Feature learning for class separability applied to COVID-19 chest X-ray case study

VII.

Source

Charte, D., Sevillano-García, I., Lucena-González, M. J., Martín-Rodríguez, J. L., Charte, F., & Herrera, F. (2021, September). Slicer: Feature Learning for Class Separability with Least-Squares Support Vector Machine Loss and COVID-19 Chest X-Ray Case Study. In *International Conference on Hybrid Artificial Intelligence Systems* (pp. 305-315). Springer, Cham.

Abstract

Datasets from real-world applications usually deal with many variables and present difficulties when modeling them with traditional classifiers. There is a variety of feature selection and extraction tools that may help with the dimensionality problem, but most of them do not focus on the complexity of the classes. In this paper, a new autoencoder-based model for addressing class complexity in data is introduced, aiming to extract features that present classes in a more separable fashion, thus simplifying the classification task. This is possible thanks to a combination of the standard reconstruction error with a least-squares support vector machine loss function. This model is then applied to a practical use case: classification of chest X-rays according to the presence of COVID-19, showing that learning features that increase linear class separability can boost classification performance. For this purpose, a specific convolutional autoencoder architecture has been designed and trained using the recently published COVIDGR dataset. The proposed model is evaluated by means of several traditional classifiers and metrics, in order to establish the improvements caused by the extracted features. The advantages of using a feature learner and traditional classifiers are also discussed.

Keywords

Feature learning - Class separability - Autoencoders.

VII.1. Introduction

Data classification [1] is one of the most studied problems in machine learning, and is applicable in many real-world contexts such as medicine, banking, robotics, natural sciences and other industries. Class complexity [2, 3] is a term which encompasses

all the intrinsic traits in data that can hinder the performance of a classifier. There are several categories of metrics that can be used to gauge the complexity of a dataset: feature overlap, linearity, neighborhoods, dimensionality, class balance and network properties. Datasets that present high levels of complexity in some of these categories have been shown to cause poor classifier performance [4].

When it comes to addressing data complexity during a preprocessing [5] phase, several specific methods can be found on the literature, but they usually tackle high dimensionality (feature selection and extraction methods) and class imbalance (resampling methods). Little research has been published on how to address other complexity types during a preprocessing phase and most of it is centered around feature selection [6, 7].

In this work, we present Slicer (supervised linear classifier error reduction), an automatic feature extractor designed with linear class separability in mind. It is based on an autoencoder (AE) model [8], using a special loss function inspired by least-squares support vector machines (LSSVM) [9]. The objective of this model is to learn an alternative representation for each instance where classes are more easily distinguishable. Once trained, the model is able to project any new instance onto the learned feature space without knowing its class. This allows to work with compact representations of the samples instead of the original, high-dimensional ones, in a way that facilitates the work of traditional classifiers, which are usually hindered by high dimensionality [10, 11] unless they specifically select features internally. Extracting features can also help when it is necessary to combine variables from different sources (e.g. images and clinical data), and working with traditional classifiers makes it easier to understand the decision-making process.

The proposed model is applied in a specific use case, aiming both to analyze the level of performance that could be gained and to open new possibilities for combination of imagery and other data, as well as interpretability of classifiers. The chosen application is recognition of COVID-19 in chest X-ray images, using the COVIDGR dataset [12] for this purpose. The fitness of the set of features learned by Slicer is evaluated by means of classification metrics using several standard classifiers and is compared against using a basic AE and learning from the original, unmodified features. The results show a noticeable advantage of the Slicer-generated variables except when using support vector machines as classifier, even though specific traits of this dataset such as "apparently negative" positive samples might affect the learned representation.

The rest of this document is organized as follows. Section VII.2 describes the new feature learner named Slicer. Afterwards, Section VII.3 outlines the main aspects of the experimentation, including the dataset and the evaluation strategy. Section VII.4 discusses the results obtained in the experimentation above and, lastly, Section VII.5 draws some conclusions.

VII.2. Class-informed autoencoder for complexity reduction

This section is dedicated to introducing a new model designed to learn features with improved class separability. The proposed model, Slicer, is based on the minimization of the error of a linear classifier, at the same time that it attempts to maximize its reconstruction abilities. As a result, the learned features are influenced both by the overall information within the data as well as their relation to the class.

Autoencoder fundamentals

An AE is an artificial neural network that is trained to reconstruct the inputs at its output [8]. It includes a certain bottleneck where the representation of the data is somehow restricted: e.g. it is lower dimensional, more sparse or robust against noise. This prevents the AE from simply copying the input instance throughout the network. Instead, an encoder f learns a new representation for the data while a decoder g must be able to recover the original features. This transformation is learned by optimizing the reconstruction error $\mathcal{J}_{\text{RE}}(x, (g \circ f)(x))$ which typically measures a distance between the original samples and their reconstructions, but different penalties and regularizations can allow to influence other aspects of the learned representation.

AEs are usually unsupervised tools, in the sense that they do not receive any information about the labels of the data nor the desired encodings for each instance. Their learning mechanism is, as a result, self-supervised [13]. This means that they can be applied in many contexts where label information is not necessarily available or just partially so: anomaly detection, semantic hashing, data compression, among others [14]. Nonetheless, some AEs do use label information [15], even though the objective is other than learning more separable features. Our objective is to regularize an AE so that it learns from the class labels during training, but does not need them during the prediction phase, and thus facilitates classification tasks.

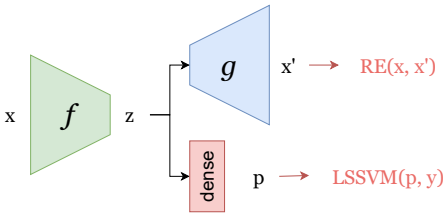

Figure VII.1.: Schematic illustration of the Slicer model

Slicer model: the loss function

Slicer is an AE model regularized by a special penalty function which takes class separability into account. In order to do this, an additional component is introduced to simultaneously fit a LSSVM to the encoded samples as well as evaluate the encoding based on the fitness of said classifier.

Figure VII.1 shows a diagram with the main components of Slicer. f represents the encoder, which is a neural network that transforms the inputs onto encodings. g refers to the decoder, a similar network whose objective is to reconstruct the original data inputs out of the encodings. These encodings are also fed to a single fully connected layer with no activation function, which acts as the support vector machine. Each of the decoder and this last layer are evaluated with their loss functions: the reconstruction error and the LSSVM minimization objective, respectively.

The reconstruction error chosen for the purposes of modeling input samples is cross entropy: a measure of disagreement between two probability distributions, for the case of Bernoulli distributions. It is usually the better option when all values in each instance are in the $[0, 1]$ interval. Its formulation for n instances with k variables is shown in Eq. VII.1.

$$\begin{aligned} \mathcal{J}_{\text{RE}}(x, \theta) = -\frac{1}{nk} \sum_{i=1}^{n} \sum_{j=1}^{k} x_j^{(i)} \log \left[(g \circ f) \left(x^{(i)} \right)_j \right] + \\ \left(1 - x_j^{(i)} \right) \log \left[1 - (g \circ f) \left(x^{(i)} \right)_j \right] . \end{aligned} \tag{VII.1}$$

If the variables are not scaled to the $[0, 1]$, other reconstruction errors like the mean squared error could be used.

For its part, the LSSVM objective assumes that labels are in $\{-1, 1\}$ and is simply a sum of quadratic errors e_i^2 subject to the equality constraint $1 - e_i = y^{(i)} - w^T f \left(x^{(i)} \right) + b$, resulting as formulated in Eq. VII.2.

$$\mathcal{J}_{\text{LSSVM}}(x, y, \theta) = \frac{\mu}{2} w^T w + \frac{\zeta}{2} \sum_{i=1}^{n} \left(y^{(i)} - w^T f \left(x^{(i)} \right) + b \right)^2 . \tag{VII.2}$$

In the previous equation, f usually refers to the kernel used in the model but, in this case, it represents the encoder of the neural network. w holds the weights of the SVM, which are associated to a penalty with coefficient μ. The term that compares classes (y) to the LSSVM output is weighted by ζ.

The resulting loss function for the Slicer model is the sum of both the reconstruction error and the LSSVM loss:

$$\mathcal{J}(x, y, \theta) = \mathcal{J}_{\text{RE}}(x, \theta) + \mathcal{J}_{\text{LSSVM}}(x, y, \theta) \tag{VII.3}$$

VII.3. Experimental framework

The objective of this experimentation is to apply the proposed complexity reduction method in a real world practical case. In particular, we aim to improve the performance of simple classifiers when dealing with a chest X-ray image dataset for COVID-19 classification. This would open several promising research lines, such as the combination of these extracted features with other clinical and laboratory variables or the possibility of using easily interpretable classifiers with the generated features.

COVIDGR dataset

The COVIDGR dataset of X-ray chest images was introduced in [12]. These images were collected under a collaboration with expert radiologists of the Hospital Universitario San Cecilio in Granada, Spain. In total, 852 images were annotated under a strict protocol: positive images correspond to patients who have been tested positive for COVID-19 using RT-PCR within a time span of at most 24h between the X-ray image and the test. Every image was taken using the same type of equipment and always with the posterior-anterior view. It is important that all images are consistent since, otherwise, classifiers could find cues to distinguish COVID-positive samples from negative ones different from the intended aspects of the X-ray that characterize the pneumonia associated to the disease [16]. Figure VII.2 includes one positive example and a negative one.

More information about class distribution is provided in Table VII.1. In the following experiments, the exact same partitions used in [12] are employed, in order to ease comparisons with previous results.

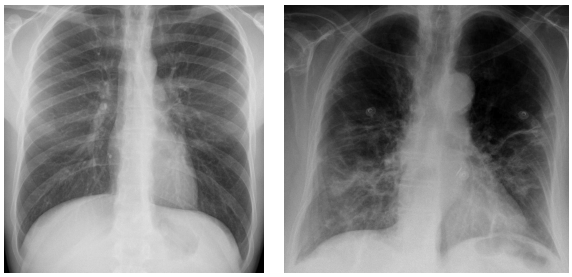

Figure VII.2.: A negative sample (left) and a positive one (right)

Table VII.1.: Class distribution in COVIDGR dataset. Normal-PCR+ refers to X-rays where COVID-19 was not detected by the experts but the patients tested positive.

class	#images	severities	
negative	426		
positive	426	Normal-PCR+	76
		Mild	100
		Moderate	171
		Severe	79

Evaluation strategy

Since the final objective of the proposed model is to improve classification performance, the evaluation framework will consist in a variety of simple classifiers that will be trained with either the original features or the encoded ones. Standard classification metrics will be computed using the predictions over test subsets. A 5-fold cross validation scheme will be applied 5 times for a total of 25 train-test runs, so as to prevent errors from statistical chance. Table VII.2 lists the available feature sets and every classifier and evaluation metric included in the experiment. Each column is independent, in the sense that every feature set has been tested with each one of the classifiers and the performance has always been assessed with all four metrics.

Table VII.2.: Evaluation framework: available feature sets, tested classifiers and evaluation metrics. TP, TN, FP and FN denote true positives, true negatives, false positives and false negatives, respectively.

Feature sets	Classifiers	Evaluation metrics	
Original	Decision tree (DT)	Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Basic AE	k nearest neighbors (kNN)	Precision	$\frac{TP}{TP+FP}$
Slicer	Support vector machine (SVM)	Recall	$\frac{TP}{TP+FN}$
	Gaussian process (GP)	F1-score	$\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Architecture of the Slicer model used with the COVIDGR dataset

The proposed Slicer model has been implemented in the Python language on top of the Tensorflow [17] library. Since the model needs to deal with image data, the specific architecture makes use

of convolutional layers for the encoder and deconvolutional (or transposed convolutional) layers for the decoder.

More specifically, most of the AE is composed of residual blocks such as the ones in the ResNet-V2 architectures [18], as can be seen in Figure VII.3. The left side of this diagram shows the detailed architecture of the AE-based model, with the encoder ranging from the input to the dense layer with 128 units, and the decoder from there to the last deconvolutional layer. The classification component consists in the fully connected (dense) layer that maps the encoding to one variable and is then connected to the LSSVM loss. The overall loss is simply the sum of both error measures, as explained above.

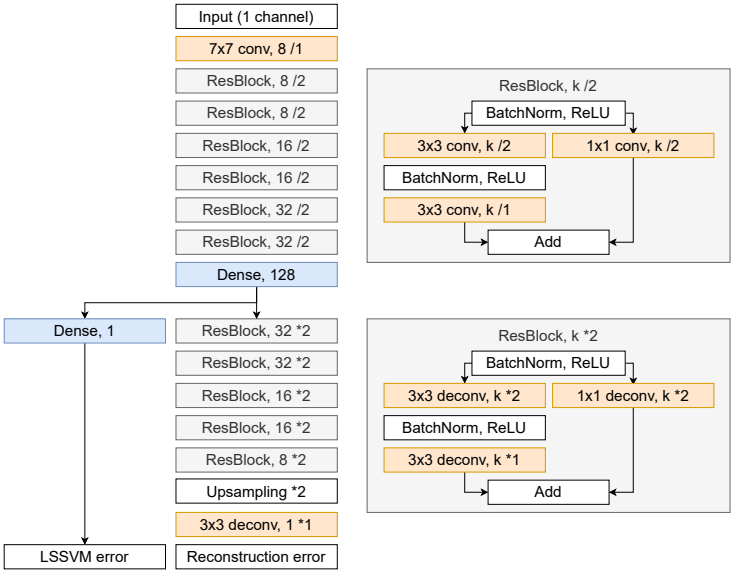

Figure VII.3.: Structure of the Slicer model used with the COVIDGR dataset. Each convolutional and deconvolutional layer is followed by the number of filters as well as an indication of the stride: /2 indicates a stride of 2 for a convolutional layer (the side of the image is halved), and *2 indicates a stride of 2 for a deconvolutional layer (the side of the image is doubled). Each of the residual blocks unrolls like the corresponding diagram on the right.

Using the previous architecture, a model was trained for each training partition of the total of 25 runs, using the parameters detailed in Table VII.3. Images were resized to a common resolution of 512×512 pixels (in total, 262144 variables in the range $[0, 1]$), and the encoding size was of 128 variables, which gives a reduction ratio of 1:2048, or equivalently, it represents the images using just 0.049% of the original variables. An equivalent basic AE was also trained using the same architecture (except for the LSSVM layer and loss) and partitions.

Parameter	Value
μ	0.01
ζ	0.1
Encoding dimension	128
Epochs	50
Batch size	8
Optimizer	Adam
DA: random rotation	$\leq 5°$
DA: horizontal flip	yes
kNN: k	5
DT: maximum depth	10
Image size	512×512

Table VII.3.: Relation of hyperparameters. DA refers to data augmentation techniques (only the techniques shown were applied).

VII.4. Results and discussion

In this section, the main results of the experimentation are analyzed. Table VII.4 contains average metrics for the total of 25 runs that were performed. We can observe drastically different behaviors according to the classifier that was used: the decision tree had similar performance independently of the set of features that were provided, while the Gaussian process only was competitive when using the features learned by Slicer.

Classifier	Features	Accuracy	Precision	Recall	F1-score
DT	original	58.098	58.259	**57.470**	57.665
DT	autoencoder	58.377	**58.889**	55.919	57.196
DT	slicer	**58.593**	58.837	57.422	**57.988**
GP	original	50.024	45.333	1.266	2.452
GP	autoencoder	50.024	4.000	0.047	0.093
GP	slicer	**62.656**	**64.339**	**56.673**	**60.032**
kNN	original	**62.585**	**65.804**	52.780	58.358
kNN	autoencoder	61.837	65.091	51.469	57.194
kNN	slicer	62.326	63.624	**57.284**	**60.092**
SVM	original	**67.329**	66.611	**69.920**	**67.931**
SVM	autoencoder	67.072	**67.006**	67.622	67.057
SVM	slicer	65.987	66.235	65.025	65.393

Table VII.4.: Average classification metrics over 25 runs (5 times 5-fold cross validation). Results from convolutional neural networks are reported in [12].

Several deductions can be made out of the results in Table VII.4. First, the classifiers that take the most advantage from the Slicer-generated features are kNN and GP. In fact, the latter struggles to find an acceptable model of the data using either the original features or the autoencoded ones. For its part, the DT shows little variance with respect to the set of variables it uses, although the Slicer-generated has a slight lead in F1 score and accuracy. The SVM, however, does not benefit from the more separable features and loses performance with respect to the original and autoencoded

ones.

Overall, it is not very surprising that the classifiers that are typically more affected by the quality of features are those which benefit more from the encodings provided by Slicer, whereas classifiers that internally perform their own feature selection or transformations either see small improvements or even decrease their performance.

It is important to note that there exist specific deep learning architectures designed for COVID-19 classification in chest X-rays, such as COVIDNet [19], COVID-CAPS [20] and COVID-SDNet [12]. Comparing the classification performance with these is out of scope for the present work, since we only aim to assess how useful the features extracted by the Slicer model are for traditional classifiers, not to find the best COVID-19 classifier.

As a summary, Table VII.5 displays the average metrics across all 4 classifiers, for each feature set. The average ranking that each one achieved for each metric is also shown. From this, we can conclude that the Slicer model produces feature sets that are consistently superior to a basic AE. Furthermore, it is able to preserve or even improve the quality of the original features, while simultaneously reducing drastically the dimensionality.

Features	Accuracy	Precision	Recall	F1-score
original	59.509 (2.06)	59.002 (1.97)	45.359 (1.92)	46.602 (1.89)
autoencoder	59.327 (2.09)	48.747 (2.16)	43.764 (2.38)	45.385 (2.29)
slicer	62.391 (1.85)	63.259 (1.87)	59.101 (1.71)	60.876 (1.82)

Table VII.5.: Average classification metrics per feature set provided to the classifiers. The average ranking achieved by each feature set in each of the tests is shown in parentheses (lower is better).

VII.5. Conclusions and future work

This work has presented a novel framework for class separability enhancement using an AE-based model with a linear classification component that contributes to the loss function. The model has been implemented as a convolutional AE for the transformation of chest X-ray images onto a more manageable number of variables in order to employ simple classifiers in COVID-19 classification. An exhaustive experimentation has shown that the proposed model improves classification performance over a basic AE with no regularizations and maintains or even improves the performance compared to using the unprocessed data, even though the number of variables is dramatically reduced.

The promising results lead to consider several ways of continuing the work for practical real-world uses:

- Analyze the impact of severity levels on the adequacy of the learned representation. For example, Normal-PCR+ samples appear to look just like negative ones, although they are positive, which could affect the behavior of the model. Removing these images could provide better separation abilities as a consequence.
- Learned features can be combined with other variables that do not come from the chest X-ray, that is, clinical and laboratory data about each patient such as age, gender, comorbidities, etc. Several scores for this kind of data have been proposed but they do not take advantage of the full chest X-ray information [21, 22].
- Several ways to provide meaning to the extracted variables, such as feature disentanglement [23], in combination with transparent classifiers like decision trees, would enable more interpretable pipelines for COVID-19 classification, where users could trace predictions back to the original features.
- Combining the proposed loss function with more advanced AE models such as variational or adversarial AEs could add more potential of improving subsequent classification tasks.

Acknowledgments.

D. Charte is supported by the Spanish Ministry of Science under the FPU National Program (Ref. FPU17/04069). F. Charte is supported by the Spanish Ministry of Science project PID2019-107793GB-I00 / AEI / 10.13039/501100011033. F. Herrera is supported by the Andalusian Excellence project P18-FR-4961. This work is supported by the project COVID19RX-Ayudas Fundación BBVA a Equipos de Investigación Científica SARS-CoV-2 y COVID-19 2020.

References

[1] Charu C. Aggarwal. "Data Classification". In: *Data Mining: The Textbook*. Cham: Springer International Publishing, 2015, pp. 285–344. DOI: 10.1007/978-3-319-14142-8_10 (cit. on p. 1).

[2] José Daniel Pascual-Triana et al. "Revisiting data complexity metrics based on morphology for overlap and imbalance: snapshot, new overlap number of balls metrics and singular problems prospect". In: *Knowledge and Information Systems* (2021), pp. 1–29 (cit. on p. 1).

[3] Mitra Basu and Tin Kam Ho. *Data complexity in pattern recognition*. Springer Science & Business Media, 2006 (cit. on p. 1).

[4] Julián Luengo et al. "Addressing data complexity for imbalanced data sets: analysis of SMOTE-based oversampling and evolutionary undersampling". In: *Soft Computing* 15.10 (2011), pp. 1909–1936 (cit. on p. 2).

[5] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Vol. 72. Springer, 2015 (cit. on p. 2).

[6] Yishi Zhang et al. "Divergence-based feature selection for separate classes". In: *Neurocomputing* 101 (2013), pp. 32–42. DOI: 10.1016/j.neucom.2012.06.036 (cit. on p. 2).

[7] Lei Wang. "Feature selection with kernel class separability". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.9 (2008), pp. 1534–1546 (cit. on p. 2).

[8] David Charte et al. "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines". In: *Information Fusion* 44 (2018), pp. 78–96. DOI: 10.1016/j.inffus.2017.12.007 (cit. on pp. 2, 3).

[9] Johan AK Suykens and Joos Vandewalle. "Least squares support vector machine classifiers". In: *Neural processing letters* 9.3 (1999), pp. 293–300 (cit. on p. 2).

[10] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. "On the surprising behavior of distance metrics in high dimensional space". In: *International conference on database theory*. Springer. 2001, pp. 420–434 (cit. on p. 2).

[11] Kevin Beyer et al. "When is "nearest neighbor" meaningful?" In: *International conference on database theory*. Springer. 1999, pp. 217–235 (cit. on p. 2).

[12] S. Tabik et al. "COVIDGR Dataset and COVID-SDNet Methodology for Predicting COVID-19 Based on Chest X-Ray Images". In: *IEEE Journal of Biomedical and Health Informatics* 24.12 (2020), pp. 3595–3605. DOI: 10.1109/JBHI.2020.3037127 (cit. on pp. 2, 5, 8, 9).

[13] Xiao Liu et al. "Self-supervised learning: Generative or contrastive". In: *arXiv preprint arXiv:2006.08218* 1.2 (2020) (cit. on p. 3).

[14] David Charte et al. "An analysis on the use of autoencoders for representation learning: Fundamentals, learning task case studies, explainability and challenges". In: *Neurocomputing* 404 (2020), pp. 93–107. DOI: 10.1016/j.neucom.2020.04.057 (cit. on p. 3).

[15] Alireza Makhzani et al. "Adversarial autoencoders". In: *arXiv preprint arXiv:1511.05644* (2015) (cit. on p. 3).

[16] Gianluca Maguolo and Loris Nanni. "A critic evaluation of methods for COVID-19 automatic detection from X-ray images". In: *Information Fusion* 76 (2021), pp. 1–7. DOI: 10.1016/j.inffus.2021.04.008 (cit. on p. 5).

[17] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (cit. on p. 6).

[18] Kaiming He et al. "Identity mappings in deep residual networks". In: *European conference on computer vision*. Springer. 2016, pp. 630–645 (cit. on p. 7).

[19] Linda Wang, Zhong Qiu Lin, and Alexander Wong. "Covid-net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest x-ray images". In: *Scientific Reports* 10.1 (2020), pp. 1–12 (cit. on p. 9).

[20] Parnian Afshar et al. "Covid-caps: A capsule network-based framework for identification of COVID-19 cases from x-ray images". In: *Pattern Recognition Letters* 138 (2020), pp. 638–643 (cit. on p. 9).

[21] Jiao Gong et al. "A tool for early prediction of severe coronavirus disease 2019 (COVID-19): a multicenter study using the risk nomogram in Wuhan and Guangdong, China". In: *Clinical infectious diseases* 71.15 (2020), pp. 833–840 (cit. on p. 10).

[22] Stephen R Knight et al. “Risk stratification of patients admitted to hospital with COVID-19 using the ISARIC WHO Clinical Characterisation Protocol: development and validation of the 4C Mortality Score”. In: *bmj* 370 (2020) (cit. on p. 10).

[23] Xiaoming Yu et al. “Multi-mapping image-to-image translation via learning disentanglement”. In: *arXiv preprint arXiv:1909.07877* (2019) (cit. on p. 10).