Supplementary information for

SERS and Siamese Neural Network for Picomolar Identification of Beta-lactam Antibiotics Resistance Gene Fragment in Complex DNA Samples

Part II: Data Processing

A. Skvortsova[a,†], A. Trelin[a,†], P. Kriz[b,c], R. Elashnikov[a], B. Vokata[d], P. Ulbrich[d], A. Pershina[e], V. Svorcik[a], O. Guselnikova*[a,f] and O. Lyutakov*[a]

[a]Department of Solid State Engineering, University of Chemistry and Technology Prague, Technická 5, 166 28, Prague 6, Czech Republic
[b]Department of Mathematics, University of Chemistry and Technology Prague, Technická 5, 166 28, Prague 6, Czech Republic
[c]Charles University, Faculty of Mathematics and Physics, Sokolovská 83, Praha 8, 186 75, Czech Republic
[d]Department of Biochemistry and Microbiology, University of Chemistry and Technology Prague, Technická 5, 166 28, Prague 6, Czech Republic
[e]Siberian State Medical University, 2, Moskovsky Trakt, 634050, Tomsk, Russia
[f]Research School of Chemistry and Applied Biomedical Sciences, Tomsk Polytechnic University, Russia
[†]These authors contribute equally

1 Neural network

Symbol	Meaning	Interpretation
$\mathcal{X}$	Input space	Set of all possible spectra
$\mathbf{x}$	Input datum	Single spectrum
$\mathcal{Z}$	Embedding space	All possible values to which input data is mapped
$\mathbf{z}$	Image of input datum	NN predictions for given spectrum
NN	Neural network	—”—
θ	Model parameters	NN weights

Why convolution? Spectra are a slightly peculiar type of data to process. Typically, information is encoded in the positions of specific peaks, i.e. values of each pixel which makes it similar to the classical multivariate data, perfectly suited for multilayered perceptrons (MLP) or other classifiers such as support vector machines. At the same time, values of neighboring pixels are strongly correlated, akin to the image data, which makes usage of convolutional networks more appropriate. Another point of view on the aptness

*Corresponding authors: guselnio@vscht.cz, lyutakoo@vscht.cz

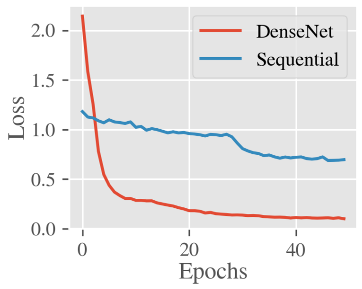


Figure 1: Loss evolution for different connectivities.

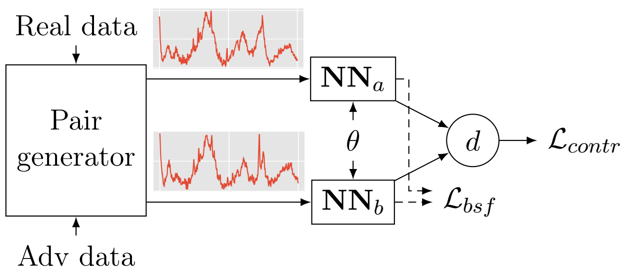


Figure 2: General NN architecture (at training phase)

of convolutional neural networks (CNN) for spectral processing is to think about convolutional layers as of neurons in frequency space that are able to filter, e.g. low-frequency background or high-frequency noise.

Convolutional layers connectivity Original CNN, proposed in 1990 by Le Cun [1] and the majority of its successors used sequential connection of convolutional layers, motivated by the belief that it can "extract features of increasing complexity and abstraction" (quote from [1]). This approach changed after the invention of ResNet [2], which demonstrated the advantages of more complex connectivity schemes. In the state-of-the-art CNNs convolutional layers are linked in an even more sophisticated manner. Examples include Inception [3], MobileNet [4], and DenseNet [5]. The latter was chosen as subjectively the most suitable architecture for spectra recognition due to its compactness and experimentally observed enhanced performance (Figure 1).

Siamese networks The major problem of machine learning algorithms, especially neural networks, is their need for extreme amounts of data. For example, the state-of-the-art image dataset ImageNet [6] contains more than 14 million of images, which is still not enough to build a reliable image recondition system. Clearly, collecting such amount of spectral data is impracticable, thus less data-hungry approaches are required. One of the methods to successfully train NN with significantly smaller datasets are Siamese networks [7]. Siamese network consists of two subnetworks with shared parameters (hence the name). These subnetworks takes a pair of inputs $(\mathbf{x}_a, \mathbf{x}_b)$ and maps them to the corresponding embedding representations $(\mathbf{z}_a, \mathbf{z}_b)$ of significantly lower dimensionality. Distance between representations is measured with some metric, frequently Euclidean, which forms the output of the network, i.e. the loss is formulated as

$$\mathcal{L}\Big(y, d\big(\mathbf{NN}_a(\mathbf{x}_a), \mathbf{NN}_b(\mathbf{x}_b)\big)\Big)$$

where $d(\cdot,\cdot)$ and y represent distance metric and labels respectively. By training on the pairs of same or distinct samples, such networks can learn mapping, such that similar inputs are mapped to close points in embedding space, i.e. it learns some kind of distance metrics and distance-preserving dimensionality reduction.

There are two approaches on how to train such networks; both of them suppose that training data pairs labeled 0 if they belong to the same class and 1 otherwise. In the simplest case, the whole range of distance values is compressed to the $[0, 1]$ range by applying tanh function, i.e. 0 will correspond to the most similar embedding representations and 1 to the most distant. Then loss is defined as a cross-entropy between transformed model output and actual pair label. Trained with that method, the model tries to map data from the same class to the same point, and data from different classes to the points, which are infinitely far from each other.

For practical applications, however, it is not necessary to map data from different classes to infinitely distant points. It is enough to map them into well-separated clusters, for which contrastive loss was developed [8]. This loss, defined as

$$\mathcal{L}_{contr}(y,d) = \begin{cases} \frac{1}{2}\max(d - m_{intra}, 0)^2 & \text{if } y = 0 \\ \frac{1}{2}\max(m_{inter} - d, 0)^2 & \text{if } y = 1 \end{cases}$$

forces network to minimize the distance between points of the same class down to distance m_{intra} and maximize distance between points from different classes, but only up to the distance m_{inter}. Margins m_{inter} and m_{intra} serves as model hyperparameters. NN, trained with this loss learns how to map input data into compact well-defined regions, which can be easily analyzed visually. One can imagine such model as nonlinear trainable analogue of linear discriminant analysis (LDA).

Uncertainty estimation One of the main disadvantages of NNs is the limited ability to access model uncertainty. Bayesian approach would treat a neural network as an abstract probabilistic model parametrized by θ. In that case, we could think of a neural network as of learning probability to observe data $\mathbb{Y}$ inputting $\mathbb{X}$ into model $p(\mathbb{Y}|\mathbb{X}, \theta)$. Unfortunately, Bayesian parameter estimation is rarely possible in practice, since it requires integration over all possible values of all model parameters.

In his work Gal et al. [9] proved that dropout can efficiently approximate the predictive posterior distribution of NN models. In short, the authors demonstrated that NN model with dropout approximates the deep Gaussian process with different kernel functions depending on the model architecture. Moreover, this approach does not require any changes to the NNs trained with dropout except for the fact that dropout should be active not only during model training but during prediction as well (which is called Monte-Carlo dropout in the original article). Thus, with dropout active at the inference phase, it is possible to think of the NN model as of the stochastic process, which maps each input $\mathbf{x}$ to some distribution $p(\mathbf{z}|\mathbf{x})$. Then, the positive and negative examples of the whole training dataset $\mathbb{X}$ will be mapped to the distribution $p(\mathbf{z}|\mathbb{X}, +)$ and $p(\mathbf{z}|\mathbb{X}, -)$, which we will call $p_+(z)$ and $p_-(z)$ for brevity (1D embedding space is used here). Then, by analyzing the similarity of the distribution to which unknown input is mapped with reference distributions $p_+(z)$ and $p_-(z)$ it is possible to conclude about belonging of the specimen to each of the classes.

Novelty detection A significant weakness of almost all machine learning classifiers is its inability to alarm when novel data is presented. In the context of spectra recognition, it is senseless to analyze classifier predictions if significantly different spectrum is given to the network. This problem is usually referred to as novelty detection and is frequently solved by separate autoencoder [10] or specialized algorithms [11, 12]. At the same time, Siamese networks are native novelty detectors, since they will map only similar data to the vicinity of the training data embeddings [13, 14]. Moreover, it is possible to slightly force the NN to map dissimilar data to distant points in the embedding space by providing examples of various dissimilar data to the classifier. In that case we have suggested augmenting contrastive loss with the term for dissimilar data that will ensure slight pushing it away from the distribution of data of interest. Obtained loss, further called triple contrastive loss is given as

$$\mathcal{L}_{contr}(y,d) = \begin{cases} \frac{1}{2}\max(d - m_{intra}, 0)^2 & \text{if } y = 0 \\ \frac{1}{2}\max(m_{inter} - d, 0)^2 & \text{if } y = 1 \\ \sigma(-d) & \text{if } y = 2 \end{cases}$$

where $\sigma(x)$ is the logistic sigmoid function, label 2 indicates pairs with abnormal data. Choice of sigmoid is justified by the fact that it its gradients are significantly smaller than gradients of the quadratic terms loss, thus it can not significantly affect the main goal – separation of two classes.

Adversarial defense Neural networks are vulnerable to so-called adversarial attacks [15], i.e. tiny perturbation of the input data can lead to the contrary prediction results. This vulnerability significantly lowers robustness of the NN and frequently limits their usability in real-life applications. Although numerous method to withstand such attack were developed in recent years, it is still challenging to build truly robust NN [16]. Madry et al. [17] suggested the approach, which is believed to be universal defense against first-order attacks. The basic idea of this method is to directly train NN on adversarial examples, generated with projected gradient descent.

Regions of importance selection Among other properties, spectral data contain regions of importance (ROI), i.e. specific parts of the spectra, crucial for their recognition, common for all spectra in the dataset. Detection of the ROI not only helps to understand the NN prediction and data-generating processes, but also makes NN less vulnerable to perturbations in unimportant regions, since they are excluded from the prediction process. The problem of ROI selection is tightly connected with feature selection process, which usually requires NN to be augmented with external feature selector. This problem can be efficiently handled with Binary Stochastic Filtering [18], a method which adds the feature selection functionality to the NN and was specifically developed for spectra analysis. In short, this method penalizes with L^1 norm feature involvement in the training process, thus minimizing of its loss $\mathcal{L}_{BSF}$, neural network learns how to solve main problem (minimization of contrastive loss) using minimal amount of data.

2 Making optimal decision

Similarity analysis It is extremely important for any machine learning model not only be able to give correct answers but also to reply "do not know" when there is not enough data or the data is contradictory. The fundamental property of the model, developed in current work is its ability to estimate uncertainties besides just assigning class labels to each of the input spectra. Through mapping to the one-dimensional space the model predictions can be understood not only numerically but also visually, which makes it more explainable. Moreover, in addition to uncertainty analysis, the model can detect data abnormalities, resulting from experimental errors or data processing failures. Since SERS data suffers from low reproducibility, it is important to analyze statistically significant number of spectra simultaneously and use robust statistics to suppress the influence of outliers. Firstly, it is necessary to calculate some reasonable statistics that will be further analyzed by the decision-making system. Thus, to analyze the similarity between the distribution for the unknown specimen[1] and reference distributions $p_+(z)$ and $p_-(z)$ following heuristics was

[1]In contrast to the main text, the term *specimen* is used here to describe the entity, being analyzed. This is done to prevent confusion with noun or verb *sample* used in statistical context.

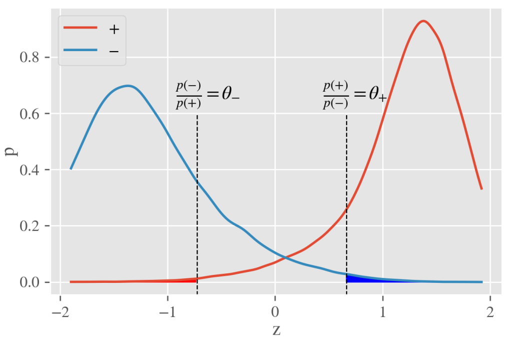


Figure 3: Reference densities and decision thresholds (for $\lambda_{\mathrm{FP}} = 1$, $\lambda_{\mathrm{FN}} = 3$, $\varepsilon = 0.1$) asymmetry of the thresholds is given by the different loss values for FN (red area) and FP (blue area) errors.

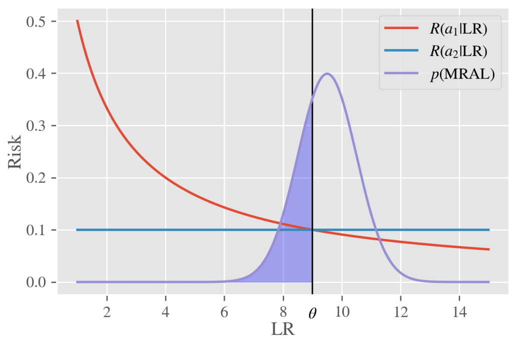


Figure 4: Example of the risks as a function of likelihood ratio, visualization of expected potential risk drop ΔR.

proposed:

1. Measure r spectra of unknown specimen $\mathbf{x}_1 \dots \mathbf{x}_r$, sample data from $p(z|\mathbf{x}_i)$ (using Monte-Carlo dropout according to [9]), use Gaussian kernel density estimation to get integrable estimate of $p(z|\mathbf{x}_i)$.

2. Calculate medians of $p(z|\mathbf{x}_i)$, $p_+(z)$, and $p_-(z)$, denoted here as $\tilde{\mu}_x$, $\tilde{\mu}_+$, $\tilde{\mu}_-$ respectively and interquartile ranges IQR_+ and IQR_-. If $\tilde{\mu}_x > \max\left\{\tilde{\mu}_+ + \mathrm{IQR}_+, \tilde{\mu}_- + \mathrm{IQR}_-\right\}$ or $\tilde{\mu}_x < \min\left\{\tilde{\mu}_+ - \mathrm{IQR}_+, \tilde{\mu}_- - \mathrm{IQR}_-\right\}$, i.e. if input spectrum is mapped significantly far from reference distributions, reject it as abnormal.

3. Compute average likelihoods of $p(z|\mathbf{x}_i)$ with respect to the reference distributions $S_{i,+} = \int p(z|\mathbf{x}_i)p_+(z)dz$ and $S_{i,-} = \int p(z|\mathbf{x}_i)p_-(z)dz$. Since $p_+(z)$ can be understood as the likelihood of a single point z assuming class $+$, but our evidence "z" is uncertain (given by density $p(z|\mathbf{x}_i)$), we may adopt the principle of Jeffrey conditioning [19] and calculate the average likelihood as the quantity $S_{i,+} = \int p_+(z)p(z|\mathbf{x}_i)dz$. Similarly, $S_{i,-} = \int p_-(z)p(z|x_i)dz$ represents the average likelihood for class $-$ corresponding to x_i. Get the ratio of average likelihoods S for ith spectrum as $\mathrm{RAL}_i = \frac{S_{i,+}}{S_{i,-}}$.

4. To robustly estimate location use bootstrapping. Generate random sample (with replacement) of the length r from the $\mathrm{RAL}_1 \dots \mathrm{RAL}_r$, and calculate truncated mean of it. Repeat this procedure $N_{bootstrap}$ times. Obtained distribution of mean ratios of average likelihood (MRAL) is then treated as an estimate of the likelihood ratio, inferred from multiple spectra (multiple observations of the single specimen).

This heuristic has two important properties. Firstly, the MRAL distribution concentrates with growing number of spectra which defines the margin for number of spectra to collect from single specimen. Secondly, it provides natural way to combine evidences from multiple specimens to strengthen the conclusions. The odds form of Bayes' theorem

$$\frac{p(c_+|\mathbf{x})}{p(c_-|\mathbf{x})} = \frac{p(\mathbf{x}|c_+)}{p(\mathbf{x}|c_-)} \frac{P(c_+)}{P(c_-)}$$

states that posterior odds $\frac{P(c_+|\mathbf{x})}{P(c_-|\mathbf{x})}$ can be computed as a product of prior odds $\frac{P(c_+)}{P(c_-)}$ with likelihood ratio. Since Bayes rule allows multiple updates of the prior beliefs, we can substitute priors with updated odds, getting

$$\frac{p(c_+|\mathbf{x}_2, \mathbf{x}_1)}{p(c_-|\mathbf{x}_2, \mathbf{x}_1)} = \frac{p(\mathbf{x}_2|c_+)}{p(\mathbf{x}_2|c_-)} \left[\frac{p(\mathbf{x}_1|c_+)}{p(\mathbf{x}_1|c_-)} \frac{P(c_+)}{P(c_-)}\right]$$

Thus, combining evidences from multiple specimens is equivalent to updating priors with product of likelihood ratios. Even if each single specimen only weakly evidences for one of the classes, obtained combination will be a strong evidence for that class.

Since we work with distributions of MRAL instead of single value, desired product distribution can be obtained by multiplying individual samples of bootstrapped MRAL estimated for each specimen. As an illustration, to combine evidences from two specimens a, b it is necessary get bootstrapped samples MRAL_a, MRAL_b for each of them as described above and calculate pointwise product $\mathrm{MRAL}^i_{ab} = \mathrm{MRAL}^i_a \cdot \mathrm{MRAL}^i_b$.

Bayes optimal classifier Bayesian decision theory states that optimal strategy is the strategy that assigns action a_i for observed data $\mathbf{x}$ that minimizes conditional risk $R(a_i|\mathbf{x}) = \sum_j \lambda_{ij} p(c_j|\mathbf{x})$ [20]. Matrix Λ defines losses for every possible combination of the true state of the nature (class c_j) and action a_i. In our case there are two possible states of nature: c_+ (positive, DNA of interest is present in the specimen) and c_- (negative, there are no such DNA in the specimen) as well as three possible actions a_1 — classify as positive, a_2 — classify as unrecognized, and a_3 — classify as negative. Then loss matrix can be defined as:

$$\Lambda = \begin{array}{c|cc} & c_+ & c_- \\ \hline a_1 & 0 & \lambda_{\mathrm{FP}} \\ a_2 & \varepsilon & \varepsilon \\ a_3 & \lambda_{\mathrm{FN}} & 0 \end{array}$$

where correct classifications correspond to zero loss, false positive and false negative to losses λ_{FP} and λ_{FN} respectively and answering of “do not know” results in loss ε.

To derive optimal strategy it is firstly needed to derive risk functions for each three cases. Starting from definition of the conditional risk

$$\begin{aligned} R(a_1|\mathbf{x}) &= \lambda_{\mathrm{FP}} p(c_-|\mathbf{x}) \\ R(a_2|\mathbf{x}) &= \varepsilon p(c_+|\mathbf{x}) + \varepsilon p(c_-|\mathbf{x}) = \varepsilon \\ R(a_3|\mathbf{x}) &= \lambda_{\mathrm{FN}} p(c_+|\mathbf{x}) \end{aligned}$$

we can solve for the risk as a function of likelihood ratio

$$\begin{aligned} R(a_1|\mathrm{LR}_+) &= \frac{\lambda_{\mathrm{FP}}}{1 + \mathrm{LR}_+[P(c_+)/P(c_-)]} \\ R(a_2|\mathrm{LR}_+) &= R(a_2|\mathrm{LR}_-) = \varepsilon \\ R(a_3|\mathrm{LR}_-) &= \frac{\lambda_{\mathrm{FN}}}{1 + \mathrm{LR}_-[P(c_-)/P(c_+)]} \end{aligned}$$

where $\mathrm{LR}_+ = p(x|c_+)/p(x|c_-)$, $\mathrm{LR}_- = p(x|c_-)/p(x|c_+)$, $P(c_+)$ and $P(c_-)$ are prior probabilities of corresponding classes. Since $R(a_1|\mathrm{LR}_+)$ and $R(a_3|\mathrm{LR}_-)$ are monotonic decreasing in terms of corresponding LR, the choice of optimal decision can be reduced to two comparisons:

$$\arg\min R(a_i|LR) = \begin{cases} a_1 & \text{if } \mathrm{LR}_+ > \theta_+ \\ a_3 & \text{if } \mathrm{LR}_- > \theta_- \\ a_2 & \text{otherwise} \end{cases}$$

```
Data: distribution of MRAL
Result: optimal decision
if $\theta \notin (\mathrm{MRAL}_L, \mathrm{MRAL}_U)$ then
    return MinimalRiskAction(MRAL)
end
$d \leftarrow$ GetRelativeRiskDrop(MRAL)
if $d <$ CriticalRiskDrop then
    return MinimalRiskAction(MRAL)
else
    if MRALConverged() then
        warn result is not reliable
        return MinimalRiskAction(MRAL)
    else
        ask for more spectra
    end
end

function GetRelativeRiskDrop(MRAL):
    a ← MinimalRiskAction(MRAL)
    if a is "do not know" then
        $\Delta\mathrm{R} \leftarrow \int_0^\theta [R(a_1|\mathrm{LR}) - R(a_2|\mathrm{LR})] p(\mathrm{MRAL}) d\mathrm{LR}$
        $\mathbb{E}\mathrm{R} \leftarrow \int_0^\infty R(a_2|\mathrm{LR}) p(\mathrm{MRAL}) d\mathrm{LR}$
    else
        $\Delta\mathrm{R} \leftarrow \int_\theta^\infty [R(a_2|\mathrm{LR}) - R(a_1|\mathrm{LR})] p(\mathrm{MRAL}) d\mathrm{LR}$
        $\mathbb{E}\mathrm{R} \leftarrow \int_0^\infty R(a_1|\mathrm{LR}) p(\mathrm{MRAL}) d\mathrm{LR}$
    end
    return $\Delta\mathrm{R}/\mathbb{E}\mathrm{R}$
```

Algorithm 1: Decision algorithm

where thresholds can be found as roots of equations

$$R(a_1|\theta_+) = R(a_2|\theta_+),\ \theta_+ = \frac{P(c_-)}{P(c_+)} \frac{\lambda_{\mathrm{FP}} - \varepsilon}{\varepsilon}$$
$$R(a_3|\theta_-) = R(a_2|\theta_-),\ \theta_- = \frac{P(c_+)}{P(c_-)} \frac{\lambda_{\mathrm{FN}} - \varepsilon}{\varepsilon}$$

Two conditions $\mathrm{LR}_+ > \theta_+$ and $\mathrm{LR}_- > \theta_-$ can not be satisfied simultaneously when $\theta_+\theta_- > 1$ since LR_+ is reciprocal to LR_-. This condition is ensured when $\varepsilon < \lambda_{\mathrm{FP}}\lambda_{\mathrm{FN}}/(\lambda_{\mathrm{FP}}+\lambda_{\mathrm{FN}})$, which is assumed further. When $\varepsilon \geq \lambda_{\mathrm{FP}}\lambda_{\mathrm{FN}}/(\lambda_{\mathrm{FP}} + \lambda_{\mathrm{FN}})$ action a_2 becomes impossible, i.e. classifier loses ability to answer “do not know.

If likelihood ratios were known exactly, the simple procedure of two comparisons would provide the optimal strategy. In our case, however, only estimated distributions MRAL_+ and MRAL_- are available. We could solve that problem by calculating expected risks, i.e. $\mathbb{E}(R(a_i|LR) = \int R(a_i|\mathrm{LR})p(\mathrm{MRAL})d\mathrm{LR}$ and chose the action that minimizes expected risk [21]. However, that approach does not take into account the variance of MRAL distribution, and thus lacks the ability to determine the number of spectra needed to collect from a single specimen, which is highly useful in practice. Therefore, the decision procedure was extended with analysis of expected relative risk drop. The obtained Algorithm 1, can be summarized as[2]:

1. If the majority of the MRAL density lies above or below threshold θ_+, decide based on θ_+.

2. Otherwise, analyze potential risk drop, i.e. expected difference in risks if exact value of LR was known. It can be calculated as integral of risk difference between current optimal and alternative decisions, weighted by the MRAL density over the region of optimality of alternative decision. For the example, shown in the Figure 4 current

[2]Since comparisons differ only in the values of θ and LR we will further focus only on deciding between a_1 and a_2. The procedure of deciding between a_3 and a_2 can be obtained by swapping subscripts 1 to 3 and replacing subscript + with −.

optimal decision is a_1, so potential risk drop is

$$\Delta R = \int_0^{\theta} [R(a_1|\mathrm{LR}) - R(a_2|\mathrm{LR})]p(\mathrm{MRAL})d\mathrm{LR}.$$

If ΔR is negligible compared to the current risk, accept current optimal decision. Although additional spectrum could potentially reduce variability of MRAL and change the decision, the corresponding improvement in risk would be negligible. Hence, there is no reason to add another spectrum.

3. If ΔR is significant analyze convergence of the ΔR, i.e. how strongly it changes after addition of one more spectrum. If ΔR converged, i.e. if the distribution of MRAL_+ is very close to θ_+ and addition of new spectra does not change it significantly than no certain decision can be made. Then warn about the unreliability of results and return current optimal decision. If ΔR has not converged yet, ask for more spectra of the same specimen.

Another question is how to choose the loss values for the decision system. Firstly, the decisions are independent on the scale of the loss values, so one of the losses λ_{FP}, λ_{FN} can be set to 1 and another is set to the relative price of that error. Choosing optimal value of the ε is slightly more nontrivial since it is difficult to quantitatively determine relative error of the decision "do not know". In that case it is possible to calculate probabilities of type I and II errors from reference distributions and set ε to the value that ensures the desired accuracy level. For example, probability of false negative error can be calculated as:

$$P(a_3|c_+) = \frac{\int_{I_3} p_+(z)dz\, P(c_+)}{\int_{I_3} p_+(z)dz + \int_{I_3} p_-(z)dz}$$

where I_3 is region of optimality of action a_3 (e.g. $(-\infty, \theta_-)$ on the Figure 3).

Implementation of the described approach can be found in the repository[3]

3 Data processing: practice

All the models described below were implemented in the TensorFlow 2 framework [22] using Keras [23].

Preprocessing Raw spectra were preprocessed by subtracting background with asymmetric least squares smoothing algorithm according to [24]. Cosmic spikes were removed from the spectra using a gradient magnitude criterion. Data was normalized to $(0, 1)$ range. No other preprocessing, such as digital filtering or manual corrections were performed.

NN architecture and training The model, used for data processing can be defined as a convolutional DenseNet-like Siamese neural network. Its architecture is described in detail in the Table 1 and visualized in the Figure 2. Euclidean distance was used in the last layer of the model. Triple contrastive loss was minimized during training.

The model was augmented with BSFilter layer, placed between convolutional and fully-connected counterparts. The weights of the BSFilter layer were shared along the channel axis, so it performed selection of importance regions, common along all channels, i.e. ROI of the input spectrum. The network was trained in an adversarial manner according

[3] `https://github.com/Trel725/Decider`

Layer	Parameters
DenseNet block	kernel size=4, strides=4
DenseNet block	kernel size=8, strides=1
DenseNet block	kernel size=8, strides=1
BSFilter	$\gamma = 1 \times 10^{-6}$
Flatten	—
Dense + Dropout	n = 100, LeakyReLU
Dense + Dropout	n = 50, LeakyReLU
Dense	1, linear

(a) Main model

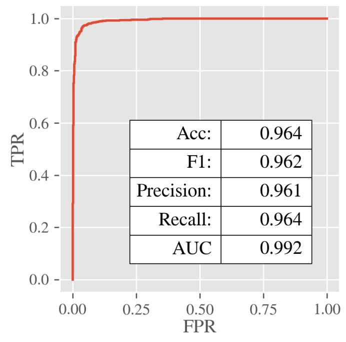
Layer	Parameters
Conv1D	kernel size, strides
BatchNormalization	—
LeakyReLU	$\alpha = 0.3$

(b) DenseNet block

$$\begin{tikzcd}
\mathbf{X} \arrow[r] & \boxed{\text{DN}} \arrow[r] \arrow[rr, bend left] \arrow[rrrr, bend left] & \boxed{\text{DN}} \arrow[r] \arrow[rrr, bend left] & \oplus \arrow[r] & \boxed{\text{DN}} \arrow[r] & \oplus \arrow[r] & {}
\end{tikzcd}$$

(c) Connectivity of DenseNet blocks. Symbol $\oplus$ denotes concatenation along channels axis.

Table 1: Structure the convolutional networks (both are identical in Siamese architecture). Each Dense layer is additionally regularized with L^2 penalty ($\lambda = 1 \times 10^{-6}$) and followed by Dropout layer with rate 0.15 (not shown in the table for brevity). For detailed description of each layer, please refer to the Keras documentation.



Acc:	0.964
F1:	0.962
Precision:	0.961
Recall:	0.964
AUC	0.992

Figure 5: Validation classification metrics

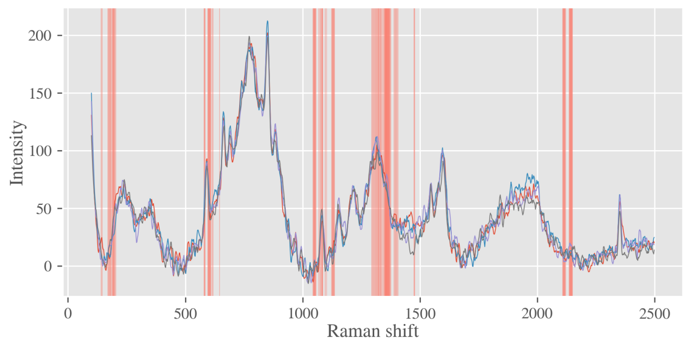


Figure 6: Examples of Raman spectra with regions of importance, selected by BSF

to [17], adversarial examples were generated using 5 steps of projected gradient descent with learning rate 3×10^{-3}, normal and adversarial data were mixed during training. In addition to the training dataset, consisting of $\approx$12500 observations, approximately 4000 various Raman spectra from different projects were used to provide the network examples of abnormal data. The training was continued until manually observed convergence, which took 50 epochs in total, training and validation losses converged to 0.1299 and 0.1291 respectively. The model achieved accuracy 96% on training data (datum $\mathbf{x}$ was assumed to be correctly recognized if $p_+\big(\mathbf{NN}(\mathbf{x})\big) > p_-\big(\mathbf{NN}(\mathbf{x})\big)$ for positive $\mathbf{x}$ and wise versa). Summary of classification metrics for validation data is shown in the Figure 5.

After training, regions of importance were visualized by calculating gradients of CNN output w.r.t. its input for all examples from the training dataset and taking the absolute value of it (Figure 6). Since BSFilter layer efficiently zeros the features in unimportant regions, the gradient is a meaningful way to visualize selected ROIs. Alternatively, it is possible to visualize regions of importance by simple plotting of BSF weights and broadening them because to a single neuron in the output of the convolutional counterpart correspond few near input pixels (known as a field of vision of that neuron).

Prediction The siamese architecture was needed only for model training. For inference, one the identical networks (shown as $\mathbf{NN}_a$ and $\mathbf{NN}_b$ in Figure 2) was used to map the input data to the corresponding distribution. Distribution similarity analysis was performed as described above using kernel density estimation algorithm implemented in [25]. Testing of NN ability to recognize spectra with different concentrations of target DNA was performed using dataset of spectra, collected from additional separately prepared specimens. A set of spectra for each of the testing specimens was analyzed simultaneously and corresponding scores are tabulated in the main article.

References

[1] Yann LeCun, et al., “Handwritten digit recognition with a back-propagation network”, in *Advances in neural information processing systems*, 1990, pp. 396–404.

[2] Kaiming He, et al., “Deep residual learning for image recognition”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] Christian Szegedy, et al., “Rethinking the inception architecture for computer vision”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[4] Andrew G Howard, et al., “Mobilenets: Efficient convolutional neural networks for mobile vision applications”, *arXiv preprint arXiv:170404861*, 2017.

[5] Gao Huang, et al., “Densely connected convolutional networks”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[6] Jia Deng, et al., “Imagenet: A large-scale hierarchical image database”, in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[7] Jane Bromley, et al., “Signature verification using a" siamese" time delay neural network”, in *Advances in neural information processing systems*, 1994, pp. 737–744.

[8] Raia Hadsell, et al., “Dimensionality reduction by learning an invariant mapping”, in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, IEEE, 2006, vol. 2, pp. 1735–1742.

[9] Yarin Gal and Zoubin Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”, in *international conference on machine learning*, 2016, pp. 1050–1059.

[10] Nathalie Japkowicz, et al., “A novelty detection approach to classification”, in *IJCAI*, Citeseer, 1995, vol. 1, pp. 518–523.

[11] Bernhard Schölkopf, et al., “Estimating the support of a high-dimensional distribution”, *Neural computation*, 13(7), 2001, pp. 1443–1471.

[12] Heiko Hoffmann, “Kernel PCA for novelty detection”, *Pattern recognition*, 40(3), 2007, pp. 863–874.

[13] Chen Zhang, et al., “Small molecule accurate recognition technology (smart) to enhance natural products research”, *Scientific reports*, 7(1), 2017, pp. 1–17.

[14] Marc Masana, et al., “Metric learning for novelty and anomaly detection”, *arXiv preprint arXiv:180805492*, 2018.

[15] Christian Szegedy, et al., “Intriguing properties of neural networks”, *arXiv preprint arXiv:13126199*, 2013.

[16] Anish Athalye, et al., “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples”, *arXiv preprint arXiv:180200420*, 2018.

[17] Aleksander Madry, et al., “Towards deep learning models resistant to adversarial attacks”, *arXiv preprint arXiv:170606083*, 2017.

[18] Andrii Trelin and Aleš Procházka, “Binary Stochastic Filtering: feature selection and beyond”, *arXiv preprint arXiv:200703920*, 2020.

[19] Richard Jeffrey, “Alias Smith and Jones: The testimony of the senses”, *Erkenntnis*, 26(3), 1987, pp. 391–399.

[20] Richard O Duda, et al., *Pattern classification*, John Wiley & Sons, 2012.

[21] Giri Gopalan, “Admissibility of a posterior predictive decision rule”, *arXiv preprint arXiv:150706350*, 2015.

[22] Martín Abadi, et al., “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems”, , 2015, URL `https://www.tensorflow.org/`, software available from tensorflow.org.

[23] François Chollet et al., “keras”, , 2015.

[24] Paul HC Eilers and Hans FM Boelens, “Baseline correction with asymmetric least squares smoothing”, *Leiden University Medical Centre Report*, 1(1), 2005, p. 5.

[25] Pauli Virtanen, et al., “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”, *Nature Methods*, 17, 2020, pp. 261–272.