Sharing and Automation for
Privacy Preserving Attack Neutralization

(H2020 833418)

D5.10 Demonstrator for visualisation support for distributed and federated learning (M30)

Published by the SAPPAN Consortium

Dissemination Level: Public

H2020-SU-ICT-2018-2020 – Cybersecurity

Document control page

Document file: Deliverable D5.10
Document version: 1.0
Document owner: Franziska Becker (USTUTT)

Work package: WP5
Task: T5.5 Demonstrator for visualisation support for distributed and federated learning
Deliverable type: Demonstrator
Delivery month: M30
Document status: ☒ approved by the document owner for internal review
☒ approved for submission to the EC

Document History:

Version	Author(s)	Date	Summary of changes made
0.1	Franziska Becker (USTUTT)	2021-09-28	Initial outline
0.2	Franziska Becker (USTUTT)	2021-10-15	Reworked outline and basic vis descriptions
0.3	Franziska Becker (USTUTT)	2021-10-25	Review ready version
1.0	Franziska Becker (USTUTT)	2021-10-29	Incorporated review comments

Internal review history:

Reviewed by	Date	Summary of comments
Lasse Nitz (FIT)	2021-10-25	Some suggestions in regard to wording/grammar/spelling, figure references (incorrect figure numbers in the text), and formatting (formalisations, citations).
Martin Zadnik (CESNET)	2021-10-26	Technical review.

Executive Summary

This is one of two deliverables for task T5.5 "*Demonstrator for visualisation support for distributed and federated learn*ing". It documents the efforts to design and implement a prototype that supports one of the federated learning scenarios developed in task T5.3 *"Federated learning of a global model without sharing local models"*. In particular, this deliverable focuses on federation after each model epoch and how such a scenario may be visualised. We discuss the goals and challenges faced during the design and implementation of a visual analytics system for this specific scenario and describe the visualisation components included in our prototype. In addition, we report on several discarded designs and why these may be insufficient to achieve specific goals. We assess the utility of our approach, where it shines, where it suffers and where future work should direct its focus.

Contents

1 Introduction

This deliverable documents our efforts undertaken to support federated learning scenarios with visualisation in task T5.5 *"Demonstrator for visualisation support for distributed and federated learning"*, as part of work package five. First, we illustrate the context of this work in SAPPAN. Subsequently, we discuss the different federated learning scenarios and why some are better candidates for visualisation than others. Then we describe the design and implementation of a visual analytics prototype we developed to address task T5.5. Here, we consider the goals such a system should aim at and which challenges are encountered in that endeavour. Then, we discuss our designs and how they relate to our goals for a federated learning visualisation. Lastly, we assess the utility of our approach, where it performs well, where it faces problems and which aspects should be considered for future work.

2 SAPPAN Context

The results of this deliverable intersect many components of the SAPPAN scheme, the latter being shown in Figure 1. The federated learning scenarios are part of sharing and the global response (as indicated by the red rectangle in Figure 1), so in that sense this deliverable is also part of those components. However, it also functions as a local assessment component, intended to be used by machine learning practitioners at an organization to test the federated model before it is incorporated into the detection pipeline. In addition, developers of federated learning approaches, whether in research or in industry, may use such a system to analyse the results of their implementation.

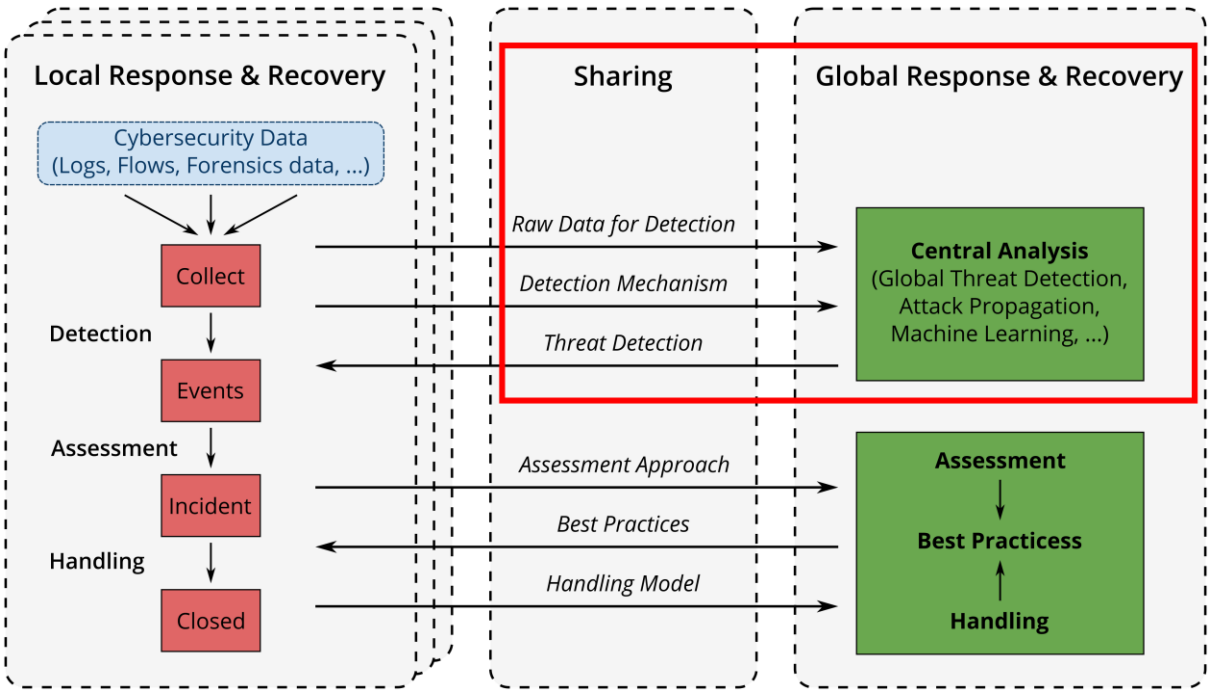

Figure 1: SAPPAN scheme showing different parts for detection, response and sharing. The red rectangle indicates where federated learning is situated.

3 Distributed and Federated Learning Scenarios

The SAPPAN project investigates several distributed and federated learning scenarios in work package five. In task T5.1 *"Distributed Learning of a global model based on shared anonymised data"*, learning and data are distributed in the sense that either anonymized data, extracted features or feature detectors are shared across organisations to construct a better classifier. For the DGA (domain generation algorithm) detection use case, examples include the sharing of anonymized domain names, sharing the extracted features of a feature based-classifier like a decision tree or sharing the first few layers of a deep learning classifier, which function like feature extractors but do not reveal their inner semantics as easily as feature-based classifiers do.

Task T5.2 *"Federated learning of a global model based on shared locally trained models"* analyses federated learning in the context of shared local models. Such sharing may be achieved by performing ensemble classification. Ensemble classification denotes the process of combining two or more classifiers in the hope of achieving a better performance than with just the individual classifiers. A prime example of an ensemble model is a random forest, which combines several decision trees for classification. In the case of DGA detection, an ensemble classifier can be constructed either by averaging the local models' confidence scores or by deciding with a majority vote. For the use case of application profiling, other ways to build a global model from shared local models involve the sharing of rule sets for rule-based approaches and the sharing of Petri nets for process-mining approaches.

Finally, task T5.3 *"Federated learning of a global model without sharing local models"* considers federated learning scenarios. Federation, given a deep learning model, may occur after model convergence or after each training epoch. Since we chose this task as the basis for our visualisations, we describe the different federated learning cases in further detail in the next paragraphs.

In the first case, a randomly initialised or a pre-trained neural network classifier is distributed among all participating parties. Each party then trains this model using their own private data, which they do not wish to share. After convergence, i.e., when the performance on the validation data no longer increases or the maximum number of epochs has been reached, a weight update compared to the initial global model is computed. This weight update is shared by each party and then averaged and applied to the initial model, to create the final global model. The second case is largely the same as the first one. However, instead of computing the weight update after convergence, it is computed after each training epoch. These weight updates are also shared, averaged and then the result is redistributed so that each party can continue training with their private data.

Another scenario is a teacher-student approach. This approach trains a global model by taking a dataset for training, querying the teacher models and then combining their predictions in some fashion. When using soft labels, i.e., probabilities or confidence scores, the final label of a training instance is equal to the average of the teacher models' confidence scores. When using hard labels, i.e., discrete labels, the final label is determined by a majority vote, which requires a tiebreaker in case there is an even number of teacher models.

We believe it is infeasible to try to create visualisations for all of these scenarios, especially in relation to how advanced the implementation and analysis of each scenario was at the start of working on this task. Consequently, we focus this task on the visualisation of the federated learning scenario from task T5.3. We chose this scenario

since it most closely resembles federated learning, was one of the most advanced scenarios at the end of month 21 and exhibited overall good performance. In particular, we are interested in investigating the case where federation is performed by sharing model weights after each epoch. Having data not only after convergence, but also after each update may provide more meaningful information to better understand how the learning process affects the global model and why the global model came to perform as it does. In addition, federation after model epoch seems to perform better than federation after model convergence. For more information regarding the different federated and distributed learning scenarios, please refer to deliverables D5.2, D5.4 and D5.6.

4 Visualisation Objectives and Challenges

This section discusses the objectives and challenges that present themselves when attempting to design a visualisation system that supports federated learning.

4.1 Visualisation Requirements

How to design a visual analytics system heavily depends on the particular goals it should accomplish and who the intended user audience is. For the previously outlined case of federated learning, we have identified the following questions as potentially relevant:

1. How do the different parties perform during the training process?
2. Are there specific instances or groups of instances for which the predictions vary significantly between parties?
3. For which reasons do the parties perform differently for specific instances or groups of instances?
4. How do the weights change over the training process?
5. How do weight changes correlate with performance?

The first question concerns performance and requires a visualisation that can show temporal data for a number of data sources, in our case different parties. The most straightforward option, familiar to both users and developers of machine learning models, is the visualisation as a line chart. This allows the user to see different performance measures at varying time points of the training process, e.g., model accuracy, false-positive rate or false-negative rate. While such visualisations can only provide an overview of the models' performance, they present an intuitive starting point for a deeper analysis.

The second question also concerns performance, but aims to understand the differences that may exist between parties in relation to subsets of the data. To test their models, developers often have specific instances or classes which they know can provide interesting insight into the model's performance and inner representations of the data (as reported by [1]). A visualisation to answer this type of question must show performance on a more detailed level than for the previous question, showing it either on the instance or on the group level.

Question 3 is an extension of the second question, going from "Are there differences in performance?" to "*Why* are there differences in performance?" This requires not only that a visualisation shows differences, but lets the user investigate potential causes for these differences. This may be achieved by giving more insight into the properties of

the data or by giving more insight into the *reaction* to the data, e.g., via attribution [2] or other explainability techniques.

The fourth and fifth questions consider how the weights change during the training process and which effects these changes may have on performance. How to visualise weights in deep learning models has been explored in different ways, with many employing pixel-based visualisations to directly show the weight values. While this may work for a single model, it is harder to use this approach efficiently for a larger number of models that are displayed simultaneously. A visualisation for weight changes in the federated learning scenario must enable visual comparison between the models as well as visual comparison between weight changes and performance.

4.2 Challenges

There are three main challenges that complicate the design and implementation of a visual analytics system that can fulfil the previously discussed objectives

- Large amounts of data
- Large(r) number of models
- High-dimensional temporal data

Handling large amounts of data is always challenging, especially in the area of visualisation where many visual encodings do not scale well to an arbitrary number of data points. In addition, the types of available interactions may be limited by the amount of data, if some kind of computation has to be performed for the interaction. For our scenario, individual datasets from the different parties that take part in the federated learning process range from a few hundred thousand to millions. Showing all datasets in an interactive manner is infeasible, so we focus on visualising smaller subsets of the data. Then, we provide the user with interaction mechanisms to analyse this data more thoroughly. While this cannot give a clear picture of the complete data involved in the training of the models, it aligns with how Hohman et al. [1] report that model developers often go about assessing their models: by looking at particular groups of instances (subset-based analysis) or by looking at specific instances (instance-based analysis) which they know to be helpful for the analysis.

Although the number of models given for our use case is not large in the same sense that the data is large, their number can still provide a challenge for the system, since it has to scale from 2 to 5 parties and from 5 to 25 training epochs. Given we have five parties and train all models for ten epochs, we already have 50 distinct models to visualise. This necessitates that a visualisation can handle this number of data sources or that it must allow the user to navigate them in a meaningful fashion.

Finally, the weights in particular prove to be a challenge for many visualisations of deep learning model. The nodes in the different layers in a model are connected via weights, which means that even for smaller models the number of weights can be quite high. Thus, visualising model weights always comes with challenges regarding scale that need to be addressed.

4.3 Related Work

This section briefly discusses some related work from the area of visualisation. To our knowledge, no visualisation with the explicit goal of exploring federated learning sce-

narios has been published to this date. However, the goals we presented for this scenario also apply to model debugging and validation using visualisation. There are a number of systems developed for this purpose, some of which we discuss here.

Some systems like Squares [3] or DeepCompare [4] only consider the scenario of comparing two classifiers simultaneously. This is often due to the design of the different incorporated visualisations, which do not scale sufficiently for a larger number of models. This makes their designs impractical for our purposes. In contrast, other systems such as ClaVis [5] or Boxer [6] are built to compare a larger number of classifiers. For ClaVis, we can see the inclusion of well-known performance metrics as a key component to facilitate comparison. Contrary to our scenario, the focus of ClaVis seems to be the exploration of the parameter space for a model and the comparison of models across different architectures. This results in a different focus, where users will try to find the best hyper-parameters and architectures instead of trying to understand why particular predictions are different between models and how exactly prediction behaviour changes over time. Boxer more closely aligns with our goals of letting the user get a clearer understanding of model behaviour. They have designed a modular system that uses basic visualisations, often related to performance metrics in some fashion that uses set-algebra to let users explore interesting subset of the training and validation data. However, the utility of their approach may suffer in scenarios such as ours, where we do not have data features or a larger number of classes available in order to find interesting subsets using the visualisation.

Manifold [7] compares models in pairs, showing true positives, false positives, true negatives and false negatives and how each pair of models agrees or disagrees in that respect, for each class of the dataset. In addition, they perform feature attribution to show how features lead to differences between data subsets. Like Boxer, these designs may not translate well for our use case where no features are given and only two classes are present in the dataset. In addition, like Boxer, Manifold only performs analysis on the final classifier, i.e., support for analysing temporal data is not included in the design.

Another relevant aspect is the visualisation of weights in deep learning models. Often, weights are visualised as a kind of heatmap with a sequential colour scale that indicates weight values (e.g., [8, 4, 9]). In a similar manner, the openly available Tensorflow Playground[1] shows weights as a dashed line between layers, where the weight of the line depends on the values and the number of dashes corresponds to the number of weights. Similarly, CNNVis [10] visualises the architecture of a convolutional neural network, showing weights as weighted lines connecting the different nodes of the network. The latter visualisations require also showing the architecture of the model, which can be challenging to do for the numerous types of architectures possible, thereby making the connected weight visualisation design less practical for our purposes.

5 Prototype

This section describes a prototype we build to test out different visualisation and interaction concepts for the federated learning scenario, using the binary DGA classifiers from T5.3. First, we explain our visualisation concepts and then we report on some technical implementation details. Lastly, we describe discarded visualisation concepts and discuss the utility we see in our approach so far.

[1] https://playground.tensorflow.org

5.1 Visualisation Components

This subsection describes the different visualisation components we developed for the visual analytics prototype. All images for the visualisation components were created using a subset of the CESNET dataset (original size approx. 360k instances) provided by partners at RWTH Aachen. Our dataset contains 50000 instances and we use the third fold for the NYU classifier that was trained with data from CESNET, MU, RWTH Aachen and Siemens and initialised with a pre-trained model.

5.1.1 Performance Overview

One vital component that the user interacts with at the start is the performance overview. This is a familiar type of visualisation, showing the performance for all parties involved in the federated learning process as a line chart. The x-axis denotes the epoch, i.e., it shows the time while the y-axis denotes the performance value. The latter can be one of the following three, which the user can choose via a dropdown widget:

- accuracy
- false-positive rate
- false-negative rate

Such a line chart, although a basic visualisation, allows users to gauge the overall performance over the course of the training process. As Figure 2 shows, the chart also indicates the currently selected epoch with a light grey rectangle, which is important for the other visualisation components that only show data for a single epoch. The user may change the selected epoch by using the slider above the chart.

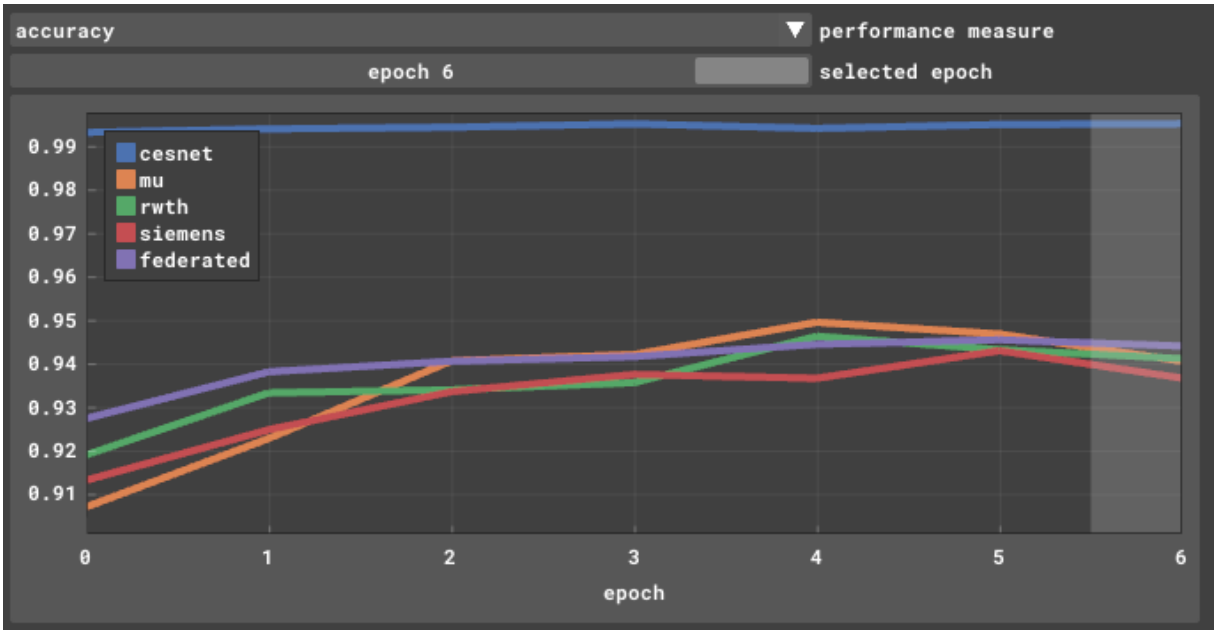

Figure 2: The performance overview visualisation showing accuracy against training epoch, with the last epoch being selected, as indicated by the light grey band in the back.

5.1.2 Overlap Matrix

The overlap matrix functions as an extension of the performance overview, but is limited to visualising data for the selected epoch. It is a matrix with rows and columns denoting the different participating parties (ref. Figure 3). Each cell of the matrix consists of four rectangles, each one showing one of the following:

- overlap of true positives between row and column party

- overlap of false positives between row and column party
- overlap of true negatives between row and column party
- overlap of false negatives between row and column party

As an example, the cell in row RWTH and column CESNET shows how many true positives, false positives, true negatives and false negatives of the RWTH classifier are treated in the same way by the model from CESNET. The percentage of overlap is indicated using the viridis colour map. Note that a cell is only drawn when its value is greater than zero. Which type a cell refers to is encoded both in its position and written across the cell in an abbreviated fashion (TP = true positive, FP = false positive, TN = true negative and FN = false negative). Similar to the performance overview, the user can see the performance for the selected epoch, but they can also get an idea of how this performance aligns between the different parties. For models with very high accuracy, the overlap should be large, while models with lower accuracy may vary more considerably in performance on different subsets of the dataset, or they may all struggle with the same data subset. The different cells also function as a means for the user to filter the data for the subsequent cluster-based visualisations by clicking on a cell. This selection can be reset using the “reset selection” button above the overlap matrix (see Figure 3).

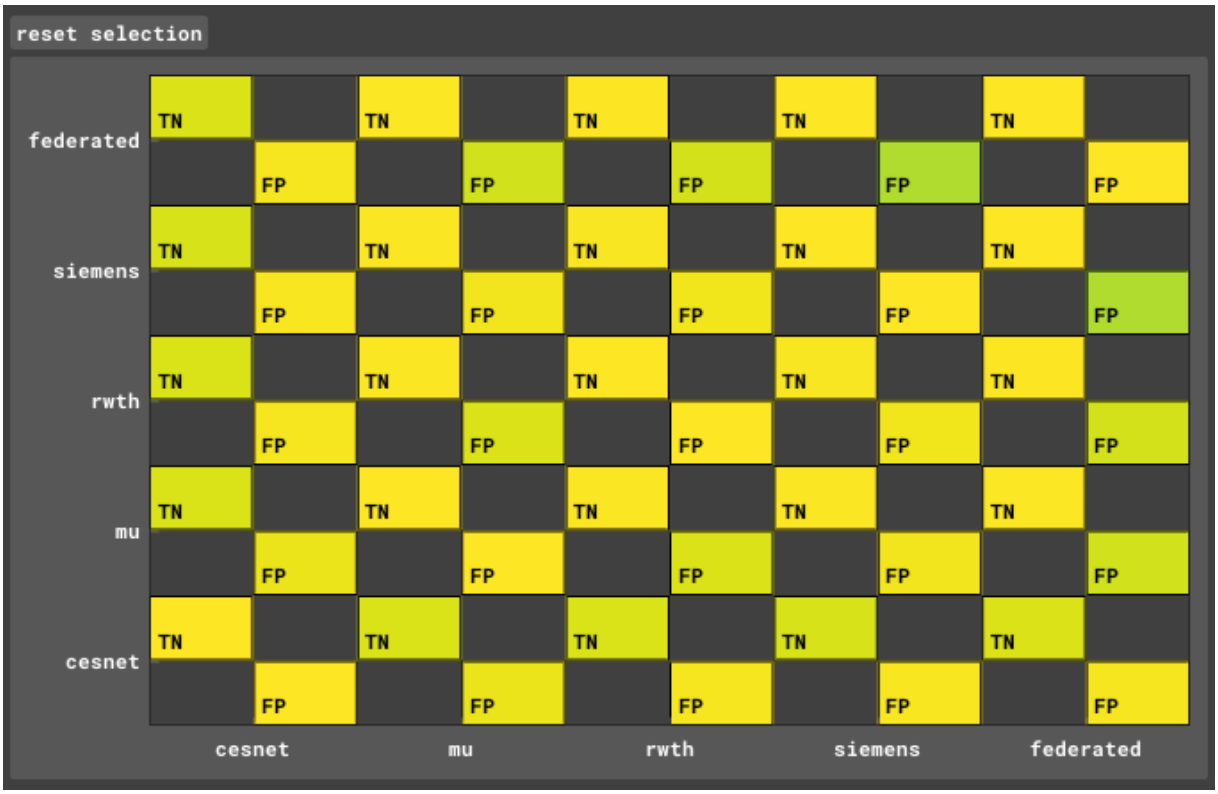

Figure 3: The overlap matrix visualisation showing overlap between models in regards to true positives, false positives, true negatives and false negatives, coloured using the perceptually uniform sequential colour map viridis (purple to blue to green to yellow).

5.1.3 Cluster Overview

Compared to other approaches that require the user to find interesting subsets in the data, we make use of clustering to let the user explore the dataset and its subgroups. The clustering is performed on the predictions scores of the different models as well as the ground truth score. Given M models from m_1 to m_M for one epoch, a dataset D

containing N instances and ground truth values t_1 to t_N, the vector used for the clustering algorithm for each d_i in D has the following form:

$$\left(p_{m_1}, p_{m_2}, \dots, p_{m_M}, t_i\right) for\ d_i \in D\ with\ i = 1 \dots N$$

$$where\ p_{m_j}\ is\ the\ prediction\ score\ for\ model\ m_j$$

In order to give the user more control and let them steer their analysis, we employ k-means as the clustering algorithm of choice, since it allows the user to set the number of clusters and itself uses data points from the dataset to seed the clustering algorithm.

There are different options concerning the initialisation of the algorithm, where the simplest strategy simply picks *k* random points from the data as centres or centroids. While easy and fast, this strategy is quite volatile and the result is very sensitive to the representative quality of these randomly chosen points. Another option is to use Forgy's method, which randomly assigns a cluster to each data point and consequently chooses the resulting cluster centres as starting points. A different strategy, found by Celebi et al. [11] to perform generally well, is k-means++ [12]. Here, the algorithm starts with one random point from the dataset as the first cluster centre. Then, the distances between all data points and the nearest centres that were already chosen are computed. A new centre is then chosen by using the previously computed (squared and normalized) distances as the probability distribution for drawing points from the pool of leftover points. This makes it more likely that a point with a large distance to all other centres is chosen. This procedure is repeated until the desired number of centroids has been found. This initialisation strategy avoids accidentally choosing cluster centres that are very close to each other, which may result in a bad clustering result. However, it may be somewhat sensitive to outliers. Although k-means++ is not the only initialisation strategy to perform well according to Celebi, it is fairly easy to implement and does not require too much time to compute.

5.1.3.1 Cluster Histogram

The first information we show concerning the clustering result is a histogram of the cluster sizes and, in case we have filtered the data, the rest of the dataset (cf. Figure 4). In addition, we let the user adjust the number of clusters via buttons and an input field above the histogram.

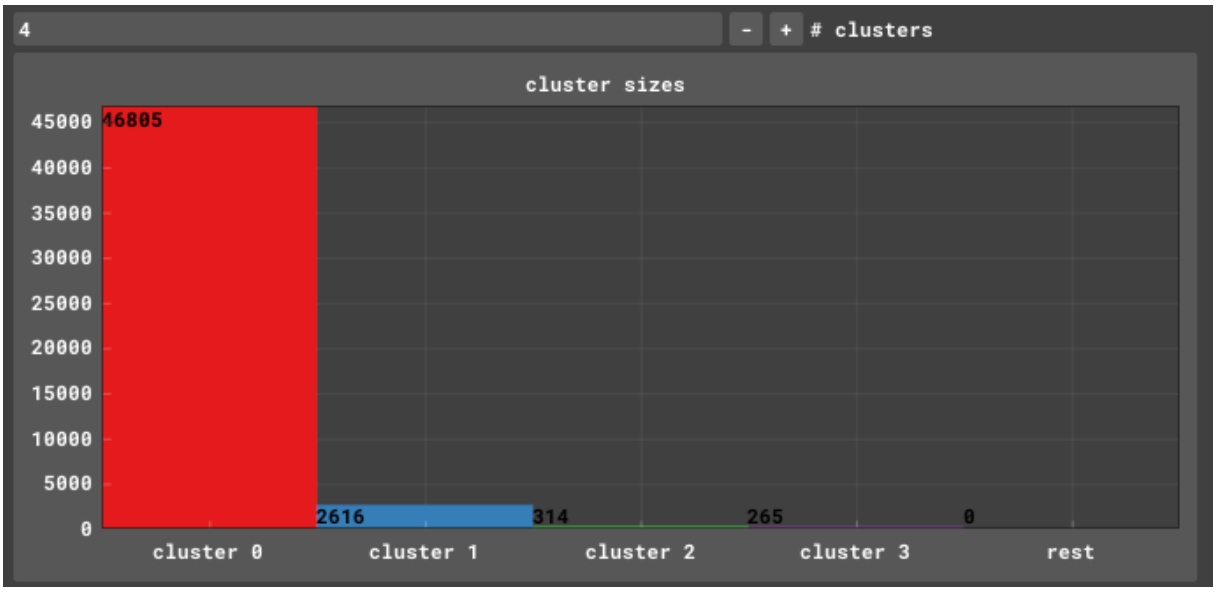

Figure 4: Cluster histogram that shows four clusters and the rest, which in this case equals zero since the data was not filtered.

5.1.3.2 Parallel Coordinates

Below the cluster histogram, the user can see the actual data that is clustered in a parallel coordinates plot, where each line is coloured by its cluster membership. Each cluster can be hidden by clicking on its legend label, which allows the user to deal with overdraw issues. Clusters with a smaller range of values may be investigated by hiding other clusters and then double-clicking the plot to automatically fit the viewport to the available data. Axis scaling or legend positioning may also be adjusted, if so desired, by right-clicking on the plot and changings its settings.

In particular, we noticed that the homogeneity of some clusters might suggest that they do not contain as many instances as other clusters. An example for such a case can be seen in Figure 5, where the red cluster does not seem to be significantly larger than other clusters. However, looking at the cluster overview in Figure 4, we can see that the red cluster actually makes up the largest part of all instances in the dataset with 46k instances, compared to a size of 3184 instances for all other clusters combined. Consequently, the cluster histogram proves to be a good companion in these cases by letting the user interpret the parallel coordinates plot more accurately.

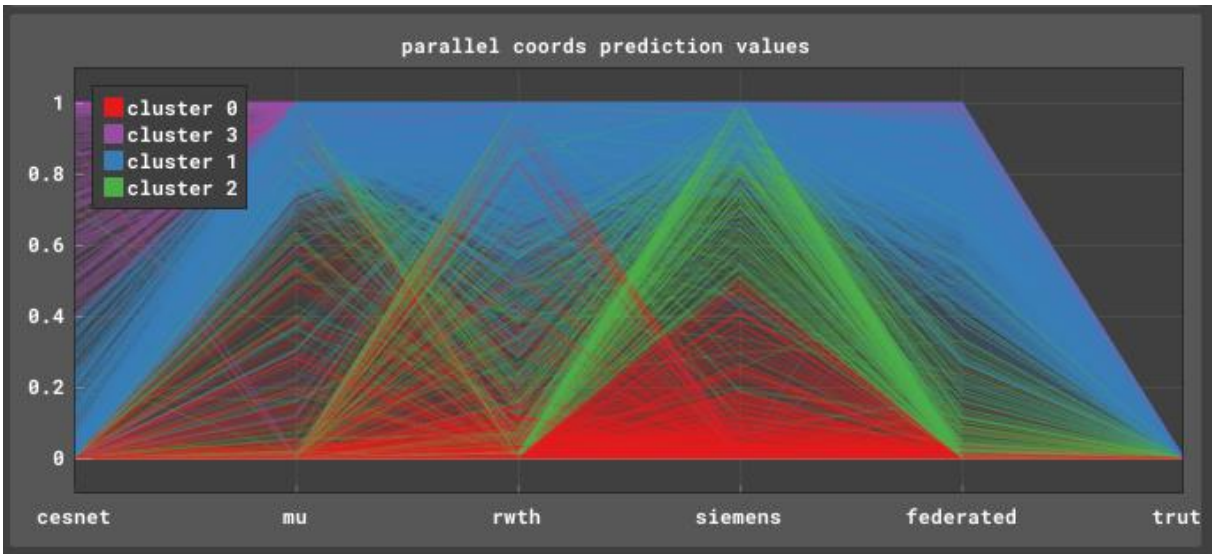

Figure 5: Parallel coordinates plot for all clusters, showing how models (at the current epoch) classify instances and how that relates to the ground truth.

5.1.4 Cluster Details

The cluster details view functions as a means to give the user a holistic view of the data in each cluster, which is especially challenging given that the data we are working with is non-natural text data (domains). The visualisations described in the following paragraphs are created separately for each cluster, which the user can switch between via vertical scrolling. Both visualisations in the cluster details view employ the perceptually uniform colour map viridis that was already used for the overlap matrix beforehand, ensuring consistency across visualisations.

5.1.4.1 Character Heatmap

First, the user can inspect a heatmap of character occurrences in the cluster (ref. Figure 6). This enables the user to quickly gauge different features of the domains, such as the number of subdomains, the length, and the frequency of specific character groups like numbers or special characters. It also makes comparison between the clusters easier, since the different patterns can be juxtaposed visually.

On the x-axis, we plot the positions of the characters, while the y-axis refers to the actual characters, which use the same character to integer mapping that is applied to prepare the data for input into the model (cf. Listing 1).

```
{
  'a': 1,'b': 2,'c': 3,'d': 4,'e': 5,'f': 6,'g': 7,'h': 8,'i': 9,
  'j': 10,'k': 11,'l': 12,'m': 13,'n': 14,'o': 15,'p': 16,'q': 17,
  'r': 18,'s': 19,'t': 20,'u': 21,'v': 22,'w': 23,'x': 24,'y': 25,
  'z': 26,'0': 27,'1': 28,'2': 29,'3': 30,'4': 31,'5': 32,'6': 33,
  '7': 34,'8': 35,'9': 36,'-': 37,'_': 38,'.': 39
}
```

Listing 1: Character to integer mapping used for the character heatmap.

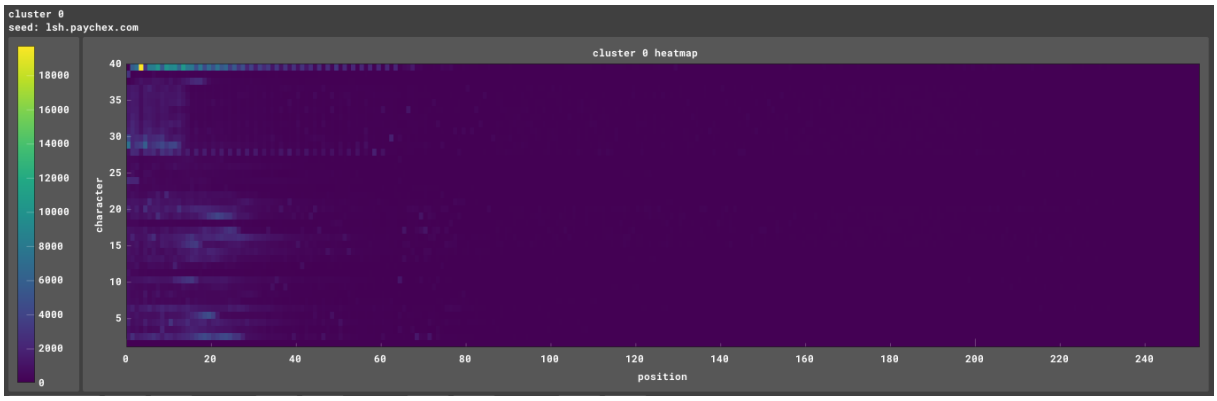

Figure 6: Character occurrence heatmap for cluster 0.

Above the character heatmap, we also show the cluster number and the seed instance used for the clustering initialisation. Left of the heatmap, the user can see the colour map and its corresponding values for the heatmap. To give users a better idea of the types of instances that match the cells of the matrix, we show a tooltip when the mouse cursor hovers over a cell that displays the following information:

- the position of the character
- the character (transformed to show the character, not the mapped integer)
- the value of that cell, i.e., the number of instances with that character at that position in the domain
- a maximum of 15 matching instances

5.1.4.2 Parallel Coordinates

In another parallel coordinates plot, we show all instances of the cluster in regards to their predictions for each party, and their distances to all existing clusters (ref. Figure 7). We colour the lines according to their distance to the centroid of their cluster, but the line colour is scaled globally across all clusters for easier comparability.

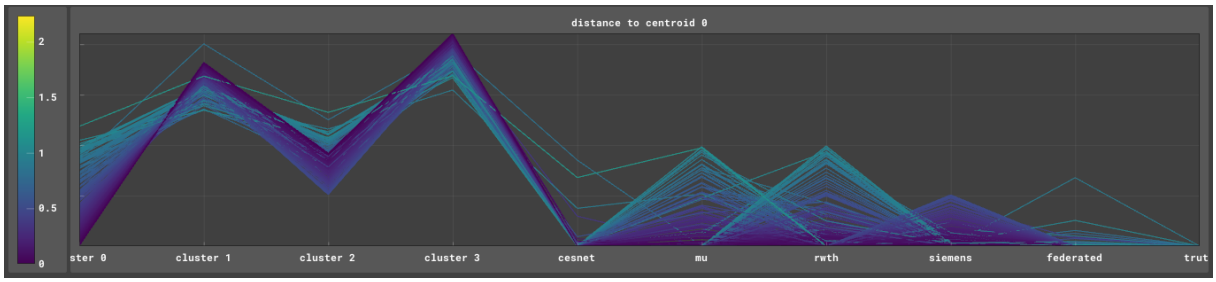

Figure 7: Parallel coordinates plot for cluster 0 showing distances to all clusters, what the different parties predict for the instances in that cluster and what the ground truth for those instances is.

5.1.5 Instance Table

The instance table is a supplementary visualisation that provides the user with the option to have a closer look at individual instances in the dataset, illustrated in Figure 8. For each instance, we show the domain, the ground truth, the assigned cluster and a small scatter plot with the predictions for each party's model. In this chart, we also indicate the ground truth's prediction score and the cluster's mean prediction score with a line and their standard deviation with a coloured semi-transparent band. This can make deviations from that mean visible at a glance, which allows the user to find

interesting instances faster. We include pagination to improve performance and allow the user to skim through the data more efficiently in comparison to showing one large table.

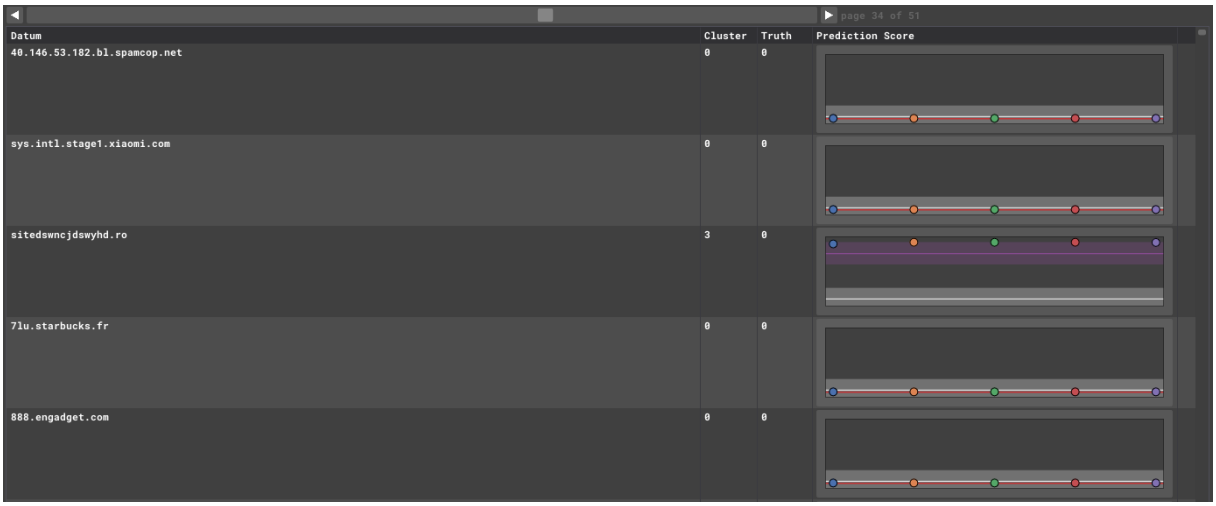

Figure 8: Instance table for a filtered selection of data.

5.2 Technical Details

Since we expected to deal with a larger amount of data and potentially perform live inferencing or apply explainable AI methods, we chose to implement the prototype as a standalone desktop application in C++. The programming language C++ has a reputation for being fast and allowing for a high degree of optimization to squeeze the maximum performance out of an application. Although it is not our goal to develop a highly optimized prototype, as the name *prototype* implies, we nevertheless believe that this choice of application and language allows for more room in regards to experimentation.

For easier handling of the data, we make use of the NumCpp[2] library, a C++ wrapper of the ubiquitously used python framework NumPy[3]. Regarding visualisation, using C++ opens up the option of rendering with GPU acceleration using a common graphics API such as OpenGL[4].

However, usually such applications consist of a lot of boilerplate code to perform even basic tasks such as rendering multiple objects. Creating a functioning user interface using just a graphics API, no less one that is modern in its appearance and responsive to users' needs, requires an inordinate amount of work. To avoid this problem and simplify prototyping, we use the ImPlot[5] and ImGui[6] frameworks to build the visualisations and interface elements. Both frameworks use the immediate mode GUI (graphical user interface) paradigm, which is not stateful in the same way many conventional retained mode GUIs are. Instead of updating visualisations and UI widgets when their (semantic) state changes, immediate mode GUIs try to avoid statefulness in order to simplify the software, for example by passing the complete state to the rendering function each time. The downside of this paradigm is that this often results in frame-by-

[2] https://github.com/dpilger26/NumCpp
[3] https://numpy.org
[4] https://www.opengl.org/
[5] https://github.com/epezent/implot
[6] https://github.com/ocornut/imgui

frame rendering, which makes it a fitting candidate for 3D rendering and scientific visualisation, which often has to render on a frame-by-frame basis anyway. This is different to how information visualisation usually operates, which is closer to drawing on demand, but this can still be a good choice if we want to display more computationally demanding visualisations that profit from GPU acceleration like continuous parallel coordinates. Even just drawing a large number of data points can often cripple web-based applications that use the HTML canvas or SVG for drawing, whereas GPU accelerated desktop applications can draw larger amounts of data more easily.

5.2.1 ONNX Runtime

For our prototype, we calculate the visualisations based on pre-computed results stored on the disk. In scenarios where the learning process has already finished, there is little to nothing to gain by performing live inference, which is slower than simply reading predictions from disk. However, for the future we can envision performing the training live in combination with our visualisation or using explainable AI (XAI) techniques that require computations on the model. Consequently, we decided to include support for the ONNX Runtime[7] to load models and perform (live) inference. The ONNX Runtime is a machine learning runtime that uses the ONNX[8] (Open Neural Network Exchange) standard format for machine learning models. We chose this runtime since the ONNX standard tries to present a unified option with which models can be represented and shared. There are many implementations of conversion tools for other software such as tensorflow[9], keras[10] and caffe[11], making it possible to create ONNX models even if the original model format is different. In addition, the ONNX runtime is available for all popular operating systems Windows[12], Linux distributions and MacOS[13] and has GPU support for accelerated inferencing and training.

5.3 Discarded Concepts

In this subsection, we detail some of the concepts we designed on paper or even implemented in a rudimentary fashion but ultimately discarded in favour of other options.

5.3.1 Parallel Coordinates for Domains

In a first version of the prototype, we showed a parallel coordinates plot with a y-axis for each possible character in a domain, i.e., 253 vertical axes. The values of the y-axes range from 1 to 39, as they do in the character heatmaps, and denote the character for that position in the domain. Then we drew the cluster centroids as lines in the plot and add the standard deviation for each cluster as a semi-transparent coloured band. However, we noticed that this type of visualisation does not provide the granularity we hoped for, which is why we opted for the combination of character heatmap and parallel coordinates in the cluster details view. In particular, showing the standard deviation does not appropriately communicate the structures of domains in the cluster, sometimes even being misleading due to the nature of the mapping from characters to integers, which does not follow the same semantics.

[7] https://onnxruntime.ai
[8] https://onnx.ai
[9] https://www.tensorflow.org
[10] https://keras.io
[11] https://caffe.berkeleyvision.org
[12] https://www.microsoft.com/
[13] https://www.apple.com/de/macos/

5.3.2 Weight Strips

In order to give the user an overview of how the weights change, we considered showing the maximum negative and maximum positive change per layer consecutively as a coloured strip, for each party. This would have the advantage of showing patterns that, with sufficient distinction in values, could point to differences in optimization goals. However, only showing these extreme values can be somewhat misleading and does not provide enough information to really reason about. In addition, the shape of the weights may differ between layers, which makes the comparison of extreme values even more unlikely to provide interesting insight.

5.3.3 Model and Class Prediction Changes

Initially, we envisioned showing the changes in predictions between the selected epoch and its predecessor based on the party and based on the class in two stacked bar charts. While there may be some benefit to seeing the number of changes, understanding these changes would require additional visualisations that let the user investigate which particular instances changed, for which parties and for which reasons. Instead, we think the clustering approach provides an easier way of achieving a similar objective: finding interesting data subsets that manifest in performance differences. In particular, the clustering in combination with the parallel coordinates can show how one instance is treated by all models, thereby allowing the user to see interesting groups that would be harder to find by looking at the prediction changes in isolation, i.e., not being able to see how a prediction (change) for one party relates to the predictions for the other parties.

6 Discussion

In this section, we report the results we observed when using our visual analytics prototype in relation to previous findings for task T5.3 and discuss the problems we see in our approach.

6.1 Observations

The performance overview gives users a familiar starting point and simultaneously sets the stage by showing overall performance across all epochs. From there on, the user can choose an epoch to investigate in more detail. We think the clustering provides a useful way for further interaction with the data. In combination with the parallel coordinates plots, it gives the user a good overview of per-epoch prediction differences between parties and how these can be grouped. During our usage of the system, the clustering also seemed to be rather stable, producing very similar results for the same data. The character heatmap makes it easy for the user to compare clusters in terms of their domains via visual pattern comparison. The per-cluster parallel coordinates show the inter- and intra-cluster distribution of data points as well as the relationship to model predictions and ground truth. Thus, these components cover the first three goals formulated in section 4.1.

In addition, all of our designed visualisations except for the character heatmap work irrespective of the data type. This means that our approach could be adapted for other models, e.g., convolutional models that classify images. This would only require substituting the character heatmap with a similarly informative visualisation for the given data type. While our case works with a binary classifier, most of our visualisations could also be employed for multi-class classification. This could be achieved by clustering either the complete output of the models directly or by using the predicted class index

in conjunction with the confidence score for that class. The latter would be easier to incorporate since it requires one more data entry for each model, while the former may provide a clearer picture of model behaviour at the expense of requiring more data. The one visualisation that would require modification for a multi-class scenario is the instance table. It would either need to only show the prediction (i.e., the class index) or the complete predictions scores for all classes, which could be achieved with the help of parallel coordinates or a heatmap.

6.1.1 Use Cases

This subsection documents two simple use cases and the insight that can be attained with the use of our developed prototype. First, we explore the different clusters taken from the same scenario as shown in previous figures in section 5.1. For a dataset of 50000 benign domains collected by CESNET, we explore the clusters found in the last epoch of the federated learning scenario. In our scenario, all models start with a relatively high prediction accuracy (cf. Figure 2). However, CESNET significantly outperforms all other models throughout the complete training process. This is likely due to the choice of dataset: Since we are using CESNET's benign data, it is to be expected that the CESNET model performs better on this data than all other models.

We start out with three clusters at default, but looking at the parallel coordinates containing all the data coloured by their cluster membership, we see that there is more variety in prediction scores, so we increase the number of clusters to four (cf. Figure 5) and proceed to the cluster details view. Looking at Figure 9 to Figure 12, we can see the character heatmaps for all four clusters. They show that the first cluster contains the longest instances and that some of these instances show a pattern of a dot alternating with another character (in the first line, starting at the top). The other three characters heatmaps are more similar to each other, but still show some minor differences regarding the choice of characters and where the most dots occur.

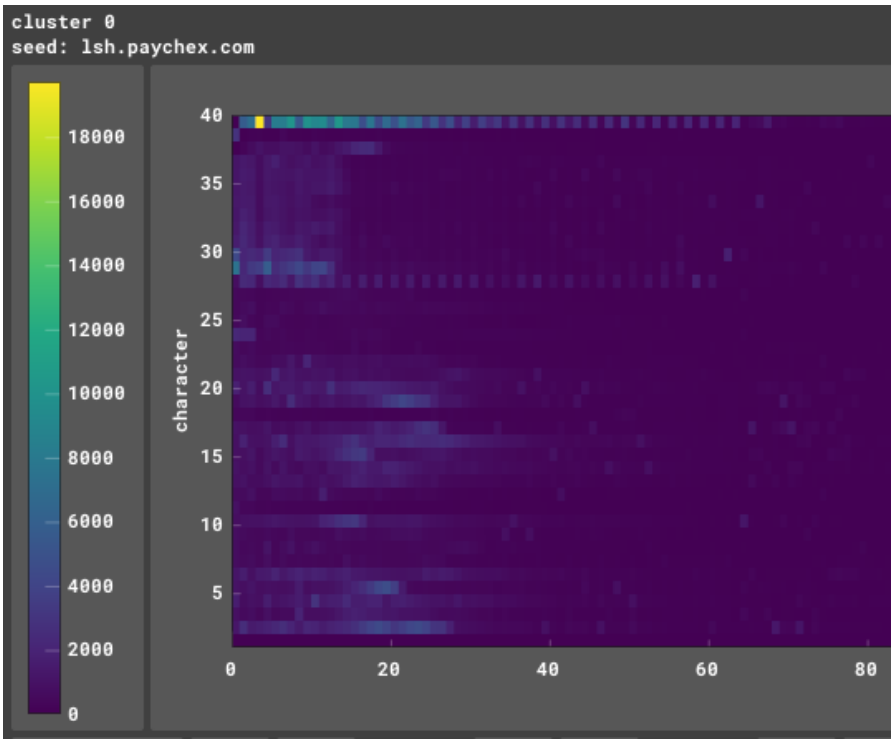

Figure 9: Character heatmap for cluster 0 in the first use case, clipped for better visibility.

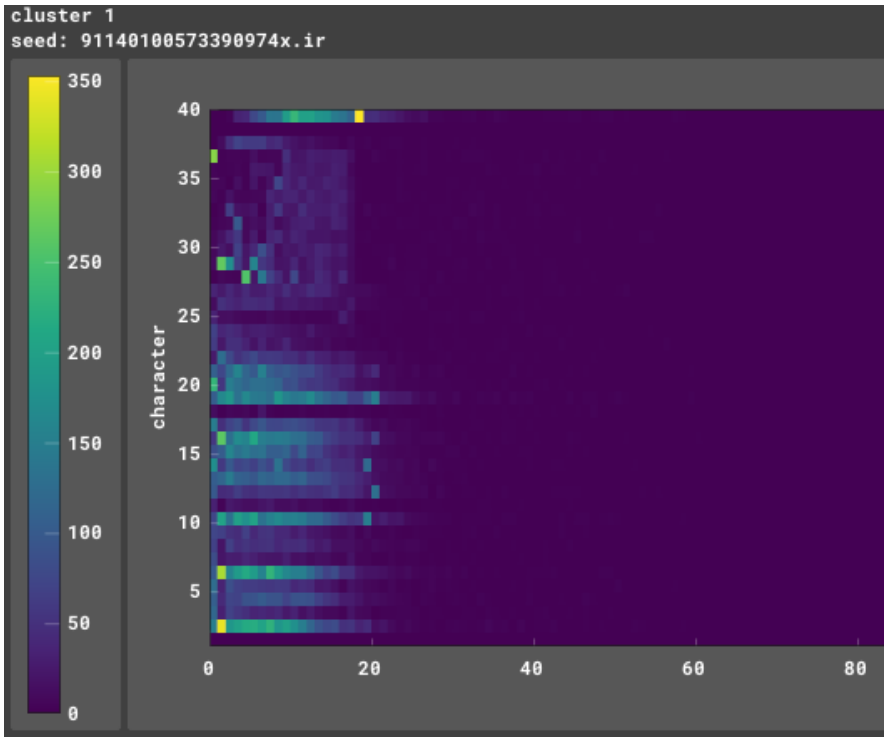

Figure 10: Character heatmap for cluster 1 in the first use case, clipped for better visibility.

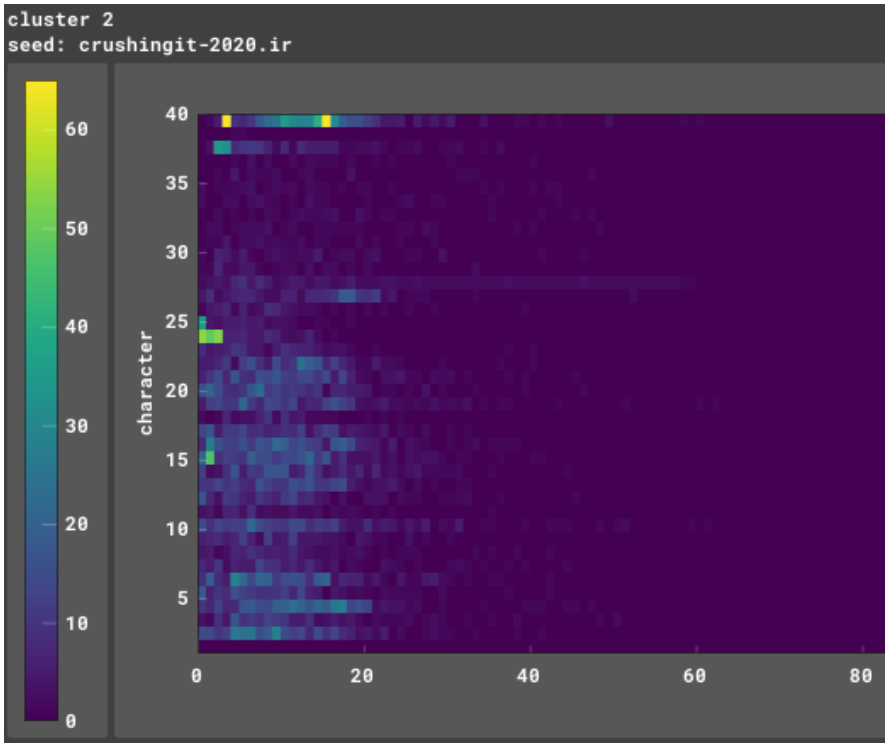

Figure 11: Character heatmap for cluster 2 in the first use case, clipped for better visibility.

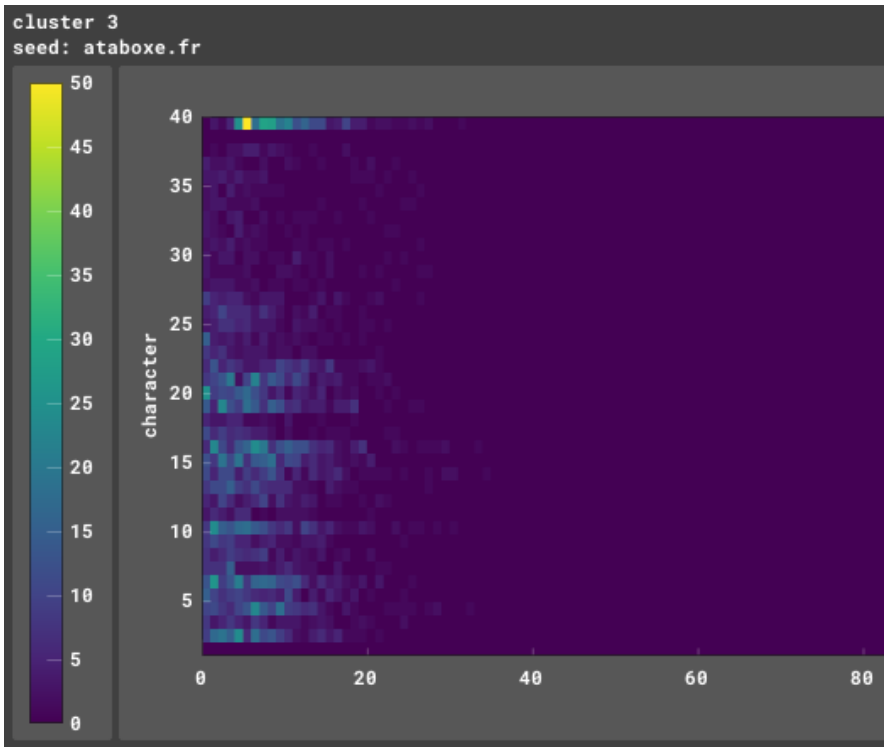

Figure 12: Character heatmap for cluster 3 in the first use case, clipped for better visibility.

In the parallel coordinates of Figure 13 to Figure 16, we can see how close the clusters are to each other, how spread instances inside a cluster are and what the different parties predict for the cluster's instances. For cluster 0 in Figure 13, we can see that CESNET is very accurate with only a few instances being misclassified, while MU and RWTH seem to struggle with the instances at the outer edge of the cluster. Incidentally, these do not necessarily seem to be the same instances, as we can see that some lines cross the y-axis for MU at 0 and for RWTH at 1, and vice versa.

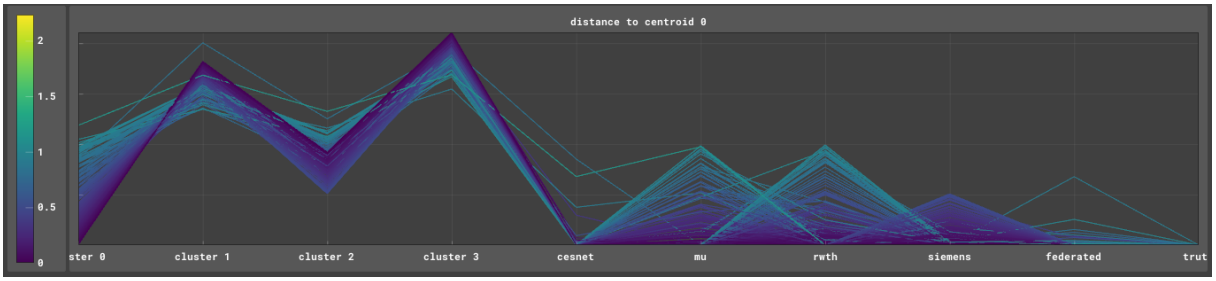

Figure 13: Parallel coordinates for cluster 0 in the first use case.

In Figure 14, we seem to have a cluster where CESNET performs well on all instances, but the other parties seem to struggle a lot more, especially SIEMENS. In Figure 15, we see a similar scenario, although the other models seem to perform a little better than for cluster 1. Finally, in Figure 16, we see that cluster 3 deviates from the previous pattern were CESNET always performed almost perfectly. It still largely performs better than other parties do, but misclassifies a larger number of the instances in that cluster. Interestingly, there seem to be a few instances in that cluster where the MU model is very confidently correct, but no other model is. These particular instances may be outliers in the cluster, since they have a much larger distance to the cluster centroid than all other instances in the cluster.

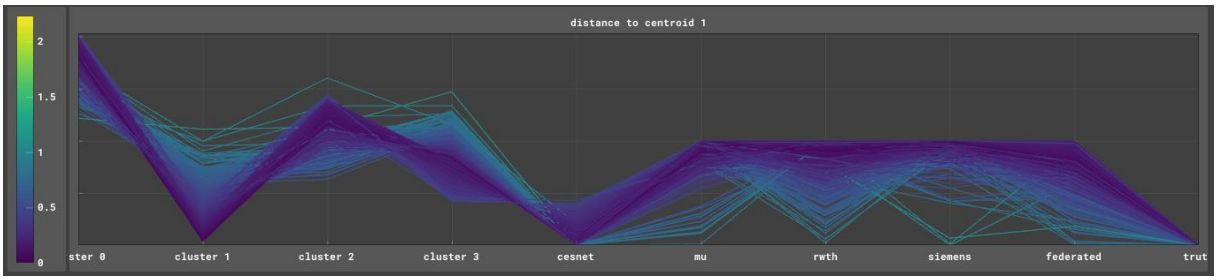

Figure 14: Parallel coordinates for cluster 1 in the first use case.

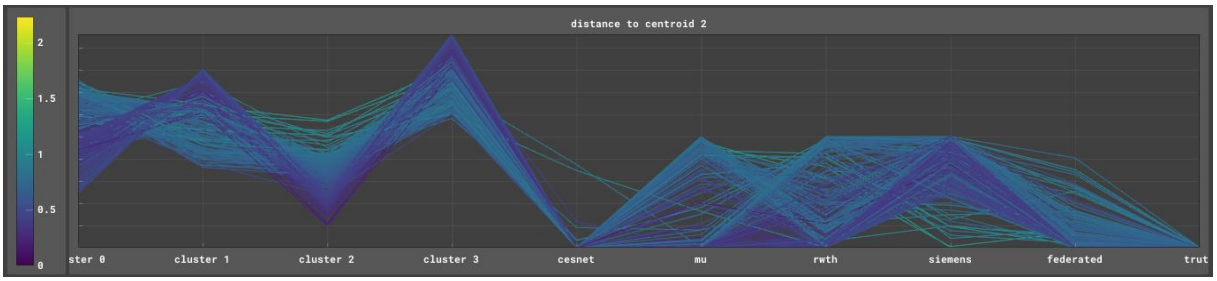

Figure 15: Parallel coordinates for cluster 2 in the first use case.

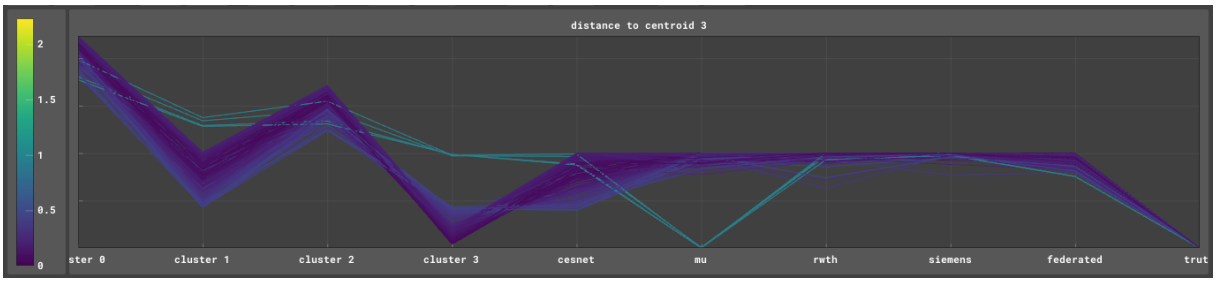

Figure 16: Parallel coordinates for cluster 3 in the first use case.

In the second case, we created a small dataset of approximately 50k instances by randomly sampling data from three other datasets, which are briefly described in the following:

- CESNET dataset, made up of only benign domains collected at CESNET
- MU dataset, made up of only benign domains collected at MU
- Binary POC dataset, made up of malicious (collected from public sources) and benign (collected at RWTH Aachen) domains

We randomly sampled the same number of samples from each dataset to create a new dataset with 8333 malicious and 41665 benign domain instances. The performance overview and overlap matrix for this dataset look quite similar to those for only the smaller CESNET dataset (cf. Figure 17) with the exception that the overlap matrix shows that the CESNET model is the only one to predict some false negatives. This overall similarity is to be expected, since the CESNET model seems to perform well on its own data and the data from MU. In the cluster overview and the related parallel coordinates plot, we can see that the clustering algorithm puts almost all malicious instances into one cluster (cluster 1).

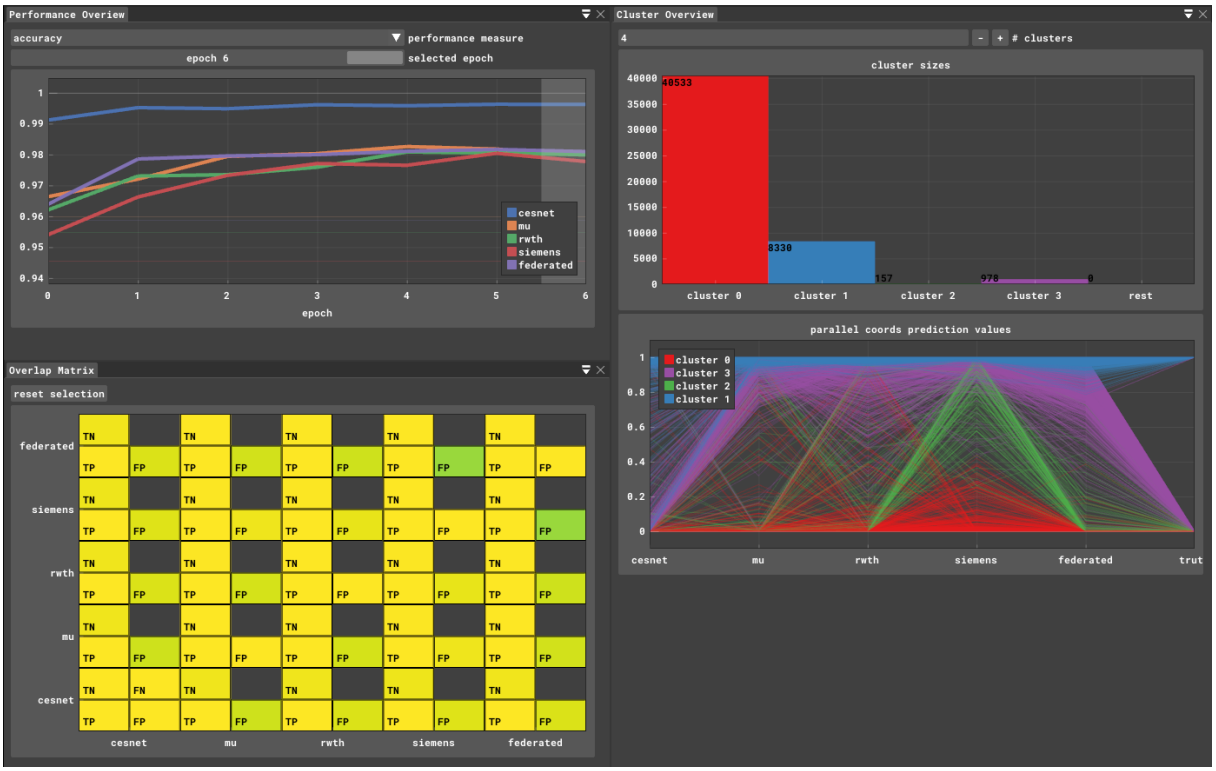

Figure 17: Overview visualisations for the second use case, showing the performance overview, overlap matrix, cluster overview and parallel coordinates plot.

In the cluster details view for the first cluster in Figure 18, we can see that the cluster contains longer domains than the other clusters and that the parallel coordinates plot also looks similar to that for cluster 0 in the first use case in Figure 13, with CESNET performing very well and MU and RWTH having trouble with some instances.

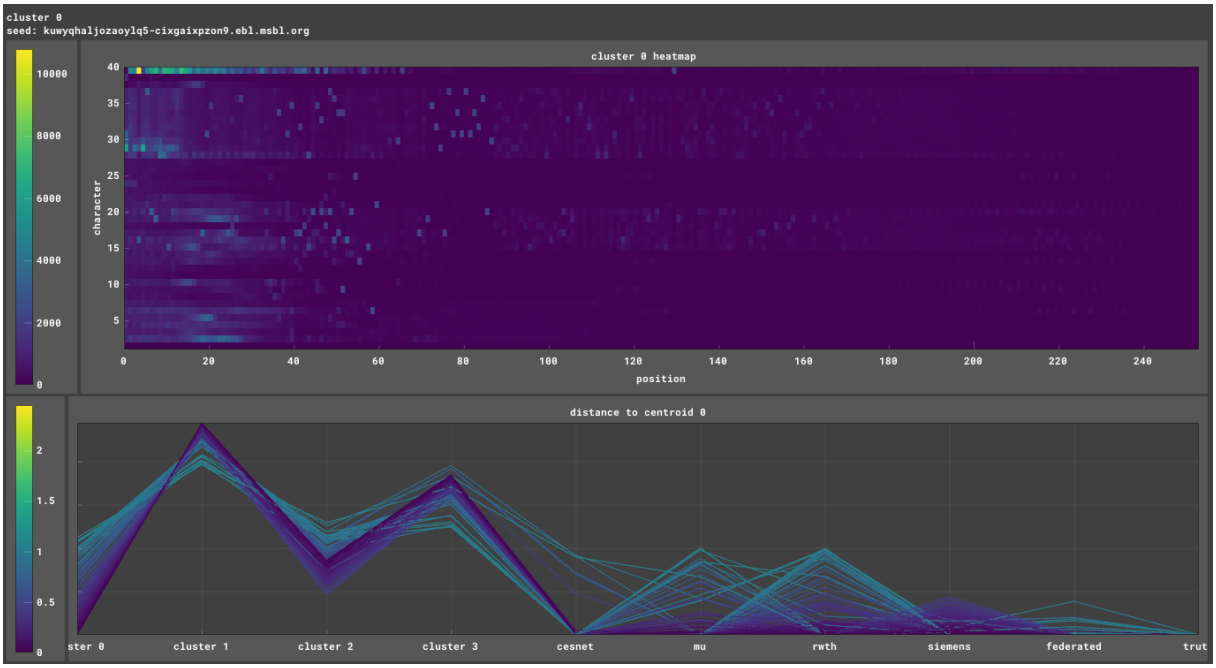

Figure 18: Cluster details view for cluster 0 in the second use case.

In Figure 19, we see that domains are not as long as for the first cluster. In addition, the seed domain may already suggest that this cluster contains malicious instances, at least to a knowledgeable user. This is confirmed in the parallel coordinates plot,

where we can see that this cluster only contains malicious instances (truth = 1). Here it is also visible that the CESNET model does not predict all of these instances correctly, while the other parties do. Interestingly, the instances furthest away from the cluster centroid seem to be those that prove troublesome for the CESNET model. Moreover, it can also be seen that the federated model, while not making incorrect predictions for this cluster, has a lower confidence score for the instances the CESNET model does not classify correctly. Since the federated model averages all other models' weight updates, this is an expected consequence.

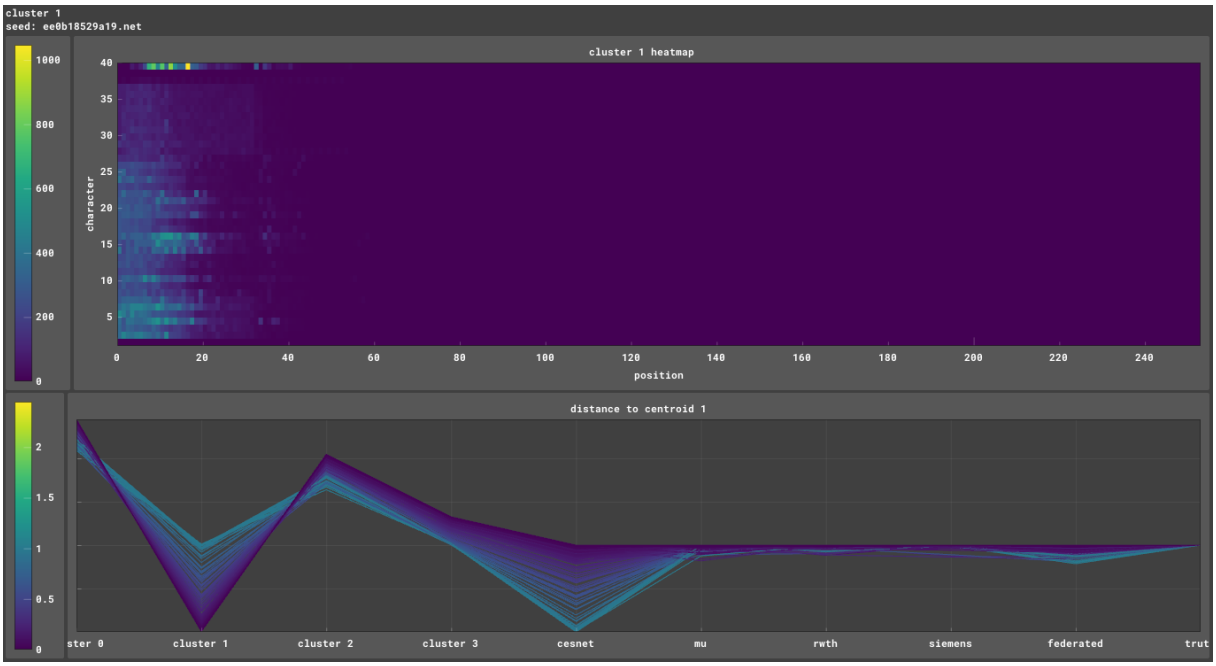

Figure 19: Cluster details view for cluster 1 in the second use case.

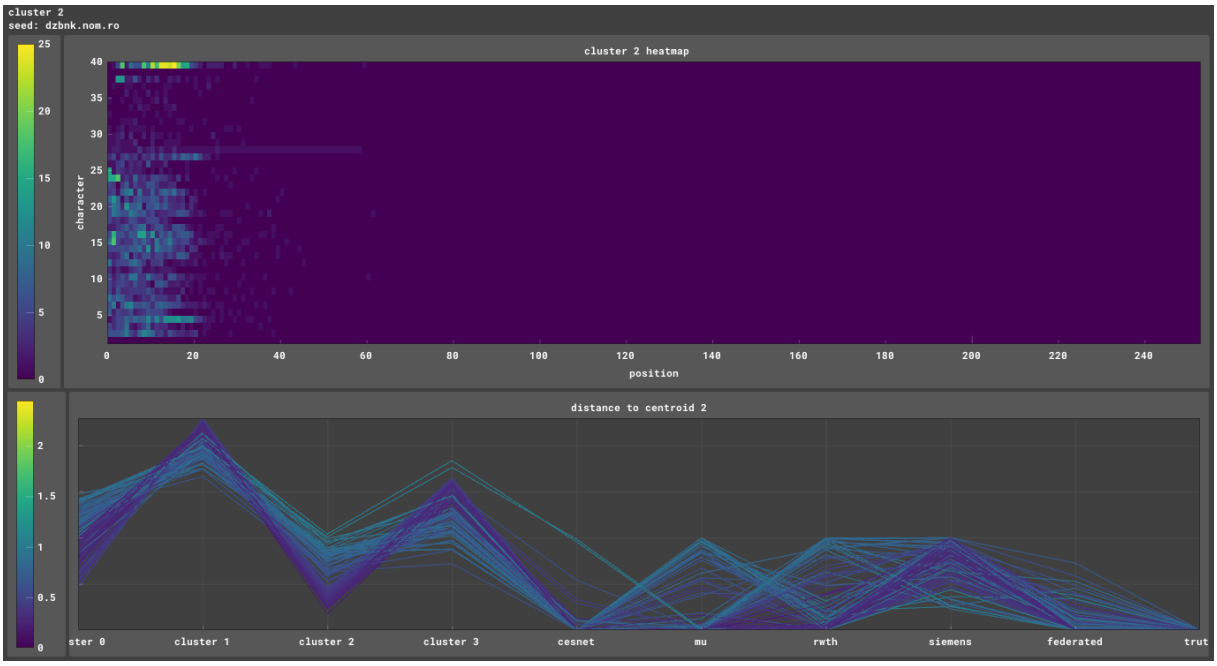

Figure 20: Cluster details view for cluster 2 in the second use case.

In the cluster details views for the two final clusters in Figure 20 and Figure 21, we can see the (largely) benign instances that several models struggle to classify with a high confidence or even correctly. These clusters are rather small, as indicated in the cluster

overview in Figure 17, but may provide interesting subsets to analyse further, e.g. using the instance table or by filtering the data with the overlap matrix and exploring the new resulting clusters.

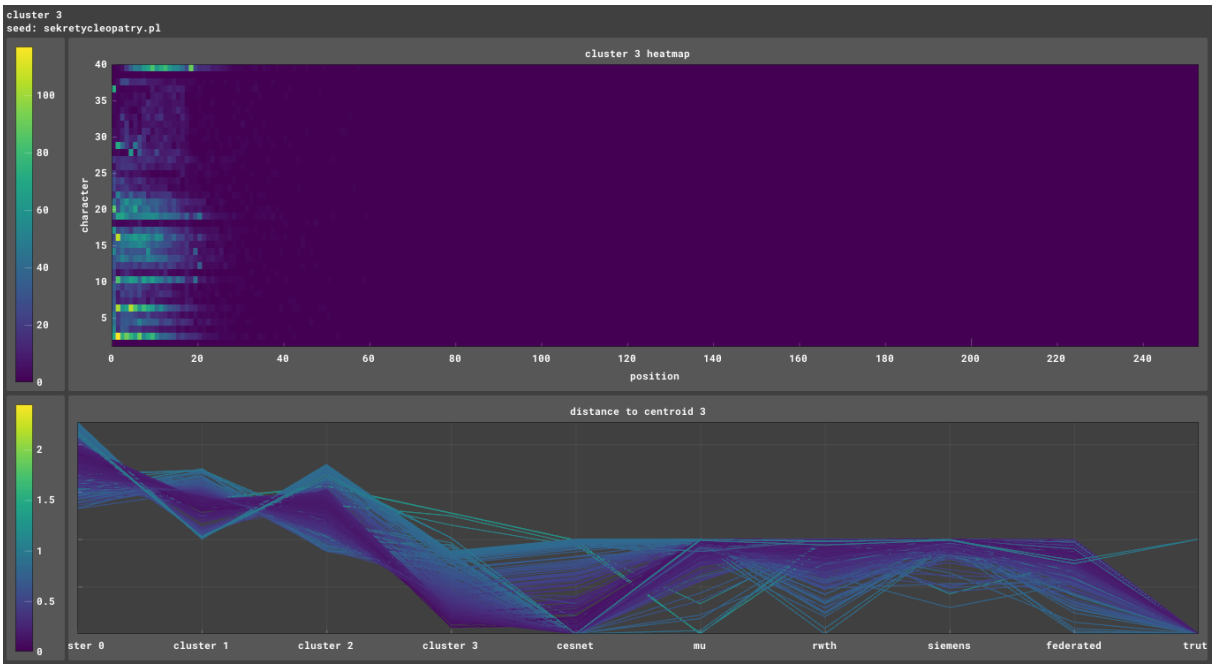

Figure 21: Cluster details view for cluster 3 in the second use case.

Overall, these insights overlap with the evaluation results reported in deliverable D5.6. When looking at federated learning from the view of one party, i.e., seeing how the federated model performs on just data from one party compared to the party's own model, it can be seen that a federated model may perform worse. This is expected, since the party's own model should always perform better on its training data, while the federated model combines many models trained to classify different-looking data. However, when considering performance on multiple data sources, it is likely that the federated model performs better or is more confident (cf. Figure 18) than any single model.

6.2 **Problems**

Concerning problems and areas for future improvements, we noticed that more work is required to make the application viable for practical use in big data scenarios: It needs to be faster, concerning the speed with which it reacts to user interaction, and the visualisations need to be modified to scale better for large amounts of data. As we mentioned in the previous section, the C++ language is a good option for such optimization, and we consider the following improvements to provide substantial benefit in terms of performance:

- Parallelise reading data from disk for the different epochs or employ a database to speed up data queries (which also allows for the easier definition of complex queries to select data subsets)
- Parallelise the clustering algorithm
- Draw continuous parallel coordinates or use aggregated data when the number of lines in parallel coordinates plots surpasses a certain threshold

Another problem we see with our current prototype is its inability to let the user fine-tune the clustering further, which may be desirable in some cases. This could be addressed by providing more filtering options for the data and by letting the user manually select seed instances for the clustering algorithm. The latter option is complicated by the fact that finding and selecting a fitting instance requires a visualisation that can give the user a good overview and at the same time provide enough granularity to pick a single instance. One possible solution for both problems would be to construct a view consisting of the following components:

1. A list of the current seed instances and a visualisation of their predictions and ground truth.
2. The parallel coordinates plot used for the cluster details: A parallel coordinates plot showing the distances to these seed instances, model predictions and the ground truth with brushable axes that filter the next component, the data table.
3. A connected data table that lets the user filter and sort based on distance to the seeds, predictions and ground truth that highlights the respective line in the parallel coordinates plot.

This view could enable the following workflow. First, the user can see the seeds used for the clustering before the clustering is performed. Then the user can get an idea of how the data points are distributed around the cluster seeds by looking at the parallel coordinates. Brushing the axis filters the data in the table, allowing for more efficient exploration of a larger number of instances.

A similar brushing functionality for the already existing parallel coordinate plots in the cluster details view is currently being worked on. In combination with the character heatmap, the user could filter the data on the axes of the parallel coordinates plot and then recalculate the character heatmap to only show the character distribution for the selected instances in the cluster. This could give the user an easy means to investigate subsets of the data in more detail.

6.3 **Weights Updates**

Visualising the weights also posed a problem during development. After discarding the weight strips idea, we debated whether we should simply adopt a pixel-based approach, showing a heatmap for each layer of each model for the selected epoch. While this would also allow for visual pattern comparison, it does not scale well in terms of space requirements. However, we sketched the following designs that may provide a better trade-off between space requirements and visual comparability. Given a one-dimensional weight update vector, we propose to use a line chart showing the weight update, where the x-axis is the value index in the weight vector and the y-axis denotes the actual value. Then we can show weight differences for one layer and all models in a single chart. For higher dimensional weight updates, the situation is more complicated. To visualise two-dimensional data, we propose the use of a 2D height-map to provide an intuitive way of looking at the weight changes. However, such a visualisation only works per-party and per-layer, meaning that the user would need to compare several of these visualisations to gauge the differences in weight updates between all participating parties.

7 Summary

In summary, this deliverable outlined the challenges and goals for a visual analytics system that aims to support developers and users of the federated learning user case described in task T5.3. We showed what a prototype might look like and that we can use this prototype to make similar observations, regarding model behaviour and performance, to those made in deliverable D5.6. With some more work, we expect our approach to provide a useful tool to analyse and asses such federated learning scenarios, even for other types of data.

For future work, adding a weight update visualisation, as proposed in the previous section, may open up more possible use cases. For example, our system may help in attack scenarios such as poisoning attacks. Usually, a poisoning attack aims to inject samples into a training data set to influence the properties of a model trained on it in a way that suits the attacker's objective. In our federated learning scenario, a malicious party could try to train their model to include a backdoor, e.g., instances that are actually generated by DGAs, but are labelled as benign. Looking at the weight updates may indicate in which directions the different parties are optimising their model, and in turn could indicate whether one party is acting in a suspicious manner. Another aspect to consider in future work is the validation with more data and with users to see whether they can work with the system as it is and whether it conforms to their ideas of how to assess federated learning scenarios.

References

[1] F. Hohman, M. Kahng, R. Pienta and D. H. Chau, “Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers,” in *IEEE Transactions on Visualization and Computer Graphics*, 2019.

[2] S. Mohseni, N. Zarei and E. D. Ragan, “A Multidisciplinary Survey and Framework for Design and Evaluation of Explainable AI Systems,” *ACM Transactions on Interactive Intelligent Systems,* pp. 1-45, December 2021.

[3] D. Ren, S. Amershi, L. Bongshin, J. Suh and J. D. Williams, “Squares: Supporting Interactive Performance Analysis for Multiclass Classifier,” in *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[4] S. Murugesan, S. Malik, F. Du, E. Koh and T. M. Lai, “DeepCompare: Visual and Interactive Comparison of Deep Learning Model Performance,” in *IEEE Computer Graphics and Applications*, 2019.

[5] F. Heyen, T. Munz, M. Neumann, D. Ortega, N. Thang Vu, D. Weiskopf and M. Sedlmair, “ClaVis: An Interactive Visual Comparison System for Classifiers,” in *Proceedings of the International Conference on Advanced Visual Interfaces*, 2020.

[6] M. Gleicher, A. Barve, Y. Xinyi and F. Heimerl, “Boxer: Interactive Comparison of Classifier Results,” *Computer Graphics Forum,* 18 July 2020.

[7] J. Zhang, Y. Wang, P. Molino, L. Li and D. S. Ebert, “Manifold: A Model-Agnostic Framework for Interpretation and Diagnosis of Machine Learning Models,” in *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 2019.

[8] M. Kahng, P. Y. Andrews, A. Kalro and D. H. (. Chau, “ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models,” in *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[9] H. Zeng, H. Haleem, X. Plantaz, N. Cao and H. Qu, “CNNComparator: Comparative Analytics of Convolutional Neural Networks,” in *arXiv preprint arXiv:1710.05285*, 2017.

[10] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu and S. Liu, “Towards Better Analysis of Deep Convolutional Neural Networks,” in *IEEE transactions on visualization and computer graphics*, 2016.

[11] M. E. Celebi, H. A. Kingravi and P. A. Vela, “A comparative study of efficient initialization methods for the k-means clustering algorithm,” *Expert systems with applications,* pp. 200-210, 2013.

[12] S. Vassilvitskii and D. Arthur, “k-means++: The advantages of careful seeding,” in *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, New Orleans, Louisiana, 2006.