# DeepCompare: Visual and Interactive Comparison of Deep Learning Model Performance

Sugeerth Murugesan*
UC Davis

Sana Malik, Fan Du, Eunyee Koh†
Adobe Research

Tuan Manh Lai‡
Purdue University

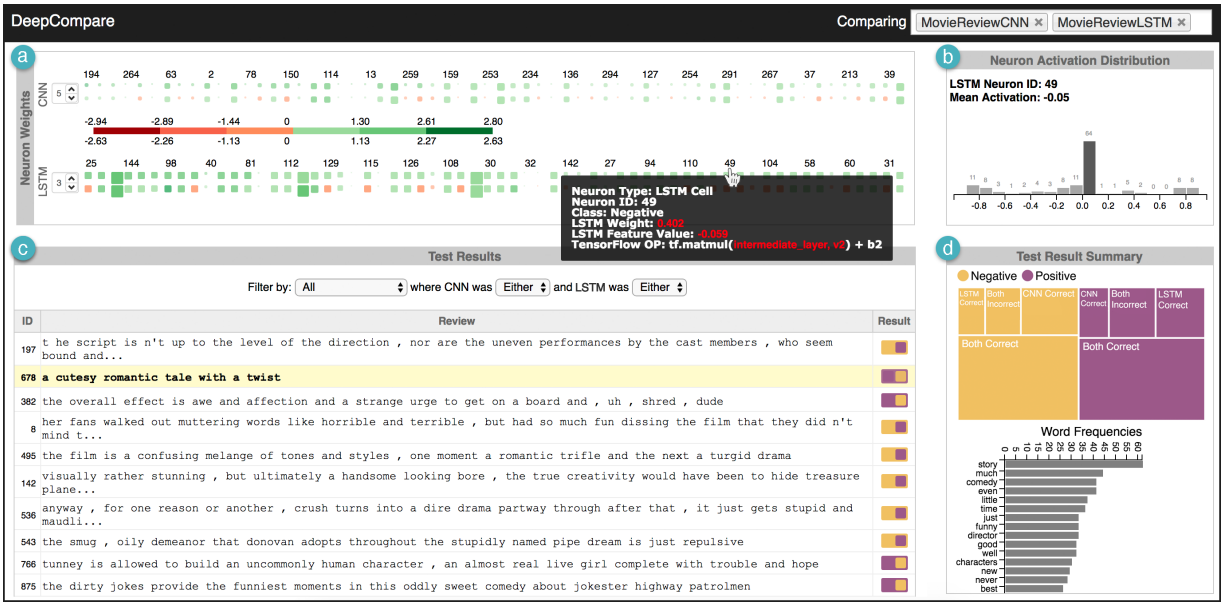


Figure 1: *DeepCompare* enables users to systematically compare results of two models in order to understand the differences and tradeoffs between them. Here, we compare a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) model performing sentiment classification on movie reviews. The CNN and LSTM have similar accuracy scores (76.3% and 76.5% respectively) but have very different qualitative behavior. The Neuron Weights Detail panel (a) visualizes the weights and feature values of one model layer as a heatmap (*red* = lower, *green* = higher). The user has selected Neuron 49 from the LSTM, and its Neuron Activation Distribution panel (b) indicates a sparse distribution from -1 to 1 with most test instances remaining inactive. The Test Result Detail panel (c) lists a detailed view of the results sorted by the selected neuron's activation values and filtered by their results. To get a high-level view of all the results, we can refer to the Test Results Summary panel (d) and see that for both classes, both models get the same proportion of predictions correct, but the CNN performs slightly better on negative reviews while the LSTM performs better on positive reviews.

## Abstract

Deep learning models have become the state-of-art for many tasks, from text sentiment analysis to facial image recognition. However, understanding why certain models perform better than others or how one model learns differently than another is often difficult yet critical for increasing their effectiveness, improving prediction accuracy, and enabling fairness. Traditional methods for comparing models' efficacy, such as accuracy, precision, and recall provide a quantitative view of performance, however, the qualitative intricacies of why one model performs better than another are hidden. In this work, we interview machine learning practitioners to understand their evaluation and comparison workflow. From there, we iteratively design a visual analytic approach, DeepCompare, to systematically compare the results of deep learning models, in order to provide insight into the model behavior and interactively assess trade-offs between two such models. The tool allows users to evaluate model results, identify and compare activation patterns for misclassifications and link the test results back to specific neurons. We conduct a preliminary evaluation through two real-world case studies to show that experts can make more informed decisions about the effectiveness of different types of models, understand in more detail the strengths and weaknesses of the models, and holistically evaluate the behavior of the models.

**Index Terms:** Human-centered computing—Visualization—Visual analytics; Computing methodologies—Machine learning

## 1 Introduction

Deep learning algorithms have become the state-of-the-art for many tasks in various domains, including medicine, security, and marketing. In natural language processing in particular, techniques like Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have achieved impressive results in sentiment analysis [17, 38], language modeling [26, 39] and speech recognition [11]. However, these improvements come at the cost of more complex models with vast parameter spaces, complex dependencies, and multi-layered architectures. Further, with the advent of deep learning, the exact structure of the model may be unknown or

*e-mail: sugeerth@gmail.com
†e-mail: {sana.malik, fdu, eunyee}@adobe.com
‡e-mail: lai123@purdue.edu

too complex to interpret. As a result, many of the algorithms are black-boxes, making it difficult to understand, diagnose, and explain results or analyze how the system has learned [20, 34].

Recent work in visual analytics has explored methods towards explaining the inner-workings of deep neural networks through interactive model analysis [16,27,37,43]. However, most efforts focus on analyzing a single model, and while they begin to *"open the black-box,"* systematic comparison of performance and models themselves remains an open problem. When refining or designing new model architectures, machine learning practitioners rely on their intuition, domain knowledge, and prior experience, using quantitative metrics such as accuracy, precision, and recall as a guide, leading to a model building processes that are primarily based on trial-and-error [30]. For incremental improvements (e.g., testing different parameters on the same architecture), these traditional metrics may be sufficient. However, when comparing models with incomparable architectures (e.g., a CNN versus an LSTM), these metrics do not accurately capture the different trade-offs between the models or answer *how* and *why* they may behave differently, for example, being biased towards certain features in the training set [44]. Additionally, because the metrics are aggregated, two models can have a similar score while having vastly different qualitative behavior [33].

We propose that a visual analytics approach can enable practitioners to more directly and deeply compare the performance of machine learning models. By interactively linking the model structure and the test instances, machine learning practitioners can better examine the similarities and trade-offs between two models.

We begin by working closely with five machine learning practitioners to better understand their evaluation workflow and understand in which contexts model comparison in necessary and the challenges they face. Through a series of interviews, we identified three tasks and six challenges. We then iteratively designed and developed a system, DeepCompare, to support these tasks. Given two trained models, the tool enables users to explore how the models perform differently. DeepCompare enables practitioners to get an overview of the performance of the two models, identify the test instances where the models agree or disagree, and systematically explore results based on classifications. To understand the root-causes of misclassifications, the tool visualizes activation patterns in the models side-by-side and further links specific neurons to highly activated test results (Figure 1a,b). Interactions like brushing and linking and details-on-demand are incorporated within multiple views providing users the ability to intuitively compare and contrast activation patterns across instances. The tool is aimed at machine learning practitioners for selection and debugging of different model architectures, enabling better understanding and improvement of models' components.

The key contributions of this work are:

- Interviews with machine learning experts to identify tasks for evaluating and comparing deep learning models,
- A visual analytics approach that enables the systematic comparison of two models with different architectures for providing insights into the model behavior and performance,
- And two case studies with real-world machine learning experts, using DeepCompare, showing its efficacy for comparing two deep learning models.

The paper is organized as follows: Section 2 provides an overview of related work in model interpretation and comparison. Section 3 describes our interviews with machine learning experts and presents the tasks and challenges for effective model evaluation. Section 4 describes the design and system overview of DeepCompare and Section 5 describes the two preliminary case studies. Lastly, Section 6 provides a discussion of the benefits and limitations of DeepCompare and Section 7 concludes the paper.

## 2 Related Work

Deep neural networks (deep learning) have emerged as an important topic with a wide range of application areas, including image recognition [13], natural language processing [7], disease diagnosis [9]. Given this broad applicability, a variety of interactive visualization techniques for interpreting deep learning models have been developed in recent years (see [4, 14, 23, 35] for insightful surveys). This section provides an overview of related literature focusing on the interpretation of deep learning models and comparison of model performances, which are most relevant to DeepCompare.

### 2.1 Interpretation of Deep Learning Models

While accuracy is a major indicator of how a model performs and which models to choose, people often want to understand how and why one model performs better than another. This can increase people's trust and provide insights for improving the model accuracy or better adapting the models to specific applications. Since the successful deep learning models often require a combination of a specific set of parameters, activations, configurations of layers, interactive interpretation of such parameters has become an active research area, which sheds light into the inner-workings on complex machine learning models and can help people better understand the major components of a learning algorithm. One common approach is to provide filters and show activations [42]. This procedure allows the user to better explore the models learned in the hidden structure of the layers. Many interactive tools have been developed to explore the parameter space of visualizing the activation information. For example, Tzeng et al. [40] and Harley et al. [12] laid out neural networks as node-link diagrams and visualized how the neuron weights are learned.

Another popular approach for interpreting deep learning models is to show the performance of each specific instance of a model, which allows users to explore how the model behaves for specific instances that are of primary interest. For example, Patel et al. enabled interpretability by having a tight coupling of data instances with performance [30]. Amershi et al. [3] used a score-based system to explore how the model performs for binary classification tasks. While these methods work in a non-agnostic fashion, there are other methods that specifically explore techniques for neural network models [12, 21]. Such techniques allow users to pick an example, feed it into a neural network, and see how the network behaves based on the input. Although it is helpful to understand the data instance-by-instance, these techniques generally have scalability issues when handling large datasets, which can make the analysis tedious and time-consuming.

To mitigate this issue, scalable techniques have found their way into visualizing neurons and their structure. Liu et al. [22] proposed a method to cluster the datasets into subgroups and enable users to explore the activation patterns of a particular instance of a neural network. CNNVis [22] presents an interactive visual system for convolutional neural networks that utilized hierarchical clustering of group neurons and bi-directional edge-bundling to improve the scalability. Besides, embedding projector [36] and dimensionality reduction [24] are also popular for exploring the underlying dynamics of the feature data. ReVACNN [6] combined multiple coordinated views with a projected view to provide a visual summary of instance activations and Rauber et al. [32] studied how the projected view can help users understand changes in the activation patterns. Moreover, Krause [18] et al. pursued the direction of exploring transparency in neural networks for raw data, using features and aggregate statistics for data distribution.

Additionally, approaches like Gestalt [29] and Prospector [19] combine feature data along with traditional metrics to enable explanation of models by describing how and why instances are predicted the way they are and its effect of the overall machine learning metrics. Further, Patel et al. [29] provided a holistic picture of the

underlying machine learning pipeline from data acquisition to prediction. Kahng et al. [16] compared a model's performances through subset analysis helping users. Compared to existing approaches for interpreting a single model, our work focused on the task of supporting machine learning practitioners systematically comparing two deep learning models. Our DeepCompare system provides visualizations and user interactions for analyzing agreement and disagreement between models and comparing two models in terms of performance, activation patterns, and neurons.

### 2.2 Comparison of Model Performances

Given a set of suitable models, the goal of model comparison is to find the best-performing model that is most generalizable, has the least loss, and is the most representative of the task to be performed. Such comparisons are important in identifying the pros and cons of architectures, providing direct guidance for model designers to better select an architecture for a given machine learning task. Many machine learning frameworks like Tensorflow [1], Weka [15], and scikit-learn [31] provide summary statistics such as accuracy, precision, recall as built-in functions for the evaluation of models. These statistics provide a simple indicator of how a model performs and allow for a one-to-one comparison between models. However, as studied in work by Ren et al. [33], interpreting this information can be misleading as the predictions are treated equally, hiding crucial information.

Visual analytics methods, which provide contextual information beyond summary statistics, have successfully applied for machine learning model selection [2, 16, 25]. Typically, these visualizations use a compact design to show an overview of the accuracy, loss, and other user-defined metrics of each model, which enable users to compare the models' performances at an abstract level. Data context information is displayed on demand, allowing users to conduct detailed analyses when necessary. The most common design choices are juxtaposition (side-by-side), superposition (overlay), and explicit encoding [10]. For example, Chuang et al. [5] and Alexander et al. [2] studied visual approaches for comparing topic models. Yu et al. [41] explored multiple recurrent neural networks' hidden states to explore the data at the sentence-level. Zeng et al. [43] compared the models in different snapshots during different epochs of the training process to better explore how the model evolves over time.

Most of the existing methods treated the underlying neural networks as a black-box, which made it difficult for users to fully understand the training data or the model. Moreover, a qualitative comparison of two different models with different architectures remains an open challenge. In our work, we focus on exploring and comparing the results produced by two models with different architectures and understanding why they perform differently. We provide an interactive system that can help users analyze major activation patterns used to produce the model results, where the activations are trained with the same input data and evaluated using the same test data but applied on different models. Our visualizations can help us better explore the neurons' weights, summary activation data, and the underlying classification results of testing instances.

## 3 Model Comparison in the ML Workflow

To clarify design requirements and better understand when and how machine learning practitioners' compare models, we interviewed five researchers who build models in their day-to-day.

All five interviewees were machine learning researchers at a large software company. Two specialized in natural language processing and three in user modeling. The semi-structured interviews were conducted in three groups based on use case and each lasted one hour. The researchers were asked to talk about a specific model on which they were currently working, their general evaluation workflows, and challenges they faced. We briefly describe the three example projects given by the researchers before discussing tasks and challenges.

*Text-based Question Answering using Context (Q&A).* The first pair of researchers were building system that could capture context in natural language systems, such as in conversation agents and question answering systems. For example, suppose a customer was interested in purchasing a washing machine and was looking at a specific model. He could ask "How much water does it use?" and receive the answer "27 gallons." The model was based on a recurrent neural network (RNN) and was trained on question-answer pairs. Given a product and a question, the model would return the answer (or specification) for the product.

*Next Action Prediction based on User Behavior Logs (NAP).* The next researcher was working on predicting users' next actions given their behavior histories by building a model to learn a latent user embedding.

*Outcome Prediction based on User Profiles (OP).* The final two researchers were working on building a model based on users' demographic information to predict the most likely outcome for a user (e.g., click an ad or make a purchase). Their major challenge was the nature of the demographic data: it was categorical, sparse, imbalanced, and of high-cardinality.

### Tasks and Challenges

In the practitioners' workflows, model comparison presented itself in two places: (1) *model tuning*, iteratively adjusting parameters of a model to improve performance and (2) *model evaluation*, comparison of a finalized model against the state-of-the-art or alternative models, which might not have a similar architecture. Across application areas and model types, the practitioners described three main tasks (T1–T3) and six challenges (C1–C6) when comparing models.

#### T1. Compare aggregate performance metrics.

Arguably the most important step in any model-building workflow is quantifying the model's success, from training and validation to evaluation against the state-of-the-art. In model tuning, changes to parameters and model architecture are often guided first by quantitative measures (e.g., accuracy, F-score, or Area-Under-the-Curve [AUC]), then by domain knowledge and intuition. In model evaluation, researchers may be proving advancements in performance over the state-of-the-art, which may have an entirely similar architecture. Aggregate performance metrics allow machine learning practitioners to compare their models against a large number and wide variety of baselines.

» **C1. Loss of detail**. For example, two classifiers, A and B, might have very similar accuracies, but A may result in fewer false positives and be preferred in cases where false positives should be minimized.

» **C2. Loss of context**. In tuning, a small change in parameters could result in a large change in performance. With simply the aggregate metric, it is difficult to understand why and how the results were changed and practitioners expressed a desire for more fine-grained results.

#### T2. Understand differences in error patterns.

From the high-level metrics, the researchers aim to understand common patterns causing errors by viewing the results in more detail. In tuning, this step informs potential modifications: perhaps similar test examples are being misclassified due to a feature that can be added. For example, in the NAP use case, the practitioner may want to isolate examples where the prediction was "open e-mail" but the actual value was "delete e-mail" and see if there are any commonalities in them. In evaluation, understanding differences in specific test examples can help differentiate models and more accurately describe their trade-offs. Because test datasets contain upwards of thousands of examples, this qualitative evaluation is typically done

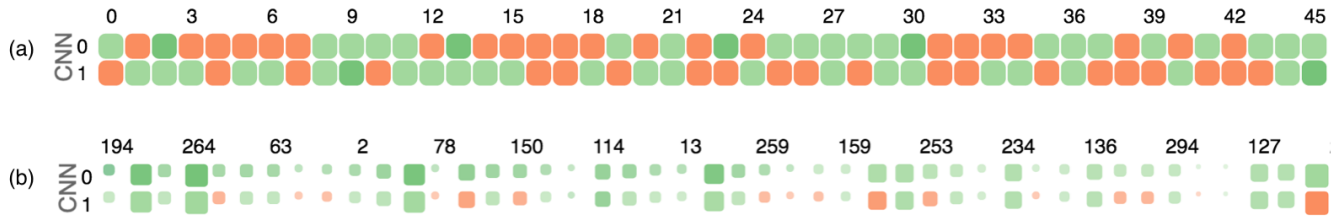


Figure 2: The Layer Weights Detail heatmap displays the learned weights for a user-selected layer of the model. (a) By default, the neurons are laid out in order of the input data, matching its size and shape. Each cell is colored by the weight in that layer. (b) When a user selects a test instance, the cells are re-ordered from most positive to most negative and resized by the activation value for that test instance.

through categorical error analysis (e.g., confusion matrices and error type breakdowns) and spot-checking.

» **C3. Lack of direct comparison.** While categorical error analysis is extremely helpful in systematically evaluating the results of a single model, the researchers must repeat this process independently for each model they are comparing. Visualizations such as confusion matrices help in presenting performance result succinctly, but finding common patterns between models becomes increasingly difficult when comparing a large number of models.

» **C4. Lack of explanation.** These detailed summaries explain *what* is happening, but not necessarily *why*. For example, in our Q&A use case, one research recalled a test example that was classified correctly, but he was surprised because there were no similar examples in the training set. He wanted to explore whether there was a latent representation he missed or if this was due to chance, but because the results were independent of the models, he could not explore if there were other similar examples.

T3. Examine underlying model layers.

In cases where the researchers had access to the underlying model, they would explore the behavior of the model as it relates to learning and inference. Three out of five of the practitioners used a similar model visualization tool (such as Tensorboard) to explore the models independently of each other. The other two did not visualize the models, but used Jupyter notebook or similar to inspect the learned model weights.

» **C5. Inability to compare different architectures.** While this worked adequately in the tuning scenario, in the evaluation case where the architectures were very different, comparison of the visualizations did not make sense. Instead, the practitioners wanted a way to explore major activation patterns for classes of errors *across models.* For example, cases where Model A was correct and Model B was incorrect.

» **C6. Inability to understand neuron behavior.** In addition to correlating examples to neuron activation patterns, the practitioners wanted to correlate *specific neurons* to highly activated examples as an exploratory way to understand its logic and summarize the neuron's behavior.

## 4 DeepCompare: Interactive Model Comparison

Through understanding the challenges laid in the previous section, we defined six design goals:

G1. Provide a high-level overview of performance (C1).
G2. Provide context into prediction results (C2, C4).
G3. Compare test results across models directly (C3, C5).
G4. Correlate test instances with model activation patterns (C4, C5).
G5. Be architecture agnostic (C5).
G6. Correlate neuron activation patterns with test instances (C6).

DeepCompare was iteratively designed with ongoing feedback from three of the five machine learning practitioners from the design requirement collection. Its final design consists of 4 coordinated panels (Figure 1): (a) layer weights details, (b) neuron activation distribution, (c) test result details, and (d) test result summary. In this section, we detail each component of the interface and describe a usage scenario for movie review sentiment analysis.

### 4.1 Layer Weights Detail

Towards design goals 2, 4, and 6, we began by visualizing the learned model weights as a heatmap (Figure 2a). In order to be architecture agnostic (G5), we display only one layer at a time, pre-selected by the user. While in its current version, the pre-chosen layer remains static, we intend to allow for dynamic switching between layers in the future.

For each neuron and each test instance, we encoded two dimensions: the learned weight as color (red-green) and the activation value as size. The size and shape of the heatmap is dictated by the architecture of the model and the dimensions of the layer.

From the Layer Weights Details, users can select a neuron and review a list of data instances in the Test Result Details (Figure 1c). Instances are sorted by their activation for the selected neuron so that the most relevant and responsive ones are on the top. This interaction enables users to investigate misclassifications by exploring properties in the data instances that may have caused a high neuron activation.

### 4.2 Neuron Activation Distribution

To understand the role of specific neurons (G2, G6), we provide a summary of the activation distribution across all test instances for each neuron. This overview helps users identify neurons that were persistently active or inactive. We employ a histogram chart to show the activation distribution (Figure 1b) where the x-axis represents bins of activation values and the y-axis represents the number of test instances that fall in each bin. Additionally, when a user selects a specific neuron to inspect, the test result details view is ranked from highest to lowest activation on that neuron.

### 4.3 Test Result Details

Users can explore the Test Result Details (Figure 1c) and review the low-level details of the test instances, such as the raw sentences, ground truth labels, and predicted classes. This table enables users to directly inspect individual instances and see if they have been classified correctly, which is useful for investigating the similarities and differences between the models (G3).

We proposed two visual designs for conveying multi-model classifications. The initial glyph design (Figure 3a) used color to indicate if model A is correct (green) or incorrect (red). The shape of the glyph indicated if model B agreed (square) or disagreed (circle) with model A. However, our users found that although this design can

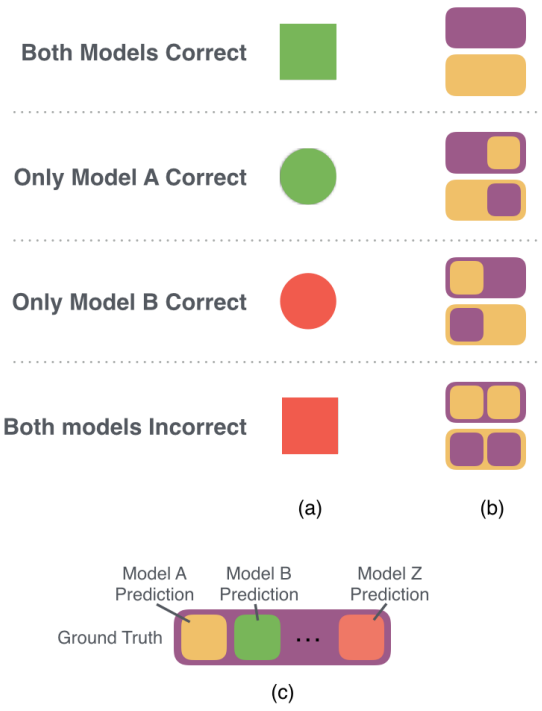


Figure 3: We explored two designs for reviewing model results of each instance: (a) colors indicate if model A is correct (green) or incorrect (red) and shapes indicate if model B agreed (square) or disagreed (circle) with model A; (b) the colors of two side-by-side squares show the model results and the color of a background rectangle shows the ground-truth. (c) The side-by-side square design can be extended to comparing multiclass classifiers.

clearly show the model agreement information, the color is misleading especially when model B is correct but model A is incorrect, which is shown as a red circle. Additionally, this method would not be generalizable across more than two models and obscures the actual value of the classification.

To address these issues, our final design (Figure 3b) showed the model results as side-by-side squares inlaid in a larger rounded rectangle. The color of the inlaid squares indicates the corresponding model's prediction and the color of the background rectangle represents the ground-truth label. With the final design, users only need to compare the colors to tell if the models agree with each other and if the results matched the ground truth. It also keeps the color scheme consistent with other views. Figure 3c illustrates how this view might be extended to comparing 3 or more multiclass classifiers.

Selecting a test instance will overlay the activation values onto the model heatmaps, indicating the values by size (Figure 2b, G4).

#### Model Agreement Controls

While it is easy to inspect every predicted label when the data is small, as the number of data instances grows, this process becomes cumbersome and tedious. We designed model agreement controls (Figure 1c) that accelerate the exploration by allowing users to quickly locate data instances and classification patterns of interest. Specifically, users can filter the data based on the classes in the ground-truth labels. They can also choose to only keep instances that are correctly classified or misclassified by both or either of the models. A typical query is *"show me negative reviews where the CNN model was correct but the LSTM model was wrong."* Such queries are effective for exploring errors such as false positives and false negatives so as to gain a deeper understanding of how the models performed differently and why the error occurred.

### 4.4 Test Result Summary

#### Metrics Treemap

Machine learning practitioners often need to compare the overall performance of two models to decide what they should further explore in detail (G1). Specifically, when working on classification models, such overviews allow users to quickly inspect misclassification patterns and identify outliers of interest. As illustrated in Figure 4, we used a hierarchical design to provide a performance overview, where the first level represents the ground-truth classification of the data and the subsequent levels show the classes predicted by the models. To support a visual comparison of the hierarchies, we showed the hierarchical structure in a treemap (Figure 4a-d). For example, when dealing with binary classification models, we divide the treemap into two subdivisions showing the value of the positive and negative classes. Then, within each subdivision, the four divisions depict the distribution of performance of both models.

#### Frequent Word Histogram

In the sentence classification problem, users often want to review frequently occurring words in the text corpus that are most influential to the models' performances. To support this task, we initially designed a word cloud which aggregates all the data instances and uses font sizes to encode word frequencies. However, our users found it difficult to precisely compare the sizes and thus we used a histogram instead where the words are ordered by frequencies (Figure 1d).

### 4.5 Example Workflow

In a general scenario of using our tool, machine learning practitioners start with two pre-trained deep learning models and the results from each model on a single test dataset. The system begins by calculating high-level statistics about the performance of each model to construct the Test Result Summary treemap and word frequency chart. Oftentimes the practitioner has already run the relevant statistics and may be interested in exploring specific test results and classes in detail. From here, the practitioner can filter test instances using the model agreement controls (Figure 1c). By reviewing the filtered data, users can directly explore the instances and compare the models in terms of the classification results and ground-truth (Figure 1c). Users can further explore the frequent word histogram (Figure 1d) to see words that are highly correlated with the selected test set. Practitioners can then explore activation patterns for specific test instances, and explore common patterns between test instances. Conversely, the practitioner can choose to focus on a specific neuron, and find test instances that highly activate the neuron. By examining the top $n$ activate instances, the practitioner can begin to draw conclusions about the neuron's behavior. With a combination of these visualization views, machine learning practitioners can effectively compare two deep learning models and gain actionable insights for improving the models or deciding which model to use.

## 5 Case Studies

We conducted two case studies to demonstrate the effectiveness of DeepCompare in helping practitioners compare LSTMs and CNNs models qualitatively. We worked closely with two machine learning experts in natural language processing (NLP) with two datasets.

### 5.1 Movie Review Sentiment Analysis

The first case study used a publicly-available dataset consisting of movie reviews and sentiment [28] (Figures 1 and 5). The dataset consisted of 5,313 positive and 5,312 negative labeled reviews. The participants trained two models with the goal of classifying reviews

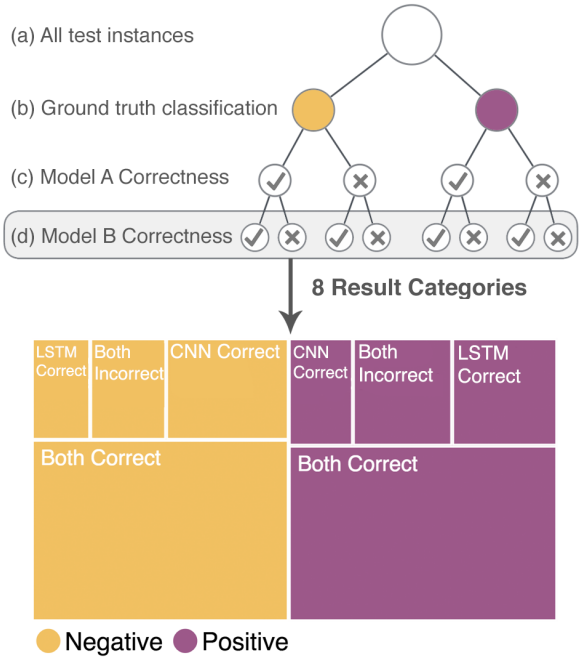


Figure 4: Metrics Treemap for visually comparing the overall performances of the models. All test instances are divided by (b) ground truth classification, (b) the correctness of model, and (c) the correctness of model B.

as either positive or negative: a Convolutional Neural Network (CNN) described by Kim et al. [17] and a Long-Short Term Memory (LSTM) model designed by the participants. The accuracies of both the CNN and LSTM models were similar, 76.3% and 76.5% respectively.

First, the analysts explored the treemap overview of the classification results for positive and negative test instances in the data (**T2**) (Figure 1d). Though the accuracies were similar, the analysts discovered the CNN misclassifies positive reviews more often than the LSTM, whereas, for negative examples, CNN performs better. Based on the overview, the analysts further explored negative reviews where the CNN was incorrect by filtering the results table. Viewing the word frequency histogram, the analyst was interested to see many positive words like *comedy* and *well*.

The analysts began to explore the activation patterns by clicking specific test instances, e.g. Neuron #194 (Figure 5a). They saw that this neuron was not activated for most test instances (Figure 5b). By examining the activation-ordered Test Result Details view, they saw that there were many Spanish language reviews captured correctly by the CNN but not the LSTM (Figure 5c). They were surprised by this because they had not expected Spanish reviews in the test set and were unsure whether any existed in the training set. This encouraged them to review their training data.

They then filtered by test instances where the LSTM was correct but the CNN was incorrect. By interacting with different patterns of activation and selecting, they found a particular signature for negative reviews where CNN classifies incorrectly. In particular, LSTM Cell #112 was consistently highly activated in classes where the LSTM was correct but the CNN was incorrect. However, the Neuron Activation Distribution was skewed highly towards 0, and they hypothesized that this neuron identifies a specific, strong signal for positive reviews, but it doesn't occur frequently in the test set.

The analysts then switched the filter to positive reviews where the CNN was incorrect and the LSTM was either to better understand the strengths of the LSTM. The word histogram showed frequent words that played a role in misclassifications by the CNN, including *little, even,* and *new*. They selected an instance *"with a romantic..."* and examined the activation results to pick a highly activated neuron, which had mostly negative activations on the test instances.

The analysts chose to further work with the CNN, based on the performance exploration in DeepCompare, despite it having a slightly lower accuracy.

### 5.2 Answering Product Questions

For the second dataset, the practitioners were interested in creating a chatbot for answering questions about specifications of a product. The product information dataset contained information as tuples: `(product id, specification name, specification value)`. Given a product, a question, and a specification, their task was to match the question to the most relevant product specification. Specifically, given a question $Q$ about a product $P$ and the list of specifications $(s_1, s_2, ..., s_M)$ of $P$, the goal is to identify the specification that is most relevant to the question $Q$. The team had been iteratively debugging and exploring different deep learning models. Each model takes a question and a specification name as inputs and outputs a score indicating their relevance. For example, given the question *"How heavy is it?"* and the specification name *"Product Weight (in pounds)"*, the system outputs a relevance score 0 to 1 (higher is better). To convert this into a binary classification problem, every pairwise specification, and question combination is tested, and the specification with the highest score is selected for each question. This is similar to the approach mentioned in [8].

Using Amazon Mechanical Turk [1], an online crowdsourcing marketplace, the team collected a dataset of 7,118 question-specification pairs in total. The dataset was then divided into a training set, a validation set, and a test set (with the proportions being roughly 80%, 10%, and 10%, respectively). Initially, the team started with training and optimizing a deep learning model based on LSTM. After that, the team also experimented with a model based on CNN. Quantitatively, the models have similar accuracy scores of 80.1% for the LSTM-based model and 76.6% for the CNN-based model. However, the team wanted to use our tool to analyze qualitatively how the models perform differently on the test instances. Major questions of the study included *"What are the misclassified instances of LSTM that CNN got it right?" "What qualitative aspects did the LSTM miss?" "Why does this happen?"* and *"Is there a hidden layer activation pattern for misclassifications?"*

While getting an overview of the correctly classified results for positive test instances (**T2**), the practitioners were specifically interested in the misclassifications of the positive instances (**T3**), where the CNN classified the instance correctly and the LSTM misclassified. They explored the major words describing the positive misclassifications. They then examined a specific test instance (a product id, product specification, and a question pair) in the interactive table to explore its corresponding activation patterns (**T4**). Interestingly, they found a lot of neurons unactivated (having the value 0) for the CNNs, however, for LSTMs, a major proportion of the neurons remain activated (**T1**).

Also, by examining different test instances, they found that there is a similar pattern for positive test instances in CNNs, the highly activated neurons are on the positive side of neuron weights (**T1**). By going over the activations of different instances, it was interesting that a neuron (34) behaved anomalously showing spurious activations for many test instances. While clicking on a specific neuron, they explored the most activated test instances. The practitioner found a wide variety distinguishing patterns in misclassifications between the two models. For example, the frequent words in the

[1] https://www.mturk.com

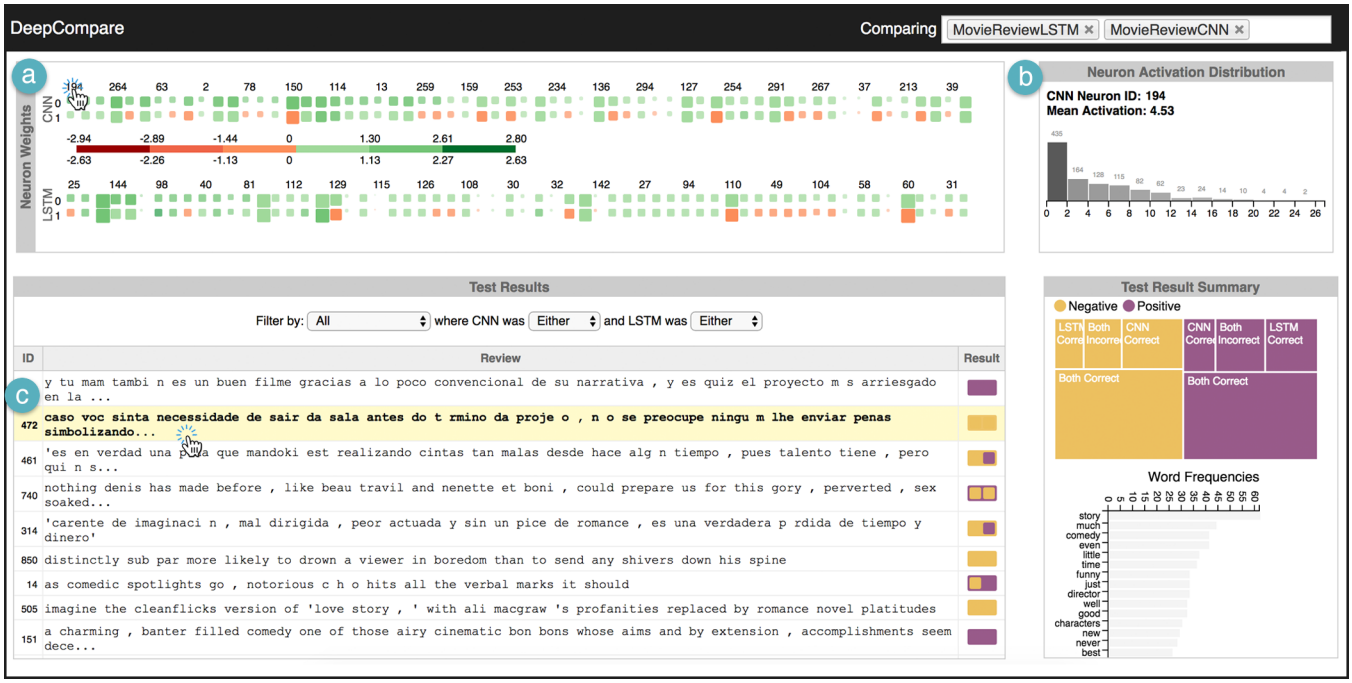


Figure 5: A machine learning practitioner explores two models for movie review sentiment classification: a Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM) model. (a) While exploring, he examined Neuron #194. (b) The Neuron Activation Distribution showed a left-skewed distribution, indicating that not many test instances activated this neuron. (c) By observing the list of test instances (ordered by activation of Neuron #194), he found that the highly activated test instances were in Spanish and the CNN was better at capturing the correct sentiment than the LSTM.

*positive* classifications where either the LSTM or the CNN were incorrect were similar, indicating that both the models may be affected by the same words in misclassifications.

The practitioners found it most interesting to be able to freely and interactively explore test results in context. In particular, they found it helpful to be able to explore activation patterns on an instance-by-instance basis, a level of granularity they were previously unable to achieve.

## 6 Discussion

Through observing two machine learning experts explore their own model performances through DeepCompare, we found that DeepCompare helped facilitate a deeper understanding of models' benefits and tradeoffs. Overall, the participants were able to make insights about the differences between the models and systematically select a model.

**Visual exploration of activation data.** We found that, initially, understanding the distribution of activation patterns for different test instances was difficult as users did not have a representative activation pattern in mind for a specific test instance. However, after multiple interactions with the tool, users were able to discover patterns across multiple test instances and neurons and gauge which neurons corresponded to which classes and features.

**Identifying incorrect or noisy data in the test set.** Inspecting high activations of specific neurons revealed noisy samples in the dataset that were misclassified. For example, in the movie review dataset, many non-English reviews were discovered. This was not clear from the traditional metrics, but by linking the raw data with categorical results it became readily apparent when the LSTM was more susceptible to noisy data. While the visualizations were helpful in arriving at conclusions about model performance, it is interesting to note the users also found some incorrectly labeled data in the test example ground truth through the word histogram and instance table. While this may only result in minor changes in accuracy, it would be interesting to examine how such discoveries affect the training process itself.

**Summarizing correlated words with false positives and negatives.** The word histogram helped the experts to link specific test words with the classification results. For example, by filtering the test examples to *"Negative Reviews"* where the CNN was incorrect, words like *well, comedy*, and *good* were highly represented than negative words like bad and little.

While the preliminary case studies begin to demonstrate DeepCompare's utility, there are, of course, many limitations and future directions for this work.

**Generalizability to non-text data types.** In this work, we focused primarily on text data. More work would need to be done in extending DeepCompare to other data types, especially in multivariate data or images where textual representations are not feasible.

**Scalability of heatmap visualization.** Further more, because we encode the learned weights as a heatmap, the system can represent most models that can be represented as a matrix. However, this is not the case for models like Decision Trees. Additionally, typical models may contain thousands of neurons, which would not extend to this method.

**Extending to domains with no ground truth.** The design of DeepCompare relies heavily on having a known ground truth, which is not always the case, as in recommender systems. Evaluating performance on models without correct labels remains an open challenge.

**Integration of training data.** Future work includes integrating training data in order to explain and understand how models learn differently.

## 7 Conclusion

We present DeepCompare, a visual analytics tool for comparing the performance between two deep learning models. Through interviews with machine learning practitioners, we identified four tasks for qualitative performance evaluation and iteratively designed a system, DeepCompare, to support these tasks. We evaluated DeepCompare with two case studies with two natural language processing researchers to compare a CNN and an LSTM on their own data. Using DeepCompare, analysts were able to better understand what and how two models learn differently from the same training data and further enable exploration of root-cause misclassifications by the algorithms. In future work, we plan to extend the framework to enable the aggregated visualization of multiple layers of the deep learning model for efficient exploration and comparison. Additionally, we will explore methods for tighter coupling between the training data and the learned weights, to further explain differences in learning.

## Acknowledgments

The authors wish to thank Trung Bui, Sheng Li, and Nedim Lipka for their guidance on this project.

## References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, pp. 265–283. USENIX Association, 2016.

[2] E. Alexander and M. Gleicher. Task-driven comparison of topic models. *IEEE transactions on visualization and computer graphics*, 22(1):320–329, 2016.

[3] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 337–346. ACM, 2015.

[4] J. Choo and S. Liu. Visual analytics for explainable deep learning. *arXiv preprint arXiv:1804.02527*, 2018.

[5] J. Chuang, S. Gupta, C. Manning, and J. Heer. Topic model diagnostics: Assessing domain relevance via topical alignment. In *International Conference on Machine Learning*, pp. 612–620, 2013.

[6] S. Chung, S. Suh, C. Park, K. Kang, J. Choo, and B. C. Kwon. Revacnn: Real-time visual analytics for convolutional neural network. In *NIPS Workshop on Future of Interactive Learning Machines*, 2016.

[7] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, 2008.

[8] L. Cui, F. Wei, S. Huang, C. Tan, C. Duan, and M. Zhou. Superagent: A customer service chatbot for e-commerce websites. In *Proceedings of ACL 2017, System Demonstrations*, pp. 97–102. Association for Computational Linguistics, July 2017.

[9] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.

[10] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011.

[11] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649. IEEE, 2013.

[12] A. W. Harley. An interactive node-link visualization of convolutional neural networks. In *International Symposium on Visual Computing*, pp. 867–877. Springer, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[14] F. M. 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*, 2018.

[15] G. Holmes, A. Donkin, and I. H. Witten. Weka: A machine learning workbench. In *Proceedings of the Second Australian and New Zealand Conference on Intelligent Information Systems*, pp. 357–361, 1994.

[16] M. Kahng, D. Fang, and D. H. P. Chau. Visual exploration of machine learning results using data cube analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, p. 1. ACM, 2016.

[17] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[18] J. Krause, A. Dasgupta, J. Swartz, Y. Aphinyanaphongs, and E. Bertini. A workflow for visual diagnostics of binary classifiers using instance-level explanations. *arXiv preprint arXiv:1705.01968*, 2017.

[19] J. Krause, A. Perer, and K. Ng. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 5686–5697. ACM, 2016.

[20] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, pp. 126–137. ACM, 2015.

[21] T. Kulesza, S. Stumpf, W.-K. Wong, M. M. Burnett, S. Perona, A. Ko, and I. Oberst. Why-oriented end-user debugging of naive bayes text classification. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 1(1):2, 2011.

[22] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):91–100, 2017.

[23] S. Liu, X. Wang, M. Liu, and J. Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 1(1):48–56, 2017.

[24] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[25] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1222–1230. ACM, 2013.

[26] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[27] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. *arXiv preprint arXiv:1710.10777*, 2017.

[28] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pp. 115–124. Association for Computational Linguistics, 2005.

[29] K. Patel, N. Bancroft, S. M. Drucker, J. Fogarty, A. J. Ko, and J. Landay. Gestalt: integrated support for implementation and analysis in machine learning. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pp. 37–46. ACM, 2010.

[30] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison. Investigating statistical machine learning as a tool for software development. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 667–676. ACM, 2008.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[32] P. E. Rauber, S. G. Fadel, A. X. Falcao, and A. C. Telea. Visualizing the hidden activity of artificial neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):101–110, 2017.

[33] 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.

[34] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?:

Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144. ACM, 2016.

[35] C. Seifert, A. Aamir, A. Balagopalan, D. Jain, A. Sharma, S. Grottel, and S. Gumhold. Visualizations of deep neural networks in computer vision: A survey. In *Transparent Data Mining for Big and Small Data*, pp. 123–144. 2017.

[36] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg. Embedding projector: Interactive visualization and interpretation of embeddings. *arXiv preprint arXiv:1611.05469*, 2016.

[37] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):667–676, 2018.

[38] D. Tang, B. Qin, and T. Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1422–1432, 2015.

[39] Q. H. Tran, T. Lai, I. Zukerman, G. Haffari, T. Bui, and H. Bui. The context-dependent additive recurrent neural net. In *In North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[40] F.-Y. Tzeng and K.-L. Ma. Opening the black box-data driven visualization of neural networks. In *Visualization, 2005. VIS 05. IEEE*, pp. 383–390. IEEE, 2005.

[41] W. Yu, K. Yang, Y. Bai, H. Yao, and Y. Rui. Visualizing and comparing convolutional neural networks. *arXiv preprint arXiv:1412.6631*, 2014.

[42] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

[43] H. Zeng, H. Haleem, X. Plantaz, N. Cao, and H. Qu. Cncomparator: Comparative analytics of convolutional neural networks. *arXiv preprint arXiv:1710.05285*, 2017.

[44] X. Zhu. *Knowledge Discovery and Data Mining: Challenges and Realities: Challenges and Realities*. Igi Global, 2007.