ConfusionFlow: A model-agnostic visualization for temporal analysis of classifier confusion

Andreas Hinterreiter*, Peter Ruch*, Holger Stitz, Martin Ennemoser, Jürgen Bernard, Hendrik Strobelt, and Marc Streit

Abstract—Classifiers are among the most widely used supervised machine learning algorithms. Many classification models exist, and choosing the right one for a given task is difficult. During model selection and debugging, data scientists need to assess classifier performance, evaluate the training behavior over time, and compare different models. Typically, this analysis is based on single-number performance measures such as accuracy. A more detailed evaluation of classifiers is possible by inspecting class errors. The confusion matrix is an established way for visualizing these class errors, but it was not designed with temporal or comparative analysis in mind. More generally, established performance analysis systems do not allow a combined temporal and comparative analysis of class-level information. To address this issue, we propose ConfusionFlow, an interactive, comparative visualization tool that combines the benefits of class confusion matrices with the visualization of performance characteristics over time. ConfusionFlow is model-agnostic and can be used to compare performances for different model types, model architectures, and/or training and test datasets. We demonstrate the usefulness of ConfusionFlow in the context of two practical problems: an analysis of the influence of network pruning on model errors, and a case study on instance selection strategies in active learning.

1 INTRODUCTION

CLASSIFICATION is one of the most frequent machine learning (ML) tasks. Many important problems from diverse domains, such as image processing [16], [23], natural language processing [14], [29], [37], or drug target prediction [27], can be framed as classification tasks. Sophisticated models, such as neural networks (NNs), have been proven to be very effective, but building and applying these models is difficult. This is especially the case for multiclass classifiers, which can predict one out of several classes—as opposed to binary classifiers, which can only predict one out of two.

During development of classifiers, data scientists are confronted with a series of challenges. They need to observe how the model performance develops over time, where the notion of time can be twofold: the general workflow in ML development is incremental and iterative, typically consisting of many experiments with different models performed sequentially; and the actual (algorithmic) training of a classifier is itself an optimization problem, involving different model states over time. In the first case, comparative analysis helps the data scientist gauge whether they are on the right track. In the latter case, temporal analysis helps to find the right time to stop the training, such that the model generalizes to unseen samples not represented in the training data.

Model behavior can depend strongly on the choice of hyperparameters, optimizer, or loss function. It is usually not obvious how these choices affect the final overall performance. It is even less obvious how they might affect the behavior on a more detailed level, such as commonly “confused” pairs of classes. However, knowledge about the class-wise performance of models can help data scientists make more informed decisions.

To cope with these challenges, data scientists employ three different types of approaches. One, they look at temporal line charts of single value performance measures such as accuracy. This approach is suitable for comparing learning behavior, but it inherently lacks information at the more detailed class level. Two, they use tools for comparison of classifier performance. However, these tools typically suffer from the same lack of class-level information, or they are not particularly suited for temporal analysis. Three, data scientists assess class-level performance from the class-confusion matrix [38]. Unfortunately, the rigid layout of the classic confusion matrix does not lend itself well to model comparison or temporal analysis.

So far, it has not been possible to simultaneously analyze classifiers from a temporal, comparative, and class-level point of view. Data scientists could benefit from such a holistic assessment of classifier performance by gaining insights from all three points of view without having to switch between tools. Especially for training NNs, additional feedback can help data scientists interpret the model performance, facilitate navigating the space of possible adaptions of the model, and improve understanding of the interaction of the model and underlying data.

We present ConfusionFlow, a visualization that assists data scientists in assessing classifier performance, enabling temporal, comparative, and class-level performance analysis at the same time. To this end, ConfusionFlow introduces an interactive, temporal adaptation of the classic confusion matrix.

We showcase how ConfusionFlow can be used for analyzing different neural network pruning strategies, and
that it is flexible enough to enable analysis of instance selection strategies in active learning. These two examples show how ConfusionFlow helps data scientists understand how classification errors change between training iterations, and how the errors are distributed over individual classes and pairs of classes.

2 BACKGROUND

2.1 Classification

Training classifiers generates a lot of data, from which several performance measures can be derived. Consider a classification problem with a training dataset X consisting of instances \( \{x_1, \ldots, x_N\} \), for which ground truth labels \( g_i \in \Gamma \) are available. The set of different classes \( \Gamma = \{\gamma_1, \ldots, \gamma_K\} \) is called the class alphabet. A classifier model \( c(\theta) \) with parameters \( \theta \) is trained on X such that the predicted class labels \( c(\theta)(x_i) \) best match the ground truth labels \( g_i \). The exact notion of “best match” is specified in the objective or loss function. This function, together with an optimization method, determines how the parameters \( \theta \) are updated at each training step \( t \). In case of neural networks, one training step is called an epoch, and is comprised of passing the whole training data through the network once. Over the course of the training, the classifier progresses through different model states \( c(\theta_t) \), each with different parameters and a different error behavior. The term parameters in this case is not to be confused with the term hyperparameters, which stands for model and optimization settings that have to be chosen before training. We chose the term configuration to refer to the set of model, optimization techniques, hyperparameters, and input data.

2.2 Performance Measures

A typical measure for a classifier’s quality is the accuracy \( A \). It is given by the fraction of correctly classified instances:

\[
A_t = \frac{|\{x_i \mid c(\theta)(x_i) = g_i\}|}{N}.
\]

Other popular aggregate measures are precision and recall, which are built on the binary classification notions of true positives, false positives, and false negatives. For a given class \( \gamma_j \), the number of true positives is given by \( TP_{j,t} = \{|x_i \mid c(\theta_t)(x_i) = \gamma_j \land g_i = \gamma_j\} \), i.e. the number of correctly classified instances of that class. The number of false positives is defined by \( FP_{j,t} = \{|x_i \mid c(\theta_t)(x_i) = \gamma_j \land g_i \neq \gamma_j\} \), i.e. the number of instances classified as \( \gamma_j \) that actually belong to another class. Finally, the number of false negatives is given by \( FN_{j,t} = \{|x_i \mid c(\theta_t)(x_i) \neq \gamma_j \land g_i = \gamma_j\} \), i.e. the number of instances of class \( \gamma_j \) that the classifier wrongly assigned to another class. With these quantities, precision and recall can be defined for each class \( \gamma_j \) by:

\[
\text{Precision}_{j,t} = \frac{TP_{j,t}}{TP_{j,t} + FP_{j,t}}, \quad \text{Recall}_{j,t} = \frac{TP_{j,t}}{TP_{j,t} + FN_{j,t}}.
\]

The harmonic mean of precision and recall is called \( F_1 \)-score.

An extension of individual class-wise performance measures is the confusion matrix \( M \), which lists confusion counts for all pairs of classes:

\[
M_{ij,t} = |\{x_i \mid g_i = \gamma_i \land c(\theta_t)(x_i) = \gamma_j\}|.
\]

This means that entry \((i, j)\) in \( M \) is the number of instances with ground truth class label \( \gamma_i \) that the classifier assigned to class \( \gamma_j \).

Figure 1 schematically shows the relationship between the confusion matrix and precision and recall.

3 PROBLEM SPACE CHARACTERIZATION

The development of new classification models or the adaptation of an existing model to a new problem are highly experimental tasks. Classifiers based on multiple different models need to be trained and evaluated in an iterative workflow. A user is confronted with many different design decisions, such as choosing an architecture, a suitable optimization method, and a set of hyperparameters. All of these choices severely affect the learning behavior and ultimately the quality of the final classifier. This design process requires the user to compare the learning behavior and performances for different models or configurations across multiple training iterations, for instance by inspecting the previously discussed performance measures. Additionally, the quality of the final classifier depends on the quality of the dataset used for learning. The co-evaluation of model and data can be performed on different levels of detail.

3.1 Analysis Tasks

In their interrogative survey on visual analytics in deep learning, Holman et al. [18] define three possible target groups for visualizations in machine learning: model developers, model users, and non-experts. ConfusionFlow is aimed predominantly at the model developer and model user groups. Based on our literature survey (Section 4) and discussion with our collaborators, we have defined two user tasks that play an important role in the process of evaluating classifier performance:

T1 Comparison of (final) classifier performance—Especially for model users, comparing the performance of final classifiers is vital for making the optimal choices for their application. The classifiers to be compared during the selection process can come from entirely different models, or they can result from experimenting with different hyperparameter settings for the same model. Model users as well as developers might also be interested in comparing how the same model performs on different datasets. A visualization tool needs to be model-agnostic to allow this broad notion of comparison.

T2 Temporal analysis of learning behavior—Access to temporal information is extremely important for both user groups,
as they need to judge whether models are capable of learning at all for a given task. Furthermore, a model’s error behavior can change drastically between different model states. In order to cope with the problems of over- or underfitting of the training data, classification model developers and users need to be able to evaluate how their model performs throughout the different iterations.

The two tasks, comparison (T1) and temporal analysis (T2), are illustrated in Figure 2. This schematic shows the most general case, i.e. when both the models (C ⇔ D) and the datasets (X ⇔ Y) are different, and the datasets additionally change over time (X_i ≠ X_j). Many specialized comparison and/or temporal analysis tasks can be derived from the general case depicted in Figure 2, when either the dataset or the model are kept constant:

- In the simple case of observing how a single model is trained on a constant data set (X_i = X for all t), the user is interested only in the sequence C(θ_t)(X) ⇔ ··· ⇔ C(θ_T)(X).
- For comparing the final performances of two classification models, C and D, acting on the same test set X, the user analyzes the pair C(θ_T)(X) ⇔ D(ζ_T)(X).
- Often, the performances of a classifier on two different datasets, such as training and test set, need to be compared temporally. This scenario implies C = D, and X_i = X and Y_i = Y for all t, but X ≠ Y. The user now needs to compare the two sequences C(θ_t)(X) ⇔ ··· ⇔ C(θ_T)(X) and C(θ_t)(Y) ⇔ ··· ⇔ C(θ_T)(Y).
- A more complex example is the comparison of two classifiers during active learning (see Section 6.2). In this case, both models are trained on the same dataset, but the dataset changes over time. The user then compares the sequence C(θ_t)(X_t) ⇔ ··· ⇔ C(θ_T)(X_T) with the sequence D(ζ_t)(X_t) ⇔ ··· ⇔ D(ζ_T)(X_T).

As Figure 2 and the above examples show, comparison and temporal analysis are orthogonal axes in the problem space. A more nuanced evaluation of classifiers requires performing both tasks, T1 and T2, at the same time. On top of this, each task can be performed on several levels of detail, as outlined below.

### 3.2 Analysis Granularity

We propose that the tasks of comparison (T1) and temporal analysis (T2) can be carried out on three different levels of detail (see Figure 3, left):

**L1 Global level**—At the global level, the classifier’s performance is judged by aggregate scores that do not take into account possible differences between classes or individual instances. These measures are typically single numbers such as the accuracy. For showing trends across multiple training iterations, they can be represented in simple line charts.

**L2 Class level**—Typical performance measures at the class level are class-wise accuracy, precision, or recall. Like for the global level, the temporal evolution of these measures throughout training can be visualized as line charts. More detailed class-level information is contained in the confusion matrix. This paper addresses the problem of visualizing the confusion matrix across multiple training iterations.

**L3 Instance level**—At the instance level, access to individual ground truth labels and predicted labels (or predicted class probabilities) allows picking out problematic input data. Strategies for how to further analyze these instances vary strongly between different models and data types. Depending on the specific problem, interesting information may be contained in input images or feature vectors, network outputs, neuron activations, and/or more advanced concepts such as saliency maps calculated by backpropagation [35].

These three levels can be seen as different aggregation levels of individual instance predictions. Figure 3 (right) shows schematically how data at these levels can be visualized across iterations to enable analysis of the training progress.

ConfusionFlow combines line charts of per-class aggregate scores with a visualization of the confusion matrix across training iterations. ConfusionFlow thus operates on the second of the three levels (L2). We consider ConfusionFlow as one step in the pipeline of inspecting/debugging classifiers together with the training data.

### 4 Related Work

The recent resurgence of ML in general and the increasing popularity of deep learning in particular have led to an increased demand for ML development and monitoring tools, but also to an increased desire to better understand existing techniques. This interplay between algorithm design on one hand, and the challenge of making ML algorithms explainable or interpretable on the other hand, has spawned high activity in the field of visualization for ML. The interrogative survey by Hohman et al. [18] gives a comprehensive overview of these recent advances.

In the following, we will discuss approaches that (1) target the user task of comparison across models and/or configurations (see task T1); (2) enable a temporal analysis of the training behavior (see task T2); and/or (3) operate on the class level (L2) as defined in our contextualization in Section 3.2. Table 1 summarizes which of these three aspects is covered by each publication.
Fig. 3. A classifier’s performance can be evaluated at three different levels of detail: global aggregate scores (L1); class-conditional scores and class confusion (L2); and detailed, instance-wise information (L3). ConfusionFlow operates on the class level L2, concentrating on the temporal analysis of training behavior.

Our literature survey will show that the combination of tasks T1 and T2 on the class level L2 is not yet properly addressed by existing tools.

4.1 Model Comparison

One of the most well-known visualization systems for developing, debugging, and evaluating neural networks is TensorFlow’s TensorBoard by Abadi et al. [1], [44]. It combines a visualization of the computation graph with a display of various performance metrics, but it is not designed for comparing multiple ML models in the same view.

For certain types of model architecture, tools with specialized comparison features have been developed: RNNVis by Ming et al. [28] for recurrent neural networks; GANViz by Wang et al. [43] for generative adversarial networks; and CNNComparator by Zeng et al. [47] for convolutional neural networks. RNNVis features a main view with a glyph based sentence visualization. On demand, two models can be compared side by side. GANViz focuses on the comparison of the outputs of the generative network with those of the discriminative network that together make up the GAN. CNNComparator consists of a histogram of parameter values for a selected network layer, a matrix visualization of the convolution operation, as well as an instance-level side-by-side comparison of two selected networks’ performances. It allows comparison of two different configurations or of two different model states for the same configuration, but does not feature immediate access to class confusion measures. ShapeShop by Hohman et al. [17] is aimed at non-experts, and enables comparison of the performances of convolutional neural networks. It is designed to support the user in understanding the basics of what the network learns, rather than provide in-depth evaluation functionality.

Zhang et al. presented Manifold [48], a model-agnostic framework for interpreting, comparing, and diagnosing machine learning models. Small multiples of scatter plots visualize how two different models generate different class outputs. Color coding gives a sense of each model’s class confusion, but there is no option to track the models’ training behaviors.

Comparison of two models cannot only be used to select which model performs better on its own. It can also be part of a workflow to construct new ensemble models or adapt models interactively. In van den Elzen’s and van Wijk’s BaobabView [41], decision trees can be pruned interactively, and the performances of the resulting tree can be compared to the initial one, e.g., by looking at the confusion matrices. EnsembleMatrix by Talbot et al. [40] displays confusion matrices for different classifiers, allowing the user to construct a weighted combination from among them. The resulting ensemble model can be evaluated, again in terms of class confusion.

Each of these techniques enables the user to compare the performance of individual model states in some way (addressing task T1), but misses either the temporal aspect (T2), or does not yield class confusion information (L2).

4.2 Temporal Analysis of Training

Some of the tools listed in the previous section on comparison also let the user explore, to a certain extent, the progression of the model behavior throughout the training. TensorBoard [1] and GANViz [43] augment their main visualization with line charts of accuracy or other performance scores. Similarly, Chung et al. [12] show training statistics in an extra window of their ReVACNN system for real-time analysis of convolutional neural networks. In CNNComparator [47], limited temporal information is accessible by comparing two model states from different training epochs.

DeepEyes by Pezzotti et al. [31] is a progressive visualization tool that combines curves for loss and accuracy with perplexity histograms and activation maps. Progressive line charts of loss during the training are also used in educational tools for interactive exploration, such as TensorFlow Playground [36] or GAN Lab [20].

DeepTracker by Liu et al. [25] displays performance data in a cube-style visualization, where training epochs progress along one of the three axes. A different approach to enable inspection of the training behavior is a selector or slider that allows accessing individual iterations, and that is linked to a main visualization or multiple visualizations in a dashboard. Chae et al. [11] made use of this technique in their visualization of classification outcomes, as did Wang et al. in DQNViz [42], a tool for understanding Deep Q-networks. In
one of the views in Bruckner’s ML-o-scope [10] an epoch slider is tied to a confusion matrix, in which cells can be interactively augmented with example instances. The Blocks system by Alsallakh et al. [3] also features a confusion matrix bound to an epoch slider. Blocks supports investigation of a potential class hierarchy learned by neural networks, which requires the visualization to be scalable to many classes.

Of all the tools for exploration of the training behavior (T2) mentioned above, none focuses on class confusion (L2) while also providing comparison functionality (T1).

### 4.3 Class Confusion

When evaluating the output of classifiers at level L2, class confusion can be interpreted in two ways. Typically, it describes the aggregate scores used in the individual cells of the confusion matrix. However, the term “between-class confusion” is sometimes also used to describe high probability values for more than one class in a classifier’s output for an individual instance. In order to avoid ambiguity, we will call this notion “per-instance classification uncertainty” in our discussion.

Of the works mentioned so far, BaobabView [41], EnsembleMatrix [40], ML-o-scope [10], and Blocks [3] all allow, at least partially, performance analysis on the class level (L2). In these tools, this is realized by visualizing class confusion in terms of standard confusion matrices, either for the final classifier or for one training step at a time.

The confusion matrix is also the heart of the ManiMatrix tool by Kapoor et al. [21], where it is used to interactively modify classification boundaries. This lets the user explore how constraining the confusion for one pair of classes affects the other pairs, aiming at class-level model optimization and interpretability.

Next to the confusion matrix, some alternative ways of evaluating classifier performance on level L2 have been proposed. Alsallakh et al. introduced the confusion wheel [2].

It consists of a circular chord diagram, in which pairs of classes with high confusion counts are connected with thicker chords. On the outside, ring charts encode FN, FP, TP, and TN distributions for each class. Squares by Ren et al. [33] is focused on visualizing per-instance classification uncertainty. Histograms of prediction scores can be unfolded to access individual instances, whose predictions are then encoded using parallel coordinates. Additionally, sparklines for each class give an impression of aggregate class confusion. Squares allows hybrid-level (L2 and L3) confusion analysis.

None of the existing tools for class-level performance analysis (L2) provide an immediate, temporal representation of the training behavior (T2), and most are relatively ill-suited for comparison tasks (T1).

### 5 CONFUSIONFLOW TECHNIQUE

The ConfusionFlow interface consists of three views, as illustrated in Figure 4: (A) the ConfusionFlow matrix presenting the confusion of one or more classifier(s) over time; (B) the class performance and distribution view, including plots of precision, recall, and F1-score, as well as visualizations of the instances’ class distributions; and (C) the detail view showing magnified plots of interactively selected confusion or performance curves. Additionally, ConfusionFlow features (D) a timeline for selecting the range of training steps that are used for exploration; and (E) an input field for loading datasets, which also serves as a legend for the whole visualization.

Figure 4 shows how ConfusionFlow can be used to compare the image classification performance of a neural network on different datasets. For this example, we have loaded confusion data for a neural network image classifier trained on the training set (T1) of CIFAR-10 [22], and evaluated on the images from the corresponding test set (T2), as well as on a recently proposed new test set (T3) from CIFAR-10.1 [32], respectively.

#### 5.1 ConfusionFlow Matrix

The ConfusionFlow matrix, shown in Figures 4.A and 5, is a visualization of classification errors that supports the two tasks of model comparison (T1) and temporal analysis (T2). While the classic confusion matrix described in Section 2.2 is limited to showing confusion data for a single model at one specific time step, the ConfusionFlow matrix visualizes the class confusion for multiple classifiers over multiple training steps (see Figure 3). As described in Section 3.1, the different classifiers can come from experiments with different models or experiments with different training/test datasets. The ConfusionFlow matrix is a small multiples approach: for each cell, the single value of the classic confusion matrix is replaced with a plot of the values for a selected time range (see Figure 5).

The ConfusionFlow matrix should enable temporal analysis and comparison at the same time, while conserving the familiar layout of the confusion matrix. This means that the confusion visualization for each classification model should only take up few space, but should still be able to show a fine-grained temporal resolution. Just as well, individual temporal progressions for different models should be easily distinguishable. Accordingly, we chose the heatmap idiom.
Fig. 4. The ConfusionFlow matrix (A) visualizes confusion of classifiers across training iterations. Performance data for multiple classifiers can be loaded (E) and compared with each other. Additionally, class-wise performance measures and class distributions are displayed in a second view (B). The timeline (D) allows interactive exploration and selection of temporal regions of interest. On demand, plots can be expanded to the detail view (C).

Here, we compare the performance of a neural network classifying images from the train set and test set of CIFAR-10 [22], and a recently proposed new test set from CIFAR-10.1 [32], respectively. The line chart (C) shows that the relative number of misclassified images for the selected classes \textit{auto} and \textit{truck} deviates notably between the original and the new test set. For the remaining classes the classifier performs similarly on the new test set and the original CIFAR-10 test set.

Users can interactively switch from the heatmap to a line chart encoding of class confusion. This option is available for two reasons. First, we found that ML users are particularly used to viewing temporal performance measures as line charts. Second, the line charts can facilitate comparison of absolute values, either across individual epochs of interest (T2) or between models (T1). After switching to the line chart encoding, plots of multiple classifiers are shown as superimposed lines. In addition to the lines, the class confusions for the currently selected iteration are encoded as a heatmap in the background of each cell (see Figure 5, right). This increases the visual saliency of problematic class-pairs, which is inherent in the heatmap encoding. The heatmap/line chart switch, as well as several other controls for visual encoding options, can be seen in Figure 4 to the left of the ConfusionFlow matrix.

To facilitate model comparison across cells for a given predicted class, time by default progresses left-to-right, regardless of the encoding choice. This choice lines up with most users’ expectations for plots of time series data. On demand, users can rotate the cell contents by 90° for easier comparison along a given ground truth class, if the heatmap encoding is selected.

The diagonal elements of the classic confusion matrix list the numbers of correctly classified instances for each class.
For usable classifiers, these numbers are typically much higher than the confusion counts. To keep the user’s focus on the exploration of the error behavior, we decided to replace the diagonal cells in the ConfusionFlow matrix by class labels. In this way, we retain high visual contrast in the off-diagonal cells, and facilitate navigating through the matrix.

To let users assess the overall performance of individual classes, we show temporal plots of false negatives for each class in an additional column to the right, slightly offset from the matrix. Likewise, an additional row at the bottom shows the false positives for each class. We use the diagonal element at the intersection of the additional row and column to show the temporal progression of the overall accuracy of the classifier(s). This allows the user to keep sight of the global performance (level L1).

To enable performance comparison between datasets of different sizes, such as training versus test sets, ConfusionFlow includes an option to switch from absolute to relative performance values. To obtain the relative performance values, the confusion counts are simply divided by the total number of classified instances.

High confusion values for particularly problematic pairs of classes can sometimes hide potentially interesting findings in other cells. To address this issue, we let users toggle between linear and logarithmic scales for the plots. Using an exponential scaling slider, lower values can be further accentuated, which essentially adjusts the contrast in case of the heatmap visualization.

If users are only interested in a subset of the class alphabet, they can narrow down the number of selected classes in a class selection dialog. In order to maintain a meaningful confusion matrix, a minimum of two classes need to be selected at all times. In terms of implementation, the number of displayed classes is not limited, but there are some practical limitations as discussed in Section 7.1.

In case of the CIFAR-10 example shown in Figure 4, the ConfusionFlow matrix reveals that the confusion between classes auto and truck is considerably higher for the CIFAR-10.1 test set. Due to ConfusionFlow’s focus on temporal analysis, it is immediately visible that this error is consistent across all training epochs. For all other pairs of classes, the network generalizes well to the new dataset. In those cells, this can be seen by the similar brightness values for all three selected datasets. Without these class-level observations, the reason for the decrease in overall accuracy would remain unclear; by performing a comparative analysis with ConfusionFlow, the performance decrease could be traced back to changes in the underlying data distribution.

### 5.2 Class Performance & Distribution View

A thorough, class-level (L2) analysis of a classifier’s performance should not only focus on pairs of classes, but also include assessment of the general performance for individual classes.

To this end, ConfusionFlow provides temporal (T2) line charts of precision, recall, and F1-score for all selected classes (see Figure 4.B). In contrast to the ConfusionFlow matrix, horizontal space is not as limited for these plots, and visual clutter is not an issue even in case of noisy data. For comparison between multiple classifiers (T1), we superimpose the line charts. Again hue is used to differentiate multiple classifier results visually, and the hues are consistent with those chosen for the ConfusionFlow matrix. To let users assess the size and class distributions for each dataset, bar charts encode the number of instances per class. The bars are placed next to the per-class performance metrics and again colored consistently with all other views.

In case of the CIFAR example (Figure 4), the class distribution charts reveal that the updated test set from CIFAR-10.1 is considerably smaller than that from CIFAR-10. The precision, recall, and F1-score charts confirm that the recall for class auto and the precision for class truck are particularly bad for the new test set, but they also suggest that the classifier could not generalize well for plane instances from CIFAR-10.1. The class performance charts can thus lead the user to re-examining the ConfusionFlow matrix.

### 5.3 Detail View

All cells in the ConfusionFlow matrix, as well as all per-class performance charts can be selected to be shown in greater detail in a separate view (see Figure 4.C). We visualize the temporal development of selected cells as line charts and superimpose the curves for multiple classifiers, keeping the hue for each model/configuration consistent. The detail view is necessary for in-depth comparison and temporal analysis, as space in the ConfusionFlow matrix is rather limited. Upon loading a new performance dataset, by default the overall (global) accuracy is shown in the detail view, as users are accustomed to this plot from many other performance analysis tools.

For the CIFAR example, the detail view confirms that the confusion value of interest (auto vs. truck) is indeed consistently higher for the updated test set across all epochs.

### 5.4 Timeline

ConfusionFlow should aid users in exploring the error behaviour of classifiers at different temporal granularities. This is enabled by the timeline shown in Figure 4.D. By default, all steps are selected upon loading performance data. Users can select a subset of iterations by dragging the left and right boundaries of the highlighted range in the timeline. All linked views, such as the ConfusionFlow matrix or the detail view, are updated automatically according to the selection. To support users in locating and comparing values for a specific time step across multiple views, they can additionally select a single training step by clicking the iteration number below the range selector. A black line in the timeline marks the currently selected single iteration. The selected iteration is then dynamically highlighted by a vertical marker in all linked components. If the line chart encoding is selected for the ConfusionFlow matrix, the background heatmap will also be updated as described in Section 5.1.

In the CIFAR example from Figure 4, the performance data spans 50 epochs. The user has selected an epoch range covering epochs 0 to 42, and found an interesting peak for the confusion auto vs. truck at epoch 22 in the detail view.
5.5 Dataset Selection
As mentioned above, a unique hue is automatically assigned to each classifier upon loading the respective performance data. An input field with dynamic drop-down suggestions lets the user select from preloaded performance data for a number of classifiers (see Figure 4.E). After the user made the selection, the input field serves as a legend for the visualization, representing each classifier with a colored box.

Since ConfusionFlow is a model-agnostic visualization technique, the kind of data on which ConfusionFlow relies does not depend on the nature of the model. During training, only the classifier output for each instance needs to be logged after each iteration, and stored along with the ground truth labels.

In Figure 4, the input field serves as a legend to remind the user that performance data for the training (■) and test set (□) of CIFAR-10 [22], as well as the recently proposed new test set (Δ) from CIFAR-10.1 [32], have been loaded.

5.6 Implementation
ConfusionFlow is a server-client application based on the Caleydo Phovea framework1 and the Flask framework2. The server side is written in Python and the client side is written in TypeScript using D3.js [9]. Along with the application, we provide Python code for logging the required classification data. A deployed prototype of ConfusionFlow is available at https://confusionflow.caleydoapp.org.

6 Usage Scenario & Case Study
To evaluate the usefulness of ConfusionFlow we describe a usage scenario and a case study performed in collaboration with machine learning researchers. For the usage scenario, we applied ConfusionFlow to investigate different strategies for neural network pruning. For the case study, our collaborators used ConfusionFlow for visually comparing labelling strategies in active learning.

6.1 Usage Scenario: Neural Network Pruning
Neural networks are heavily over-parameterized and often contain millions of weights. While many of these parameters might be necessary for the network to learn something about the data during optimization, the final model will typically contain a lot of redundant parameters which might not be relevant for the network’s performance. One technique for removing redundant parameters from a neural network is called pruning. During learning, connections in the network are successively removed (pruned) according to different selection strategies. Successful pruning results in a compressed model that retains its accuracy while requiring fewer computations and less memory. This enables deployment in embedded systems or in other contexts where energy consumption and/or latency are an issue [15]. Furthermore, under certain circumstances pruned networks are able to learn faster and generalize better than the initial dense model [13], [26]. It is an interesting question how removing certain weights will influence the model’s performance. For instance, removing certain weights might impair the model’s ability to distinguish certain classes. Faceting the model errors can provide a better understanding of how different pruning strategies affect the model quality.

For this usage scenario, we examined the performance of several fully connected networks with different architectures trained on the Fashion-MNIST dataset [46]. This dataset consists of grey-scale images of fashion objects organized in 10 classes (trouser, pullover, sandal, etc.). Specifically, we investigated the effects of pruning a neural network and re-initializing the resulting sparse network with the initial weights as described by Frankle and Carbin [13], trying to find out whether the phenomenon they observed also generalizes to uncommon network architectures.

Visual Comparison of Network Pruning Strategies
Figure 6 shows the ConfusionFlow visualization for three different networks trained on the Fashion-MNIST dataset. The original network (■) had 6-layers, each with 200 hidden units and ReLU activation functions. The learning rate was 0.012, with a batch size of 60. In the second network (□), 20 percent of the connections were removed at each epoch (this approach is called online pruning). The sparse network resulting after 15 epochs was then re-initialized with the same weights as the original dense network, and re-trained from scratch (△). In this network less than 4 percent of the original connections remain.

It is immediately obvious from the overall accuracy plot (Figure 6.A) that the training of the original model (■) fails completely after 10 epochs. Training of the online-pruned network (□) fails slightly later. The performance of the re-initialized sparse network (△), however, remains high. Remarkably, it even performs better than the other two networks right from the start. ConfusionFlow allows to relate this global (L1) accuracy improvement to pairwise class confusions.

Inspection of the ConfusionFlow matrix shows that the confusion counts for all shoe-related pairs of classes (sneaker vs. sandal, ankle boot vs. sneaker, etc.) increase considerably during later epochs for the non-pruned and online-pruned networks (Figure 6.B). The re-initialized sparse network, on the other hand, continues to learn to better distinguish between these classes. Another reason for the complete failure of the original network seems to be related to the classes trouser and coat (see Figure 6.C), with extremely high FP values for these two classes in two of the later epochs.

Even though the global accuracy plot showed a pronounced accuracy drop for the non-pruned and the online-pruned networks, both models retain an accuracy of about 30 percent. The ConfusionFlow matrix reveals that this remaining accuracy in the later epochs is most likely related to the pairs shirt vs. T-shirt/top and pullover vs. shirt (Figure 6.D). The better generalization of the re-initialized sparse network across many other pairs of classes comes at the cost of slightly worse performance for the upper body garments.

These findings demonstrate that ConfusionFlow allows a more nuanced evaluation of classifier performance, enabling the user to trace accuracy changes back to the class level.

1. Caleydo Phovea: https://github.com/phovea
2. Flask: http://flask.pocoo.org
6.2 Case Study: Effective Labeling in Active Learning

For virtually any supervised ML task, labeled (ground truth) data for training and testing ML models is an essential prerequisite. The labeling process requires human supervision, as humans need to attach “semantic” information to data instances. Subsequently, labels and associated instances can be used as training data, e.g., to train a classifier with categorical labels. Given a small but representative set of labeled data that already yields high classifier accuracy, the remaining labeling process can often be automated.

Hence, a central challenge in the early labeling process is the identification of meaningful instances to be labeled by humans next.

In the ML field, active learning [34] is a class of model-driven approaches that aims to support the meaningful selection of instances in this incremental labeling process: humans are asked for labels for the instances selected by algorithmic models. Visual-Interactive Labeling (VIAL) [8] is a concept that aims to combine the strengths of humans and algorithmic models to facilitate effective instance selection and labeling. To this end, VIAL combines active learning (i.e., ML) with visual-interactive interfaces for the exploration, selection, and labeling of instances [6].

Our collaborators conduct research about human factors in instance selection strategies for labeling. They aim at making the labeling process a more effective, efficient, and human-friendly endeavor. In a recent study, they observed users during the labeling process and identified that labeling strategies applied by humans (with visual interfaces) and traditional model-centered active learning approaches have complementary strengths [7]. One of the strategies that is frequently applied by humans refers to Dense Areas First, reflecting that humans tend to select instances in the dense areas of the data. ConfusionFlow allows our collaborators to compare model-based active learning strategies (e.g., Smallest Margin [45]) with human-based strategies (e.g., Dense Areas First), and so-called quasi-optimal selection strategies used for upper-limit performance comparison (e.g., Greedy Labeling trials) [4]. For the experiments, our collaborators decided to use the well-established MNIST data set [24] consisting of images of handwritten digits.

Visual Comparison of Instance Selection Strategies

The visualization of the performances of these three strategies—Smallest Margin, Dense Areas First, and Greedy Labeling —, at the granularity of individual classes and class confusions during the labeling process, is shown in Figure 7. Each training step corresponds to one additional, labeled instance fed to the model.

The Greedy strategy (Figure 7, ■) has the best overall accuracy from the start (A1). This method requires the ground truth information, and is supposed to depict the theoretical upper-limit of performance. With only 50 labels the Greedy strategy already achieves almost 80 percent accuracy. Our collaborators identified a known pattern in the accuracy curve, which happens after ten instances (i.e., when all labels have been visited exactly once; see Figure 7.A2). With the eleventh label the training set becomes unbalanced, leading to a significant temporal decrease of classification accuracy. The effect takes place a second time after the 20th iteration, successively. Taking the analysis to a more fine-grained level, our collaborators identified that the described anomaly pattern is related to the classes 0, 4, and 9. Future repetitions of the experiment will clarify whether this effect...
Fig. 7. Visual comparison of strategies for effective labeling. In an experiment, our collaborators tested three different labeling strategies: Greedy (●), Smallest Margin (■), and Dense Areas First (■). Using ConfusionFlow, our collaborators made a series of findings regarding the overall performances (A, D1, D2) as well as the temporal progression of class confusions (B, C, D3) for the different strategies.

can be related to the semantics of the particular classes or can be explained by other influencing factors.

The Smallest Margin strategy (Figure 7, ■) starts with a peak (instances 3 to 6) until the performance decreases considerably. With instance 50, the Margin strategy has almost 60 percent accuracy. In the matrix, our collaborators identified considerably high confusion values of class 8 with almost any remaining class (Figure 7.B1). This poor performance for class 8 is also clearly visible in the precision curve. An interesting pattern was the significant decrease of confusion between classes 0 vs. 8, roughly beginning at the 35th time step (B2). It seems that sometimes a single label can make a difference and support the accuracy of a classifier. Additionally, up to around instance 50, confusion values for class 2 are relatively high, leading to many false positives for this class (B3).

The Dense Areas First strategy (Figure 7, ■) exhibits a small but steady increase in the early phase of the labeling process. After 50 labels, the strategy achieves almost 55 percent accuracy. By inspecting the ConfusionFlow matrix, the analysts gained a series of different insights. Some class confusions lasted for the entire labeling process (1 vs. 3, 5 vs. 3, 2 vs. 6; see Figure 7.C1). Conversely, some class confusions seemed to have different levels of intensity during the labeling process (2 vs. 4, 7 vs. 4, 9 vs. 4). One class confusion even increased during the process (7 vs. 9), visible both in the matrix and in the FP chart (C2). Some training steps introduced peaks of confusion (class 6, roughly at instance 10). Finally, some classes did not suffer from considerable confusions at all (0, 1, 5, and 7).

While our collaborators made a series of observations about the three individual strategies by performing a temporal analysis with ConfusionFlow (task T2), the visual comparison features, in addition, helped them to identify similarities and differences between strategies (combination of tasks T2 + T1).

As expected, the Greedy strategy performed best, leading with a significant margin compared to the remaining strategies (see Figure 7.D1). The model-based Smallest Margin and the human-based Dense Areas First strategy performed similarly: the collaborators were not able to pick a winning selection strategy from among these two (D2). According to the analysts, one of the most interesting findings at the granularity of class confusions was that for some pairs of classes the class confusion patterns differed considerably over time. In case of the classes 9 vs. 4, for example, the confusions of the model-based Smallest Margin and the human-based Dense Areas First strategy showed strongly contrasting behavior (see Figure 7.D3). The general observation that class confusions over time can be very different for different strategies indicates that different strategies have complementary strengths. This encouraged our collaborators to further investigate such effects at the granularity of classes and class confusions. These different strengths also became apparent from the FP and FN plots, which our collaborators did not expect to differ that strongly.

6.3 Summary

The pruning usage scenario and the case study on labeling strategies in active learning both highlighted the importance of being able to perform model comparison (T1) and temporal analysis of learning (T2) at the same time. In both contexts ConfusionFlow allowed the users to trace back changes in the overall accuracy to the confusion counts for individual pairs of classes (L2). This way, our visualization approach can help
answer questions such as whether the model errors change uniformly over all classes, whether they are limited to certain difficult pairs, or how these errors change between training iterations. It also became obvious that ConfusionFlow should not be the last step in the workflow for analyzing classifier performance. Our collaborators mentioned that they would be particularly interested in using their findings from ConfusionFlow as a starting point for drilling further down to the instance level. We also gained important feedback about some other limitations of ConfusionFlow, which we will discuss in the following section.

7 DISCUSSION

7.1 Scalability of the ConfusionFlow Matrix

The ConfusionFlow matrix supports up to around fifteen classes. Due to the matrix layout, the cell size shrinks with an increasing number of classes. The maximum number of classes also depends on the number of selected classifiers and the number of selected training iterations. For instance, evaluating the performance across multiple epochs might still be possible in case of twenty classes for a single model configuration, while comparison of two or more configurations would already reduce the size for each individual heatmap too much. As the data density in the heatmaps increases (with increasing number of classes and/or selected time steps), the limited space in a ConfusionFlow matrix cell can lead to sampling problems. For the current implementation, we chose not to reduce the temporal resolution by means of interpolation or sub-sampling, because this can obscure interesting patterns or events in the data.

In case of applying ConfusionFlow to datasets with notably more classes (e.g., ImageNet with 1,000 non-overlapping classes), a pre-processing step consisting of class sub-sampling or class aggregation is necessary. We argue that these large datasets are rare, because they are expensive to obtain.

7.2 Visual Clutter

Our collaborators pointed out that overplotting can be an issue in the FP and FN charts. Currently, each of these charts shows $K \times d$ lines, where $K$ is the number of selected classes, and $d$ is the number of performance datasets loaded for comparison. Already for quite few classes and/or loaded models this leads to visual clutter, an issue that sometimes persists even in the detail view. In the current implementation, hovering over a cell in the row/column for which the FN/FP plot is selected highlights the respective line. In a future version, we plan to add an option to reduce the number of lines to $d$ by aggregating over classes.

So far, we have chosen to limit the FN and FP plots to line charts, in order to visually separate them from the matrix. We also wanted to appeal to the habits of most ML researchers, who are used to line charts for temporal plots of performance metrics. Based on the feedback, we may add an option to display the FP and FN data as stacked heatmaps (lasagna plots), just like the confusion matrix cells. For the plots of precision, recall, and $F_1$-score, more horizontal space is available, and the line chart encoding worked well.

7.3 Comparison of Class Distributions

Currently, the bar charts encoding the number of class members in each dataset are aligned such that comparison of multiple datasets for a given class is facilitated. However, judging the class distribution of a single dataset, i.e., comparing the number of members of one class to that of another, is difficult. An additional option to rotate the bars by 90° (and left-align them) could solve this issue.

7.4 Multi-Label Classification

Multi-label classification is a machine learning problem, in which multiple class labels can be assigned to instances. In principle, ConfusionFlow can be extended to visualize the confusion of multi-label classifiers, by using combinations of class labels instead of single classes along each of the matrix axes. However, since the number of possible combinations grows quickly with the number of overall classes, aggregation methods would need to be incorporated in the workflow. Thus, in case of multi-label classification, an instance-level approach seems more appropriate.

7.5 Instance-Level Analysis

Since ConfusionFlow was specifically designed to visualize class-level information, it does not feature tools for analyzing the confusion on an instance-based level over the learning process (neither regarding assignment to the wrong class, nor per-instance classification uncertainty). However, exploring the learning dynamics at the level of instances could provide valuable additional insight into the underlying dataset and the classification behavior for individual experiments. The detection of problematic, misclassified instances could potentially help discover labeling errors or outlier instances that might otherwise go unnoticed.

We are currently working on InstanceFlow, a visualization tool for addressing exactly these issues. InstanceFlow will visualize the classification progression of individual instances throughout the training. It will allow users to filter instances by certain metrics, such as the frequency of “hopping” between classes.

Especially for NN classifiers, linking this instance-level class-confusion to a feature-based detail view could further improve the understanding of the learning behavior of the classifier. Depending on the supported model architectures, this detail view could build upon previous work by Olah et al. [30] or work regarding activation visualization [19], [31].

8 CONCLUSION

In this paper we introduced ConfusionFlow, a novel tool for visualizing and exploring the temporal progression of classifier confusion. ConfusionFlow combines a visualization of the confusion matrix over time with charts for global and per-class performance metrics. We evaluated the usefulness of ConfusionFlow’s interactive exploration capabilities by means of a usage scenario in the context of NN pruning and by a case study on the performance of instance selection strategies in active learning.

We would like to point out that we designed ConfusionFlow not as a catch-all, standalone tool, but to be used in combination with other tools and visualization components.
In particular, we plan to complement ConfusionFlow’s class-level information with a novel visualization tool focused on temporal observation of instance-level confusion. However, offering model comparison and temporal training analysis at the class level, ConfusionFlow can fill an important gap in an ML workflow towards understanding and interpreting classification models.

ACKNOWLEDGMENTS

This work was supported in part by the State of Upper Austria (FGF 851460, Human-Interpretable Machine Learning) and the Austrian Science Fund (FWF P27975-NBL).

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv:1603.04467 [cs], Mar. 2016.

[2] B. Alsallakh, A. Hanbury, H. Hauser, S. Miksch, and A. Rauber. Visual Methods for Analyzing Probabilistic Classification Data. IEEE Transactions on Visualization and Computer Graphics, 20(12):1703–1712, Dec. 2014. doi: 10.1109/TVCG.2014.2346660

[3] B. Alsallakh, A. Jourabloo, M. Ye, X. Liu, and L. Ren. Do Convolutional Neural Networks Learn Class Hierarchy? IEEE Transactions on Visualization and Computer Graphics, 24(1):152–162, 2018. doi: 10.1109/TVCG.2017.2744683

[4] J. Bernard, M. Hutter, M. Lehmann, M. Müller, M. Zeppelzauer, and M. Sedlmair. Learning from the Best Visual Analysis of a Quasi-Optimal Data Labeling Strategy. In European Conference on Visualization (EuroVis), p. 5, 2018. doi: 10.2312/eurovisshort.20181085

[5] J. Bernard, M. Hutter, H. Reinemuth, H. Pfeifer, C. Bors, and J. Kohlhammer. Visual-Interactive Preprocessing of Multivariate Time Series Data. Computer Graphics Forum, 2019. doi: 10.1111/cgf.13698

[6] J. Bernard, M. Hutter, M. Zeppelzauer, D. Fellner, and M. Sedlmair. Comparing Visual-Interactive Labeling with Active Learning: An Experimental Study. IEEE Transactions on Visualization and Computer Graphics, 24(1):298–308, Jan. 2018. doi: 10.1109/TVCG.2017.2744818

[7] J. Bernard, M. Zeppelzauer, M. Lehmann, M. Müller, and M. Sedlmair. Towards User-Centered Active Learning Algorithms. Computer Graphics Forum, 37(3):121–132, June 2018. doi: 10.1111/cgf.13406

[8] J. Bernard, M. Zeppelzauer, M. Sedlmair, and W. Aigner. VIAL: a unified process for visual interactive labeling. The Visual Computer, 34(9):1189–1207, Sept. 2018. doi: 10.1007/s00371-018-1500-3

[9] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-Driven Documents. IEEE Transactions on Visualization and Computer Graphics, 17(12):2301–2309, 2011. doi: 10.1109/TVCG.2011.185

[10] D. Bruckner. ML-o-Scope: A Diagnostic Visualization System for Deep Machine Learning Pipelines. Technical report, Defense Technical Information Center, Fort Belvoir, VA, May 2014. doi: 10.21236/ADA605112

[11] J. Chae, S. Gao, A. Ramanthan, C. Steed, and G. D. Tourassi. Visualization for Classification in Deep Neural Networks. In Workshop on Visual Analytics for Deep Learning at IEEE VIS, p. 6, 2017.

[12] S. Chung, S. Suh, and C. Park. ReVACNN: Real-Time Visual Workshops on Interactive Data Exploration and Analytics (IDEA), p. 7, 2016.

[13] J. Frankle and M. Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. arXiv:1803.03635 [cs], Mar. 2018.

[14] X. Glorot, A. Bordes, and Y. Bengio. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In Proceedings of the 28th International Conference on Machine Learning, p. 8, 2011.

[15] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both Weights and Connections for Efficient Neural Networks. arXiv:1506.02626 [cs], June 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. IEEE, Las Vegas, NV, USA, June 2016. doi: 10.1109/CVPR.2016.90

[17] F. Hohman, N. Hodas, and D. H. Chau. ShapeShope: Towards Understanding Deep Learning Representations via Interactive Experimentation. In Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA ’17, pp. 1694–1699. ACM Press, Denver, Colorado, USA, 2017. doi: 10.1145/3027063.3035103

[18] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. arXiv:1801.06889 [stat], Jan. 2018.

[19] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau. ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models. IEEE Transactions on Visualization and Computer Graphics, 24(1):88–97, Jan. 2018. doi: 10.1109/TVCG.2017.2744718

[20] M. Kahng, N. Thorat, D. H. P. Chau, F. B. Viegas, and M. Wattenberg. GAN Lab: Understanding Complex Deep Generative Models using Interactive Visual Experimentation. IEEE Transactions on Visualization and Computer Graphics, 25(1):1–11, Jan. 2019. doi: 10.1109/TVCG.2018.2864500

[21] A. Kapoor, B. Lee, D. Tan, and E. Horvitz. Interactive optimization for steering machine classification. In Proceedings of the 28th international conference on Human factors in computing systems - CHI ’10, p. 1343. ACM Press, Atlanta, Georgia, USA, 2010. doi: 10.1145/1753326.1753529

[22] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical Report No. 4, University of Toronto, 2009.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90, May 2017. doi: 10.1145/3053554

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2282, Nov. 1998. doi: 10.1109/5.726791

[25] Y. Liu, Y. Cui, K. Jiang, Y. Guo, and H. Qu. DeepTracker: Visualizing the Training Process of Convolutional Neural Networks. arXiv:1808.08531 [cs], Aug. 2018.

[26] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the Value of Network Pruning. arXiv:1810.05270 [cs, stat], Oct. 2018.

[27] A. Kapoor, B. Le, J. S. S. van Gemert, B. P. Lelieveldt, E. Eisemann, and M. Viergever. DeepConvNet: Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014), pp. 69–78. Dublin, Ireland, 2014.

[28] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The Building Blocks of Interpretability. Distill, 3(4), Mar. 2018. doi: 10.23915/distill.00010

[29] N. Pezzotti, T. Hollt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. DeepEyes: Progressive Visual Analytics for Designing Deep Neural Networks. IEEE Transactions on Visualization and Computer Graphics, 24(1):98–108, Jan. 2018. doi: 10.1109/TVCG.2017.2744358

[30] B. Settles. Active Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[31] K. Simonanyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv:1312.6034 [cs], Dec. 2013.
[36] D. Smilkov, S. Carter, D. Sculley, F. B. Vargas, and M. Wattenberg. Direct-Manipulation Visualization of Deep Networks. arXiv:1708.03788 [cs, stat], Aug. 2017.

[37] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642, 2013.

[38] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. Information Processing and Management, 45:427–437, 2009. doi: 10.1016/j.ipm.2009.03.002

[39] B. J. Swihart, B. Caffo, B. D. James, M. Strand, B. S. Schwartz, and N. M. Punjabi. Lasagna Plots: A Saucy Alternative to Spaghetti Plots. Epidemiology, 21(5):621–625, Sept. 2010. doi: 10.1097/ede.0b013e3181e5b6a

[40] J. Talbot, B. Lee, A. Kapoor, and D. S. Tan. EnsembleMatrix: Interactive Visualization to Support Machine Learning with Multiple Classifiers. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’09, pp. 1283–1292. ACM, New York, NY, USA, 2009. doi: 10.1145/1518701.1518895

[41] S. van den Elzen and J. J. van Wijk. BaobabView: Interactive construction and analysis of decision trees. In 2011 IEEE Conference on Visual Analytics Science and Technology (VAST), pp. 151–160. IEEE, Providence, RI, USA, Oct. 2011. doi: 10.1109/VAST.2011.6102453

[42] J. Wang, L. Gou, H.-W. Shen, and H. Yang. DQNViz: A Visual Analytics Approach to Understand Deep Q-Networks. IEEE Transactions on Visualization and Computer Graphics, 25(1):288–298, Jan. 2019. doi: 10.1109/TVCG.2018.2864504

[43] J. Wang, L. Gou, H. Yang, and H.-W. Shen. GANViz: A Visual Analytics Approach to Understand the Adversarial Game. IEEE Transactions on Visualization and Computer Graphics, 24(6):1905–1917, June 2018. doi: 10.1109/TVCG.2018.2810223

[44] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mane, D. Fritz, D. Krishnan, F. B. Viegas, and M. Wattenberg. Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow. IEEE Transactions on Visualization and Computer Graphics, 24(1):1–12, Jan. 2018. doi: 10.1109/TVCG.2017.2744878

[45] Y. Wu, I. Kozintsev, J. Bouguet, and C. Dulong. Sampling Strategies for Active Learning in Personal Photo Retrieval. In 2006 IEEE International Conference on Multimedia and Expo, pp. 529–532, July 2006. doi: 10.1109/ICME.2006.262442

[46] H. Xiao, K. Rasuli, and R. Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747 [cs, stat], Aug. 2017.

[47] H. Zeng, H. Haleem, X. Plantaz, N. Cao, and H. Qu. CNNComparator: Comparative Analytics of Convolutional Neural Networks. arXiv:1710.05285 [cs, stat], Oct. 2017.

[48] 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. IEEE Transactions on Visualization and Computer Graphics, 25(1):364–373, Jan. 2019. doi: 10.1109/TVCG.2018.2864499

Andreas Hinterreiter is a PhD student at the Institute of Computer Graphics, Johannes Kepler University (JKU) Linz. His research interests include explainable AI and uncertainty visualization. Starting from October 2019, he is on secondment at the Biomedical Image Analysis Group at Imperial College London. He received his Diplomingenieur (MSc) in Technical Physics from JKU.

Peter Ruch is a Master student in Bioinformatics at the Johannes Kepler University (JKU) Linz. He is interested in the application of machine learning methods, especially neural networks, in real world scenarios, with special focus on the interpretability and transparency of the results.

Holger Stitz is a postdoctoral researcher at the at the Institute of Computer Graphics, Johannes Kepler University (JKU) Linz. His main research interest is multi-attribute time-series data in the context of large networks. For more information see http://holgerslitz.de.

Martin Ennemoser is a researcher and developer at Salesbeat GmbH. His research interests include deep learning, natural language processing and reinforcement learning. He received a Diplomingenieur (MSc) in Computer Science from JKU.

Jürgen Bernard is a postdoctoral research fellow at the University of British Columbia, with the InfoVis Group in the computer science department. His research interests include visual analytics, information visualization, interactive machine learning, and visual-interactive labeling in particular. Bernard received a PhD degree in computer science from TU Darmstadt. He has received the Dirk Bartz Price for Visual Computing in Medicine and the Hugo-Geiger Preis for excellent Dissertations. He is also organizing the Workshop on Visual Analytics in Health Care (VAHC) at IEEE VIS since 2015.

Hendrik Strobelt is a researcher in information visualisation, visual analytics, and machine learning at IBM Research AI, Cambridge. His research interests include visualization of large data sets of unstructured/semi-structured data, biological data, and neural network models. He received his PhD in computer science from Uni Konstanz. For more information see http://hendrik.strobelt.com.

Marc Streit is a Full Professor of Visual Data Science at the Johannes Kepler University Linz. He finished his PhD at the Graz University of Technology in 2011. His scientific areas of interest include visualization, visual analytics, and biological data visualization. He is also co-founder and CEO of datavisyn, a spin-off company developing data visualization solutions for applications in pharmaceutical and biomedical R&D. For more information see http://marc-streit.com.