InstanceFlow: Visualizing the Evolution of Classifier Confusion on the Instance Level

Michael Pühringer*
Johannes Kepler University Linz

Andreas Hinterreiter†
Johannes Kepler University Linz
Imperial College London

Marc Streit‡
Johannes Kepler University Linz

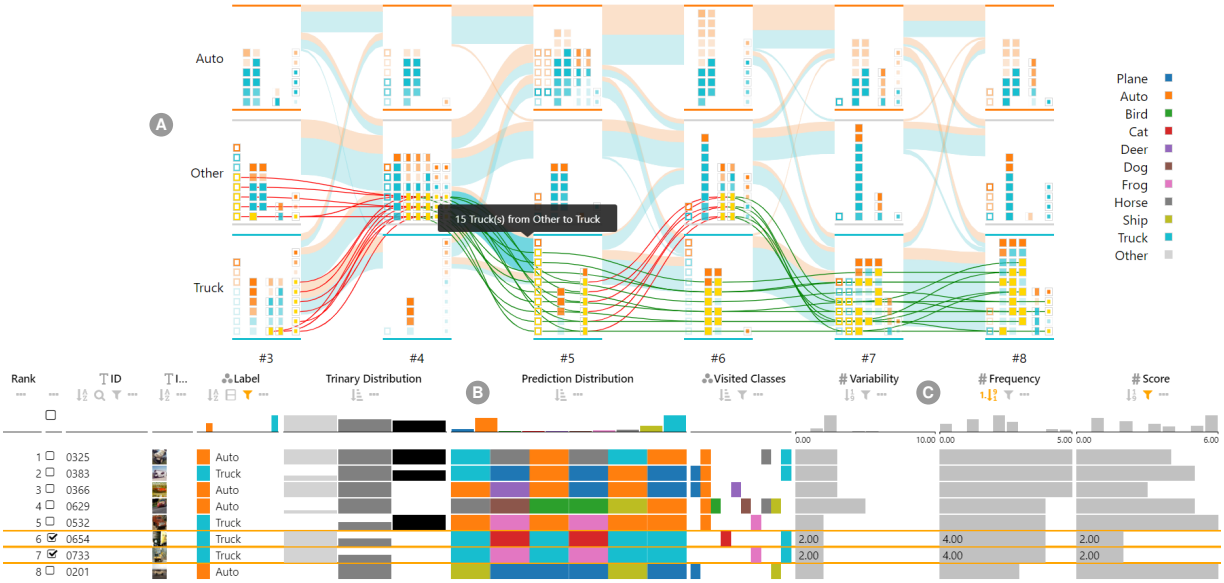

Figure 1: InstanceFlow visualizes the evolution of a classifier's predictions throughout the training process on an instance level. The *Flow View* A shows all instances and their corresponding class association as rectangular glyphs. A Sankey diagram shows the fractions of instances moving between classes. Additionally, the traces of single instances can be highlighted. The *Tabular View* B of the instance predictions over time along with custom performance scores C allows finding, ranking, and grouping instances.

Abstract

Classification is one of the most important supervised machine learning tasks. During the training of a classification model, the training instances are fed to the model multiple times (during multiple epochs) in order to iteratively increase the classification performance. The increasing complexity of models has led to a growing demand for model interpretabilty through visualizations. Existing approaches mostly focus on the visual analysis of the final model performance after training and are often limited to aggregate performance measures. In this paper we introduce InstanceFlow, a novel dual-view visualization tool that allows users to analyze the learning behavior of classifiers over time on the instance-level. A Sankey diagram visualizes the flow of instances throughout epochs, with on-demand detailed glyphs and traces for individual instances. A tabular view allows users to locate interesting instances by ranking and filtering. In this way, InstanceFlow bridges the gap between class-level and instance-level performance evaluation while enabling users to perform a full temporal analysis of the training process.

Keywords: Classification. Performance analysis. Time series visualization.

*E-mail: michipueh@gmail.com
†E-mail: andreas.hinterreiter@jku.at or a.hinterreiter@imperial.ac.uk
‡E-mail: marc.streit@jku.at

1 Introduction

The real-world application of increasingly complex machine learning models has led to a growing interest in visualizations for post-hoc model explainability [2,5,12]. One of the most important supervised machine learning tasks, with a wide variety of application areas, is classification. The performance of classification models can be analyzed and visualized on three levels of detail [11]: global, class-level, and instance-level. Additionally, extending the analysis to cover the whole training process (multiple training iterations, i.e. epochs) has been identified as a promising research direction [5, 12, 18]. However, existing approaches often focus on fully trained models and disregarding the temporal evolution that led to this final model state. Furthermore, tools that enable temporal performance analysis are typically limited to global, single-value performance measures [8].

In previous work, we argue that extending a temporal performance analysis to the class-level can lead to new insights [11]. Still, the aggregated nature of class-level performance measures showed that a full analysis of a classification model's learning behavior also requires drilling down to the level of individual instances.

To address this issue, we introduce InstanceFlow, a novel visualization that combines aggregated temporal information in a Sankey diagram with detailed traces of individually selected instances. These interesting instances can be located via a tabular view that allows users to rank and filter instances by several temporal difficulty measures. With this dual approach, InstanceFlow aims at bridging the gap between class- and instance-level analysis of the learning behaviors of classification models.

2 User Tasks

As stated in the introduction, InstanceFlow focuses on a temporal analysis of instance-level classification performance. Such an analysis can lean either towards exploring instance-based properties of certain *epochs*, or analyzing the temporal characteristics of individual *instances*. Consequently, we have structured the user tasks that we seek to address with InstanceFlow by whether they are epoch- or instance-focused (see Table 1). We based the individual user tasks on a survey of existing instance-level visualizations (Section 3), with a focus on filling gaps related to model-agnosticism and temporal analysis.

The instance-focused tasks (**IT**) are concerned with finding instances which are hard to classify correctly (**IT1**), or whose predictions evolve unusually (**IT2**–**IT4**). This allows users to assess temporally (un)stable weaknesses of the model or detect input data with potentially wrong ground truth labels. The epoch-focused tasks (**ET**) are related to analyzing epoch-wise class distributions **ET1** or locating problematic epochs (**ET2**, **ET3**). Problematic epochs are those for which weight or parameter changes produce a non-beneficial outcome, such as an increase of confusion between two critical classes.

Table 1: User tasks addressed by InstanceFlow, categorized by their focus on epoch- (**ET**) or instance-driven (**IT**) analysis.

Task	Description
IT1	Find *difficult* instances
IT2	Trace an instance's *path* over multiple epochs
IT3	Analyze if an instance *visits many or few* classes
IT4	Find instances *oscillating* between classes
ET1	Assess *class distributions* for a given epoch
ET2	Find momentarily *wrongly and/or correctly* instances
ET3	Find instances *staying* in their class or *moving* between classes at a given epoch

3 Related Work

Previous work on visualizing instance-level information in machine learning has mostly focused on model-dependent parameters such as the activation of neurons in deep neural networks in response to a given input instance. Often, the visualizations are tailored to exploring the behavior of individual layers of the networks [7, 13, 19, 25]. A number of works have focused particularly on the convolutional layers of CNNs [3,17,24]. Similar visualizations exist for GANs [23] and Deep Q-networks [22]. Most of these model-specific approaches are further limited in that they only provide information for a single iteration at a time.

Likewise, visualizations focused on the performance analysis of classifiers typically do not enable a true temporal analysis. Chae at al. [4] show instance-wise predictions and aggregated distributions; Alsallakh et al. [1] focus on class-confusion with basic drill-down functionality to explore problematic instances. In both cases, limited temporality is achieved via single-epoch selection sliders.

Squares by Ren et al. [21] is closely related to our work in terms of visual design and the type of information shown. Users can switch between aggregated prediction distributions and a fine-grained instance-wise visualization using rectangular glyphs. However, in Squares only the final model predictions can be explored.

InstanceFlow aims at enabling a true temporal performance analysis on the instance level. In this regard, it is closely related to our previous work, ConfusionFlow [11], which enables temporal class-level analysis of classification models via a novel adaptation of the confusion matrix.

Visually, InstanceFlow combines a multiform Sankey diagram similar to VisBricks [15] and StratomeX [16] with a sortable, aggregable tabular view (cf. Table Lens [20], LineUp [10], and Taggle [9]).

4 InstanceFlow

The InstanceFlow interface consists of two main components, as illustrated in Figure 1: The *Flow View* (A) shows a Sankey diagram of the model's instance predictions throughout the selected training epochs; the *Tabular View* (B) lists detailed temporal instance information including performance scores (C).

The Flow View supports different levels of granularity. In its basic form, the Flow View visualizes "class changers" in a Sankey diagram. *Distribution Bar Charts* emphasize the fraction of correctly versus incorrectly classified instances. At the finest granularity, *Instance Glyphs* encode each individual sample, with *Instance Traces* connecting the instances to reveal their paths through the epochs.

The Tabular View lists all instances along with their associated predictions over time and allows finding, ranking, and grouping instances via custom instance-level performance measures.

The Flow View and Tabular View are fully linked, such that traced or selected instances are highlighted in both views simultaneously.

4.1 Flow View

The flow visualization can be seen in Figure 2 (top left), where the x-axis denotes the epoch and the y-axis denotes the predicted classes. A Sankey diagram visualizes the Flow, i.e., how many instances move from one class to another in the following epoch. The user selects classes of interest, and each class is assigned to a vertical region in the Sankey diagram. All non-selected classes are aggregated as "Other" and also assigned to a dedicated vertical region. The range of epochs to be visualized can be selected via an epoch slider. Hovering over a section of the Sankey diagram reveals the exact number of instances moving between the corresponding classes. Clicking on a section of the Sankey diagram selects those instances.

Distribution Charts The height of each flow implicitly encodes the distribution of the predictions in each class. To emphasize the distributions at each epoch, they can be explicitly encoded in Distribution Bar Charts placed between the Flow visualizations (see Figure 2, top right).

Instance Glyphs For a more detailed view, the instances themselves can be represented as by rectangular glyphs (see Figure 4, bottom left). The color of the Instance Glyphs denotes the actual class of the instances (e.g., ■ and ■ in Figure 2). Additional information is encoded in their shape, opacity, and position. The shape and horizontal position indicate if the instance predictions are temporally stable (■), changing from a different class (□), leaving

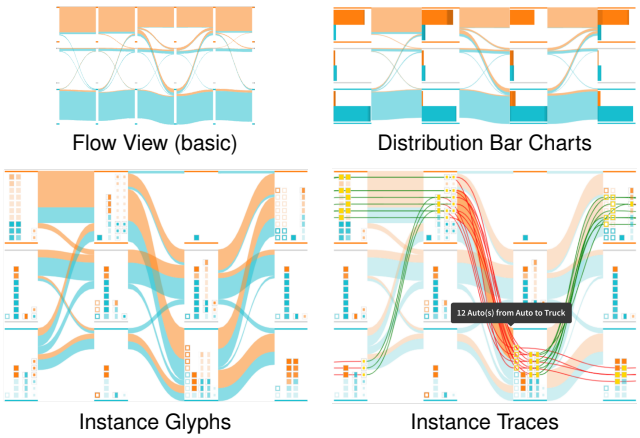

Figure 2: Flow View with all possible extensions. Distribution Bar Charts emphasize the class distribution, Instance Glyphs show the underlying instances, and Instance Traces reveal individual paths through epochs.

for a different class (), or coming from and leaving for different classes (). The opacity and vertical box position encode one of the calculated numerical difficulty measures described in Section 4.2, which visually ranks the instances by the model's performance.

Instance Traces To allow users to track the path of specific instances throughout the training epochs, their traces can be visualized as lines connecting the corresponding instance glyphs (see Figure 4, bottom right). The color of these Instance Traces indicates if the instance is moving to the correct (—) or incorrect (—) class. Instance Traces are only shown for selected instances, i.e., by clicking on an Instance Glyph, a section of the Sankey diagram, or selecting instances from the Tabular View.

4.2 Tabular View

The per-class distribution flow is effective for finding anomalies in the learning process, but recognizing specific instances can be hard due to the high information density. To facilitate the tasks of identifying problematic instances (**IT1**–**IT4**), all instances are organized in a sortable, filterable, and flexibly customizable table. The LineUp technique allows an interactive exploration of rankings based on multiple attributes of a given tabular dataset [10]. Each instance is a row in the LineUp table. By default, only instances with at least one incorrect classification are shown in InstanceFlow's Tabular View. The columns include the input data (i.e., images in case of image classification), the ground truth class label, and several "difficulty" measures defined in Section 4.2. One column shows the class predictions over time as a colored heatmap (see sixth column in Figure 1), using a categorical color scheme to encode the sequence of predicted classes. An additional column shows a histogram of correct (), incorrect (), and other () predictions (see fifth column in Figure 1). Here, "incorrect" refers to predictions of wrong classes from among the selected subset of classes, while "other" refers to wrong predictions of non-selected classes. The encodings in both of these columns can be switched between time-dependent heatmaps and summarizing histograms.

The LineUp technique includes a number of interactive features to explore the instance predictions: (1) *Ranking:* instances can be sorted by each of the attributes in the columns, or by user-defined combinations of attributes; (2) *Filtering:* Users can further filter the instances, again either by an individual column's value, or by using combined filters on multiple columns. Advanced filtering with respect to temporally changing attributes is possible via regular expressions. (3) *Grouping and Aggregating:* Users can gain an overview of the table by switching to a display mode in which the height of each row is reduced to a minimal height of a single pixel (see Figure 3). As a result, the previously individual heatmaps and bar charts now form a dense, two-dimensional table that reveals overall patterns, similar to the Table Lens technique [20]. User can further condense the display by using the group aggregation feature of LineUp, which shows only summary visualizations for the selected classes. Depending on the attribute type, classes are summarized using histograms or box plots (see Figure 4). The summary histograms for the prediction distributions encode the same information as a confusion matrix.

Difficulty Measures The ranking and filtering operations can help users to identify interesting instances when used in conjunction with measures that describe how difficult an instance is to classify. In this section, we describe three such measures.

Let m be the total number of instances, n the number of classes, and k the number of selected epochs. Let $C(i)$ be the actual class of instance i and $P(i,j)$ the prediction for instance i in epoch j.

The *misclassification score* S of an instance is the fraction of epochs in which it was assigned to the wrong class: $S(i) = (1/k)\sum_{j=1}^{k}[P(i,j) \neq C(i)]$. An misclassification score of 0 means the model predicted the correct class in every epoch, whereas a score of 1 means that the model never predicted the correct class.

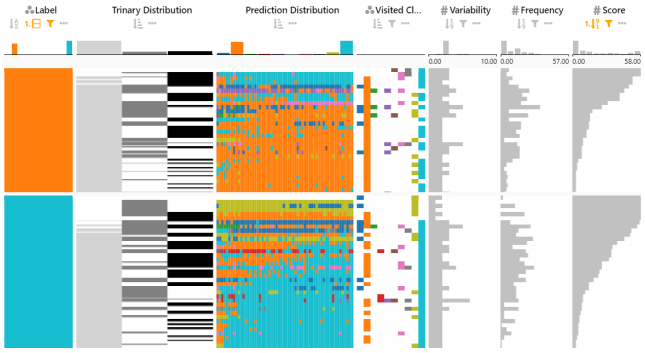

Figure 3: Condensed mode of all instances revealing patterns of successful learning in the classification process.

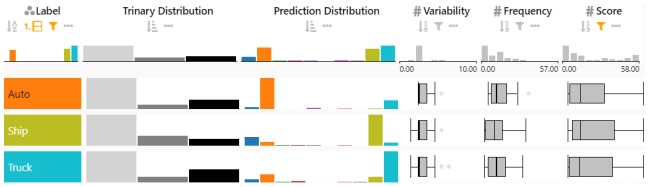

Figure 4: Summary mode of the Tabular View. The overview is similar to a confusion matrix, with correct classifications along the diagonal.

The *variability* V is the ratio of how many classes were predicted for an instance across all epochs: $V(i) = (1/n)|\{P(i,j)\}_{j\in\{1,\ldots,k\}}|$. A variability of $1/n$ means that the model predicted the same class in every epoch, whereas a variability of 1 means that the model predicted every possible class at least once.

The *frequency* F is the ratio of epoch transitions for which the model's prediction jumps between classes: $F(i) = 1/(k-1)\sum_{j=1}^{k-1}[P(i,j) \neq P(i,j+1)]$. A frequency of 0 means that an instance always stayed in the same class, whereas a frequency of 1 means that the prediction changed after every epoch.

4.3 Relationship between Views and Tasks

The different levels of detail in the Flow View and the Tabular View with its different numerical measures have complementary strengths. Table 2 assigns the proposed user tasks from Table 1 to the different visualizations/measures, depending on whether the tasks are well supported (✔), partially supported (✔), or not supported. Instance-focused tasks (**IT1**–**IT4**) are primarily enabled by the Flow View at full detail, whereas the epoch-focused tasks (**ET1**–**ET3**) are better supported by the more aggregated visualizations. The Tabular View supports a wide range of tasks.

For the instance-level analysis, the Flow View is focused primarily on the free exploration of a classifier's behavior, or for tracing individual instances once they have been located. This location of interesting instances is enabled by the Tabular View with its ranking and filtering operations based on the difficulty measures. For epoch-level analysis, the aggregated Sankey visualization provides a good overview of the class distributions and overall flows.

4.4 Implementation

InstanceFlow is a client-side web application built using the React framework. The code for InstanceFlow is available on GitHub[1]. A deployed prototype of InstanceFlow with example datasets and the ability to upload new datasets is available online[2].

[1]Repository: `https://github.com/puehringer/InstanceFlow`
[2]Prototype: `https://instanceflow.pueh.xyz/`

Table 2: Comparison of InstanceFlow visualization components & difficulty measures with respect to the user tasks introduced in Section 2.

Visualization / Metric	IT1	IT2	IT3	IT4	ET1	ET2	ET3
Flow View (basic)					✔	✔	✔
Distribution Bar Charts					✔	✔	✔
Instance Glyphs & Traces	✔	✔	✔	✔		✔	✔
Tabular View	✔	✔	✔	✔	✔	✔	✔
Misclassification Score	✔		✔	✔			
Variability	✔		✔	✔			
Frequency	✔		✔	✔			

5 Use Case: Classification of CIFAR-10 Images

For this use case, we consider a simple neural network trained to classify thumbnail images from the CIFAR-10 dataset. This training and test set is a popular choice in the machine learning field and consists of 60,000 color images (32×32 px) divided into 10 different classes such as *Auto*, *Truck*, *Cat*, and *Dog* [14]. A model developer build a simple CNN using Keras [6]. The developer is satisfied with the overall performance, but notices errors for *Auto* and *Truck* instances. The model developer (user) analyzes the training process with InstanceFlow to better understand what causes these errors.

1. The user trains the neural network to classify CIFAR-10 images, and loads the classification results into InstanceFlow.
2. In the Tabular View, the user groups instances by their actual class and enables the condensed mode with the predictions shown as histograms. This gives the user a hint about class confusion over the total selected epoch range (i.e., similar to the time-integral of a confusion matrix).
3. The user notices that most classes are predicted correctly (with the bin for correct classification being by far the highest). However, for the *Auto* class the user finds that the *Truck* bar is similarly high as the actual class (and vice versa). This is an immediate hint for a high class confusion between *Auto* and *Truck*.
4. The user is now interested in why the neural network incorrectly classifies *Auto* images as *Truck*. To focus on this confusion, the user hides all other classes. Additionally, the user filters the instances to only show those classified as *Auto* or *Truck* at least once. Finally, the user switches from the condensed mode to the normal mode to gain access to the actual underlying instances. Sorting by high misclassification score and low variability reveals to the user that the most problematic instances are mainly *Auto* images classified as *Truck*.
5. Now the user notices a common pattern: the topmost images all show old and bulky cars (see Figure 5).

6. The user proceeds by investigating the flow of these images (see Figure 6). It becomes clear that all of them were correctly classified in early epochs, but then suddenly change to *Truck* one after another.
7. The user checks the traces of random modern-looking cars and finds, in stark contrast to the previous instances, that many of them are temporally stable and correctly classified *after an initial misclassification*. This leads the user to hypothesize that the network, over time, learns features that tend to prioritize modern cars over bulky, antique cars.
8. The user can use these new insights in the subsequent model development or refinement process, e.g., by increasing the number of problematic instances in an attempt to improve the accuracy and temporal stability for *Auto* images.

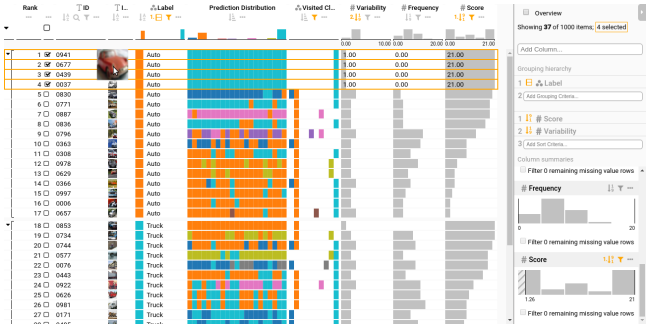

Figure 5: InstanceFlow showing *Auto* and *Truck* instances of CIFAR-10 sorted by high score and low variability, and grouped by the ground truth label.

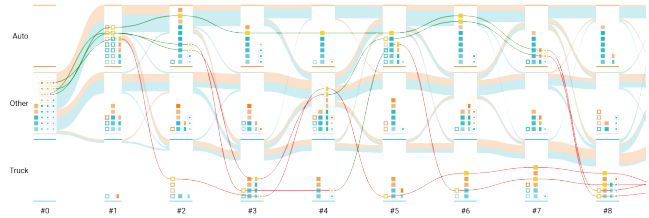

Figure 6: Instance Traces for several selected *Auto* images of bulky, antique cars. These images are correctly classified as *Auto* in the beginning, but tend to be consistently classified as *Truck* over time.

6 Limitations & Future Work

Scalability Due to the combination of aggregated information in the basic Flow View (without Instance Glyphs and Traces) with the functionality of the Tabular View, InstanceFlow scales well to large datasets. However, for more than ~ 100 *selected* instances and at full detail, the InstanceFlow visualization can get cluttered. Additionally, with each selected class, the number of possible paths in the Sankey visualization increases. Thus, additional class aggregation or automatic class and/or instance selection mechanisms would be necessary for exploring datasets with many ($\gtrsim 15$) classes.

Comparison of Datasets A comparison of multiple classification models can be helpful for evaluating the effectiveness of modifications applied during model development. A combined temporal-comparative approach was introduced for class-level analysis with ConfusionFlow [11]. However, it is not straightforward how to extend InstanceFlow to allow similar comparison tasks in a way that is more effective than simply using two InstanceFlow visualizations side by side.

7 Conclusion

We introduced InstanceFlow, a visualization of the evolution of instance classifications in machine learning. The Flow View supports users in understanding the temporal progression of predicted class distributions in a Sankey diagram. Detailed visualizations allows users to trace the predictions for individual instances over time. Interesting instances can be located effectively in the Tabular View, which allows ranking and filtering by numerical difficulty measures. With its different aggregation levels, InstanceFlow bridges the gap between class-level and instance-level performance evaluation while enabling a full temporal analysis of the training process.

REFERENCES

[1] 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

[2] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

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

[4] 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.

[5] A. Chatzimparmpas, R. M. Martins, I. Jusufi, and A. Kerren. A survey of surveys on the use of visualization for interpreting machine learning models. *Information Visualization*, 19(3):207–233, 2020. doi: 10.1177/1473871620904671

[6] F. Chollet. Simple MNIST convnet. `https://keras.io/examples/vision/mnist_convnet/`, 2015. Accessed: 2020-7-10.

[7] S. Chung, S. Suh, and C. Park. ReVACNN: Real-Time Visual Analytics for Convolutional Neural Network. In *ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA)*, p. 7, 2016.

[8] C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27 – 38, 2009. doi: 10.1016/j.patrec.2008.08.010

[9] K. Furmanova, S. Gratzl, H. Stitz, T. Zichner, M. Jaresova, M. Ennemoser, A. Lex, and M. Streit. Taggle: Combining overview and details in tabular data visualizations. *Information Visualization*, 19(2):114–136, 2019. doi: 10.1177/1473871619878085

[10] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit. LineUp: Visual Analysis of Multi-Attribute Rankings. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, 19(12):2277–2286, 2013. doi: 10.1109/TVCG.2013.173

[11] A. Hinterreiter, P. Ruch, H. Stitz, M. Ennemoser, J. Bernard, H. Strobelt, and M. Streit. Confusionflow: A model-agnostic visualization for temporal analysis of classifier confusion. *arXiv:1910.00969 [cs]*, 2019.

[12] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *IEEE Transactions on Visualization and Computer Graphics*, 25(8):2674–2693, 2018. doi: 10.1109/TVCG.2018.2843369

[13] 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, 2018. doi: 10.1109/TVCG.2017.2744718

[14] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical Report Vol. 1, No. 4, University of Toronto, 2009.

[15] A. Lex, H.-J. Schulz, M. Streit, C. Partl, and D. Schmalstieg. Visbricks: Multiform visualization of large, inhomogeneous data. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '11)*, 17(12):2291–2300, 2011. doi: 10.1109/TVCG.2011.250

[16] A. Lex, M. Streit, H.-J. Schulz, C. Partl, D. Schmalstieg, P. J. Park, and N. Gehlenborg. StratomeX: visual analysis of large-scale heterogeneous genomics data for cancer subtype characterization. In *Computer graphics forum*, vol. 31, pp. 1175–1184, 2012.

[17] D. Liu, W. Cui, K. Jin, Y. Guo, and H. Qu. DeepTracker: Visualizing the training process of convolutional neural networks. *ACM Transactions on Intelligent Systems and Technology*, 10(1):1–25, 2018.

[18] S. Liu, X. Wang, M. Liu, and J. Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 1:48–56, 2017. doi: 10.1016/j.visinf.2017.01.006

[19] 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, 2018. doi: 10.1109/TVCG.2017.2744358

[20] R. Rao and S. K. Card. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94)*, pp. 318–322. ACM, 1994. doi: 10.1145/191666.191776

[21] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams. Squares: Supporting Interactive Performance Analysis for Multiclass Classifiers. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):61–70, 2017. doi: 10.1109/TVCG.2016.2598828

[22] 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, 2019. doi: 10.1109/TVCG.2018.2864504

[23] 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, 2018. doi: 10.1109/TVCG.2018.2816223

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

[25] W. Zhong, C. Xie, Y. Zhong, Y. Wang, W. Xu, S. Cheng, and K. Mueller. Evolutionary Visual Analysis of Deep Neural Networks. In *ICML Workshop on Visualization for Deep Learning*, p. 9, 2017.