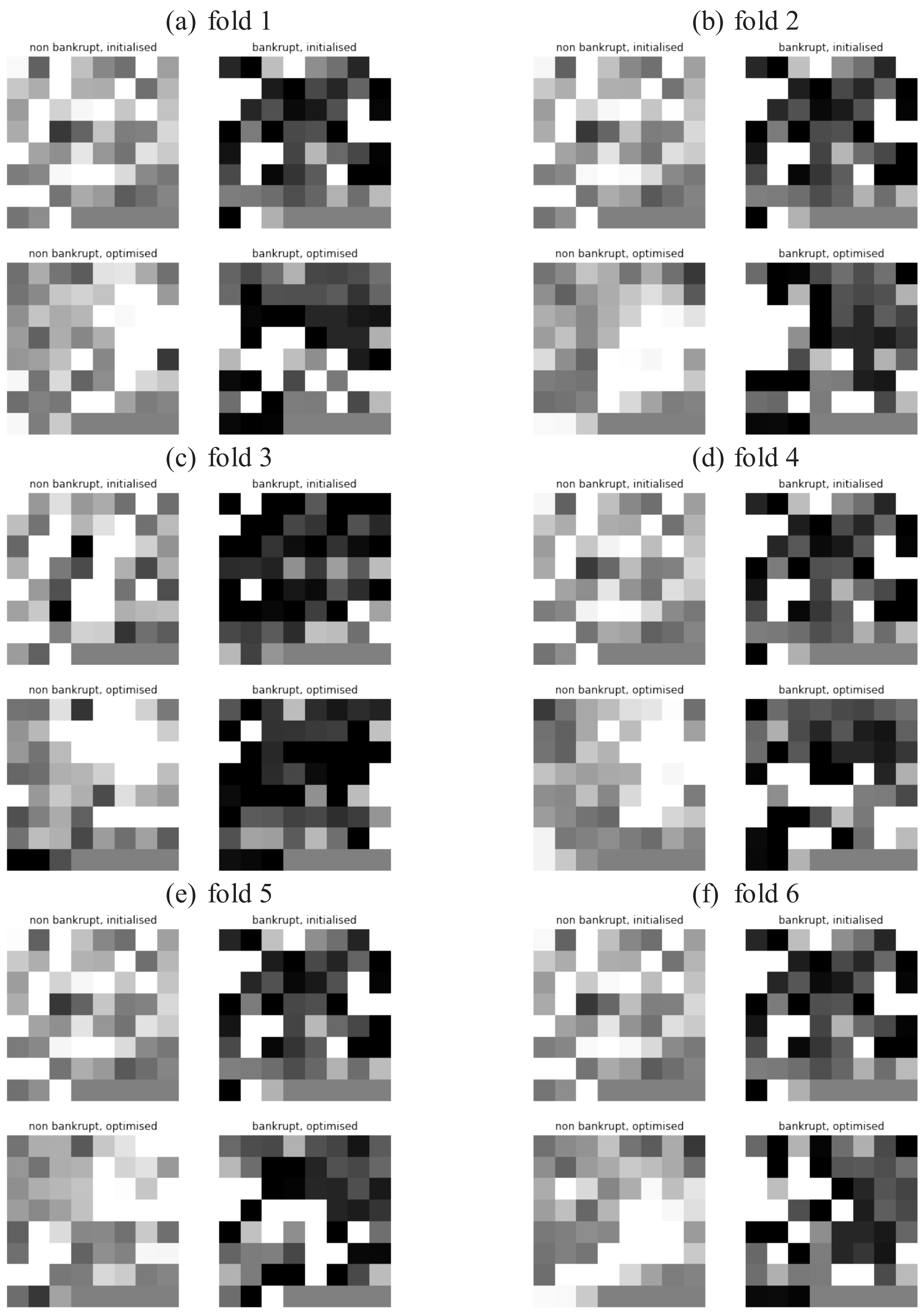


**Figure A3:** 8 × 8 images, outlier-free, power transformation, normalised, validation, one-year horizon
Source: Authors' calculations.u

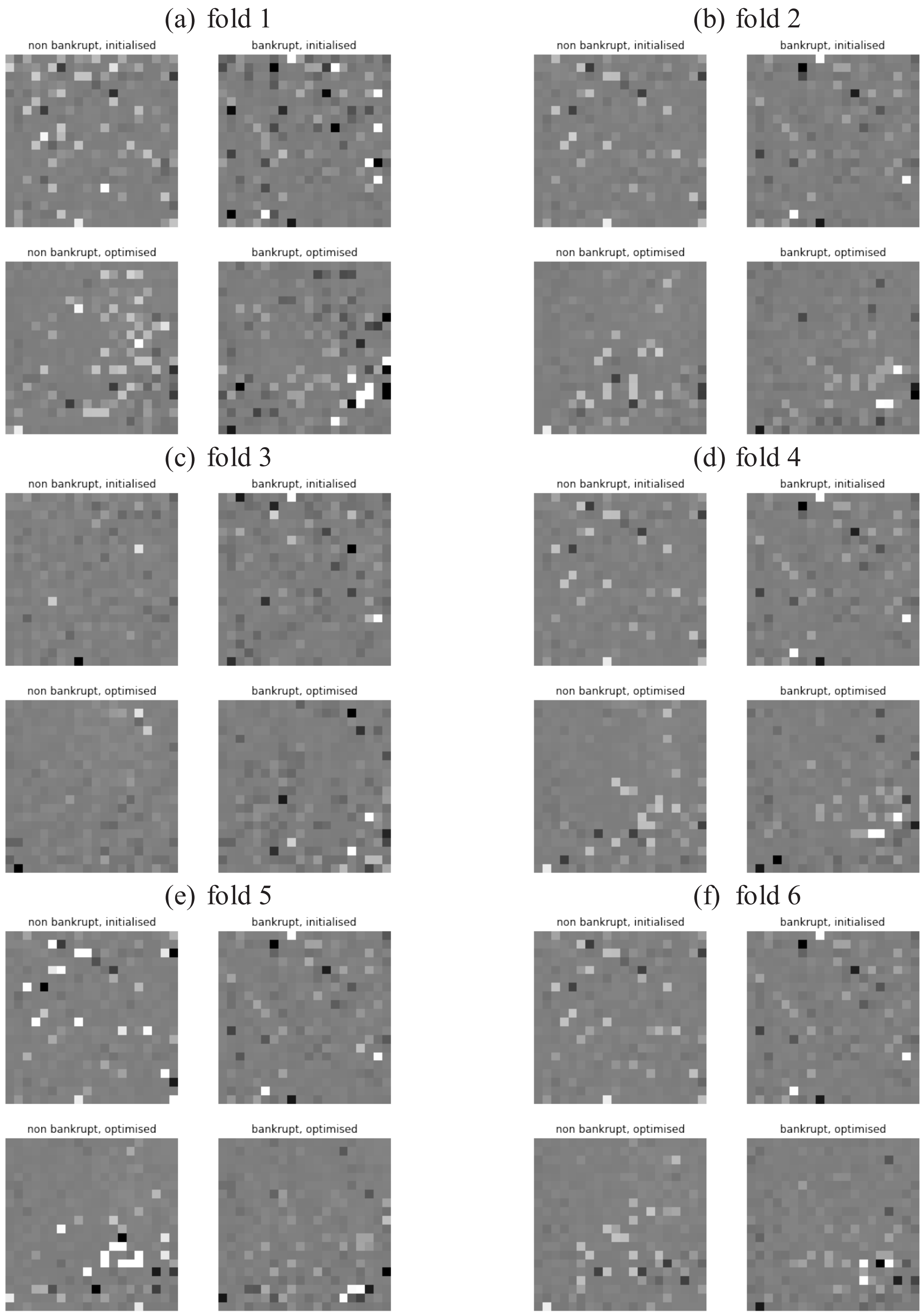


**Figure A4:** 20 × 20 images, raw data, normalised, validation, one-year horizon
Source: Authors' calculations

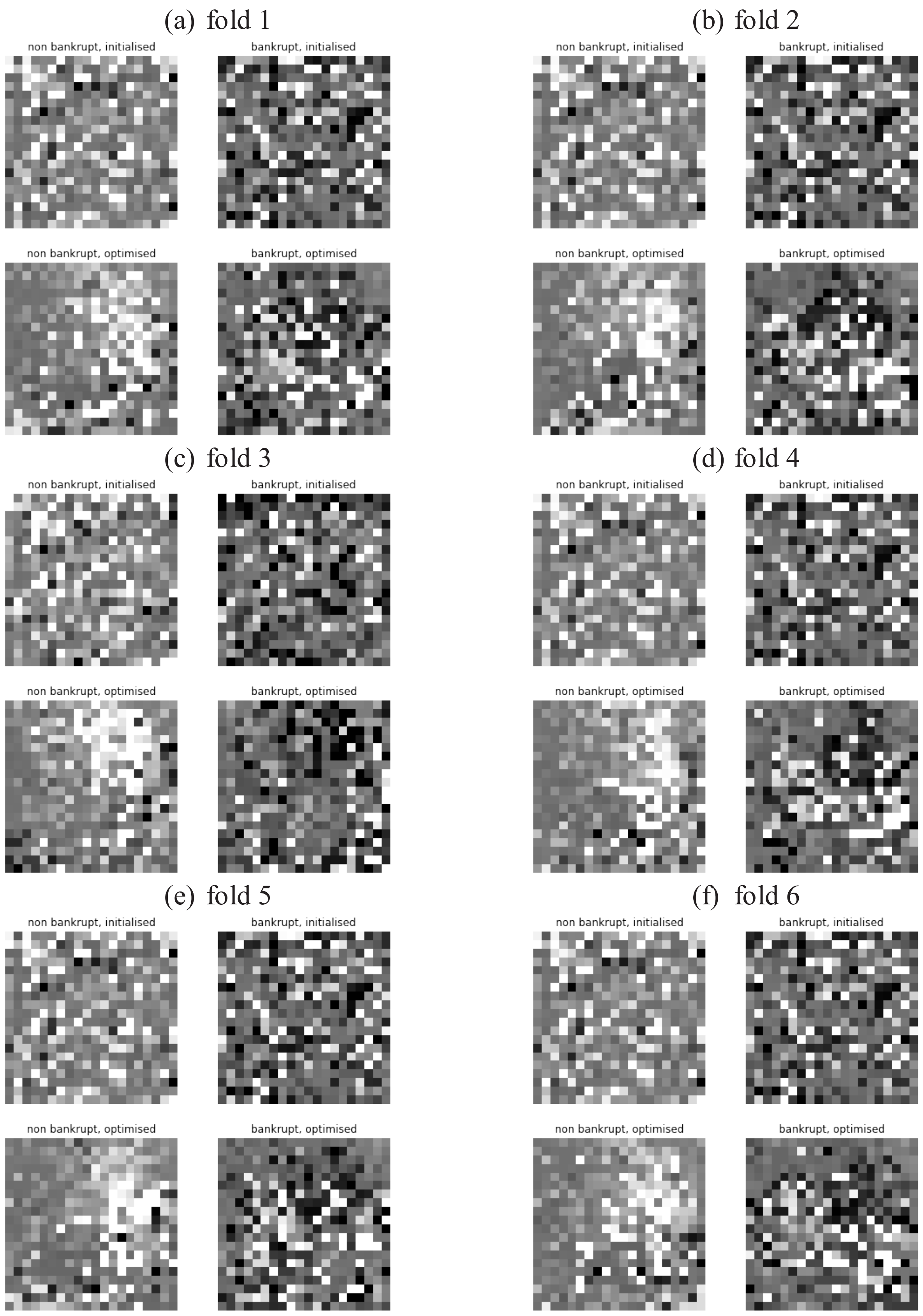


**Figure A5:** 20 × 20 images, data without outliers, normalised, validation, one-year horizon
Source: Authors' calculations

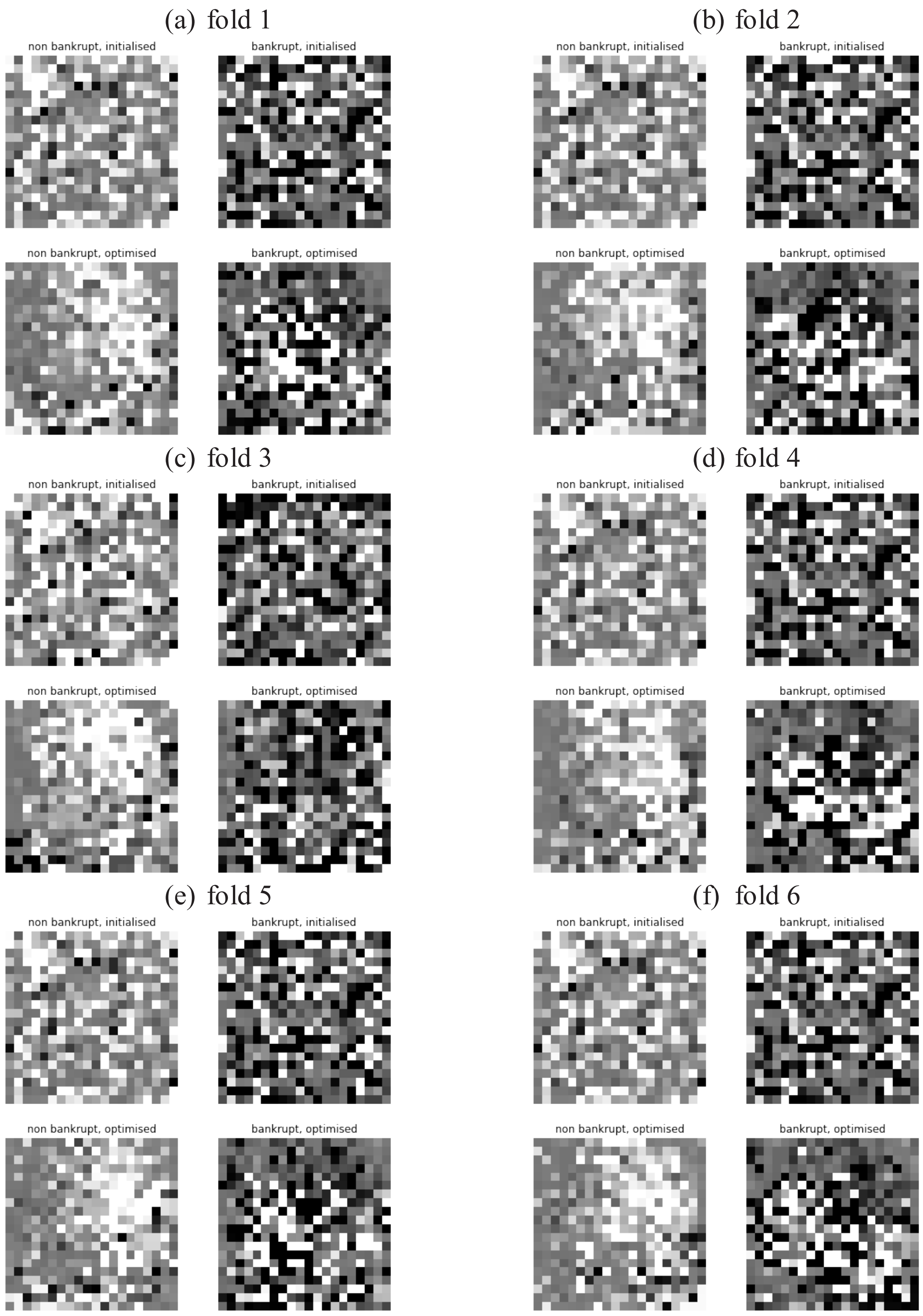
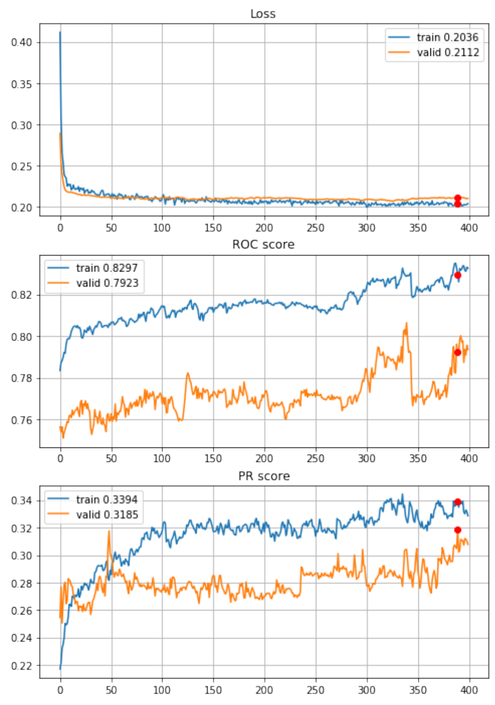
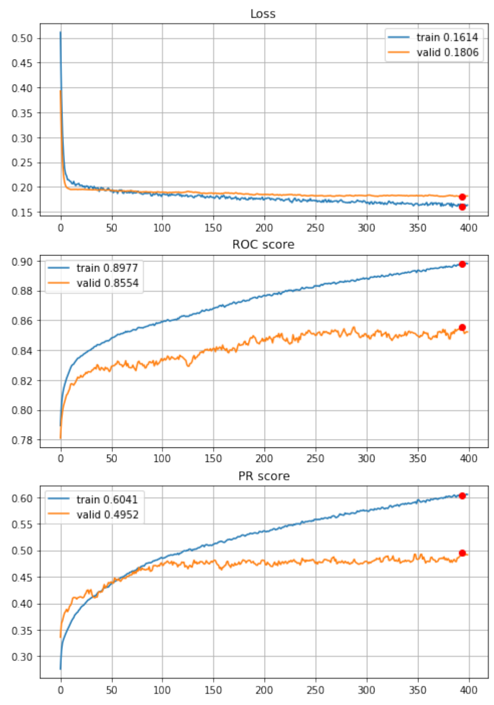


**Figure A6:** 20 × 20 images, outlier-free, power transformation, normalised, validation, one-year horizon
Source: Authors' calculations

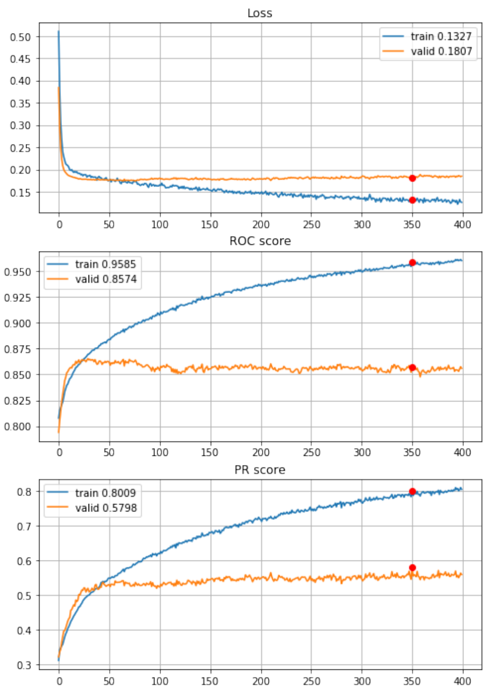
(a) raw data



(b) normalised data



(c) outlier-free data, normalised



(d) outlier-free data, power transformation, normalised

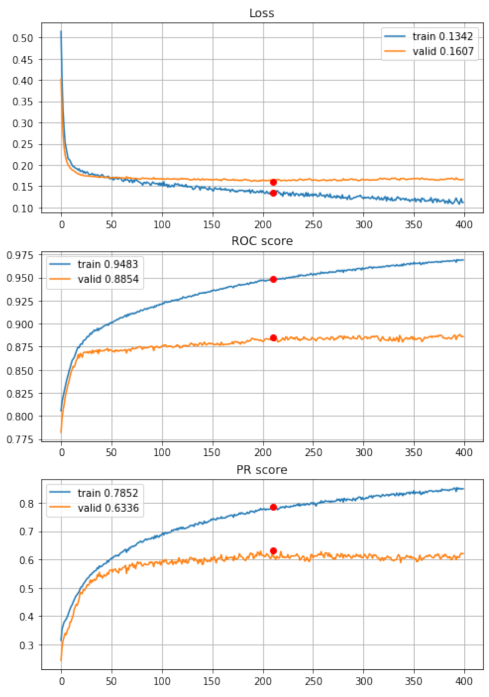
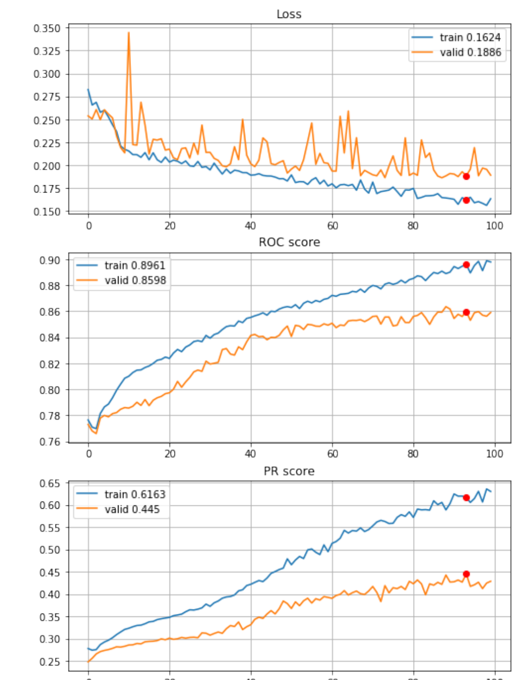
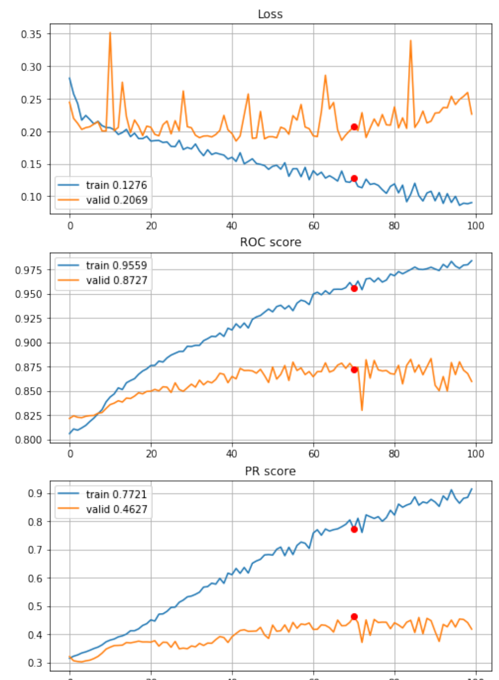


**Figure A7:** Learning curves depending on the variant of preprocessing with early stopping for the MLP model example.
Source: Authors' calculations

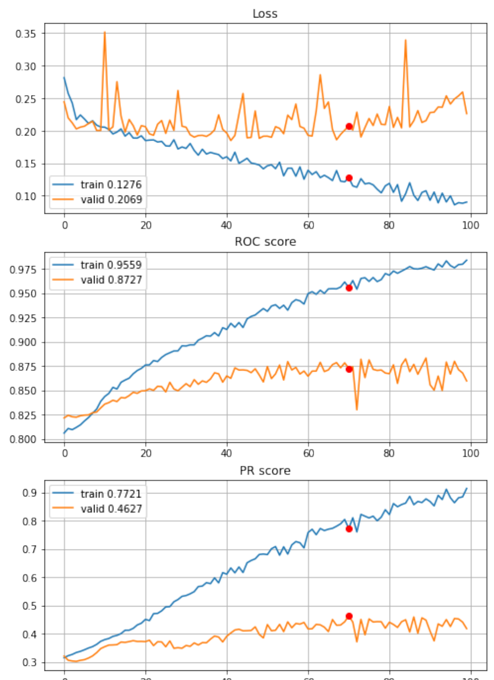
(a) normalised data, inadequate architecture



(b) outlier-free data, normalised, inadequate architecture



(c) outlier-free data, power transformation, normalised, inadequate architecture



(d) outlier-free data, power transformation, , normalised,appropriate architecture

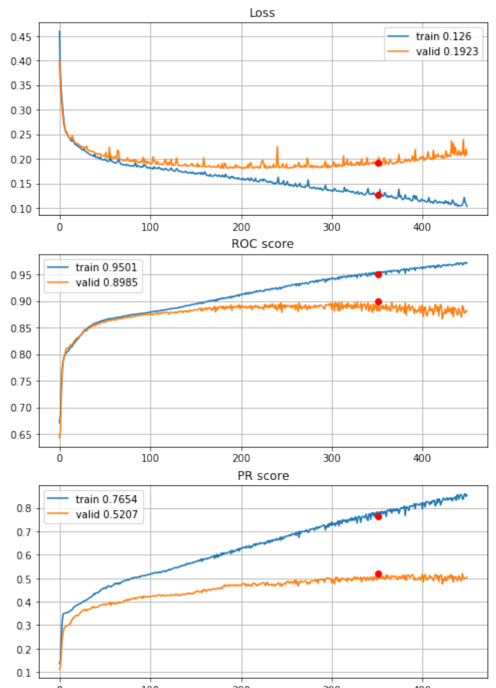


**Figure A8:** Learning curves depending on the variant of preprocessing with early stopping for CNN model examples
Source: Authors' calculations