Visualising the Training Process of Convolutional Neural Networks for Non-Experts

Michelle Peters[0000−0001−8884−730X]*, Lindsay Kempen[0000−0003−0556−0894]*, Meike Nauta[0000−0002−0558−3810], and Christin Seifert[0000−0002−6776−3868]

University of Twente, P.O. Box 217, 7500AE Enschede, The Netherlands
`cnn@michellepeters.eu`, `research@linths.com`, {`m.nauta`,`c.seifert`}`@utwente.nl`

Abstract. Convolutional neural networks are very complex and not easily interpretable by humans. Several tools give more insight into the training process and decision making of neural networks but are not understandable for people with no or limited knowledge about artificial neural networks. Since these non-experts sometimes do need to rely on the decisions of a neural network, we developed an open-source tool that intuitively visualises the training process of a neural network. We visualize neuron activity using the dimensionality reduction method UMAP. By plotting neuron activity after every epoch, we create a video that shows how the neural network improves itself throughout the training phase. We evaluated our method by analysing the visualization on a CNN training on a sketch data set. We show how a video of the training over time gives more insight than a static visualisation at the end of training, as well as which features are useful to visualise for non-experts. We conclude that most of the useful deductions made from the videos are suitable for non-experts, which indicates that the visualization tool might be helpful in practice.

Keywords: Explainable AI · Convolutional Neural Network · Visualisation · Dimensionality reduction · Image recognition

1 Introduction

Image recognition has become a great beneficial technology in various fields. It is being used for face recognition, visual search engines, e-commerce, healthcare and much more. To classify images, a convolutional neural network (CNN) needs to be trained. Neural networks have been found very useful for finding extremely complex patterns [12]. Although image recognition is thus a very impressive and useful technology, its complexity (and especially the high number of parameters) makes it difficult to understand how a trained model makes its decisions and why

* Both authors contributed equally.

it gives certain results. Getting a better understanding and interpretation of the model can help with debugging and improving the model, as well as generating trust for non-experts [1]. We focus on facilitating the non-expert: an individual without knowledge of neural networks who may not necessarily have a technical background. The only knowledge required for our tool is a basic understanding of classification. That is, a machine labelling input after learning from labelled training data.

The need to further interpret these models has given rise to numerous CNN visualisation techniques [6]. We use the *dimensionality reduction* method Uniform Manifold Approximation and Projection (UMAP) [11] to plot the neuron activations for every input in a single 2D graph. Our approach generates a visualisation with three plots: (i) a 2D plot of the test data with the input images as the data points, (ii) a simple 2D plot of the training data, and (iii) a line plot of the accuracy of both the testing and training data. Most importantly, we visualise throughout the CNN's training phase, generating a video frame for each epoch. We evaluate our tool by manually inspecting and interpreting the videos, and question the accessibility for a non-expert. The data set we classify contains grey-scale small sketches of various objects [5]. We limited us to 10 object categories.

We have released a tool[1] that intuitively visualises the training process of a convolutional neural network. The tool can be simply adapted to visualise other architectures of feedforward neural networks as well. We evaluated that the visualisations can provide non-experts with a general insight into how a CNN learns and how it can deduct the CNN's decisions.

2 Related work

Much research concerns improving computational sketch classification [3, 5, 15, 16, 19]. The complexity of classifying sketches lies within the aspects being very dependent on the person who drew it, such as their skill level. Using neural networks is a valid approach to classifying these sketches. As neural networks in general are not transparent and the task at hand is complex, this setup offers the optimal testing ground to evaluate if our visualisation tool can help non-experts gather insight of a neural network's workings.

2.1 Training phase visualisation

Most training visualisations show whether a model is improving during its iterations, rather than how it is training and why it makes certain decisions. One of these visualisations is the one proposed in [2], which proposes a way to visualise a novel type of training error curve on three levels of detail.

Since we want to show how the model trains, we have to look at more information than just the accuracy or error. Two tools that give insight into the

[1] https://github.com/Linths/TrainVideo

training process of a neural network are DeepTracker [8] and Tensorview [4]. DeepTracker is a proposed visual analytics system which helps domain experts in giving a better understanding of what has occurred in the CNN's training process. TensorView uses visualisations of different attributes of the CNN, such as the weights of the convolution filters, trajectories of the first two dimensions of the convolution weights, and the activations of the filters. Both DeepTracker and Tensorview specifically aid model-builders so they can improve their network. However, since they give a lot of information that is only relevant to an expert or model-builder, it also becomes less understandable for non-experts.

Therefore, when building our visualisation tool, we need to find the right amount of information to show, to give insight into the training process, without losing the understandability for the non-expert.

2.2 Dimensionality reduction

Our goal is to make a CNN's behaviour understandable for non-experts. This behaviour is determined by how every neuron activates for the given images. Because the number of neurons can grow large and the activations are essentially a set of formulas, showing all this information can overwhelm non-experts. One could show *all exact* neuron activity by plotting *all separate* activation values per input image, taking a dimension per neuron. However, to make a human-readable plot, tools such as t-SNE [9] and UMAP [11] reduce the dimensionality to 2 or 3. These new dimensions are compressed versions of the old dimensions. Dimensionality reduction (DR) approximates the neural network's behaviour. While the absolute DR values do not carry intuitive meaning, the relative values do: generally, if two images get similar DR values, the neural network behaves similarly to them. When it becomes clear what images the CNN considers similar, one can theorise about what aspects the CNN based its decisions on.

There is much variety in the existing DR approaches as well as the further functionality of these visualisation tools. While t-SNE has been a very popular DR implementation [9], the more recent DR implementation UMAP shows competition in visual and computational terms [11]. Comparative tests on benchmark data sets, such as COIL-100 [13], MNIST [7] and Fashion MNIST [18] – all comparable to our data – have shown that UMAP consistently visualises global structures significantly better than t-SNE and in considerably less time than t-SNE. Therefore, we use UMAP as our DR method.

Using t-SNE, Rauber et al. developed a 2D method that gives insight into how an artificial neural network learns [14] over time, by overlaying all t-SNE frames that were taken between training epochs, showing a t-SNE "trail" for every input entry. The imagery however becomes cluttered by compressing the whole timeline. One can also not see the additional information – such as the changing performance of the neural network – evolve together with the network's activations. There are also tools that create dynamic DR visualisations. TensorFlow's Embedding Projector, is an open-source online web application that allows for DR with rich functionality [17]. The tool shows the changing t-SNE plots while the neural network is being trained. The tool does not have

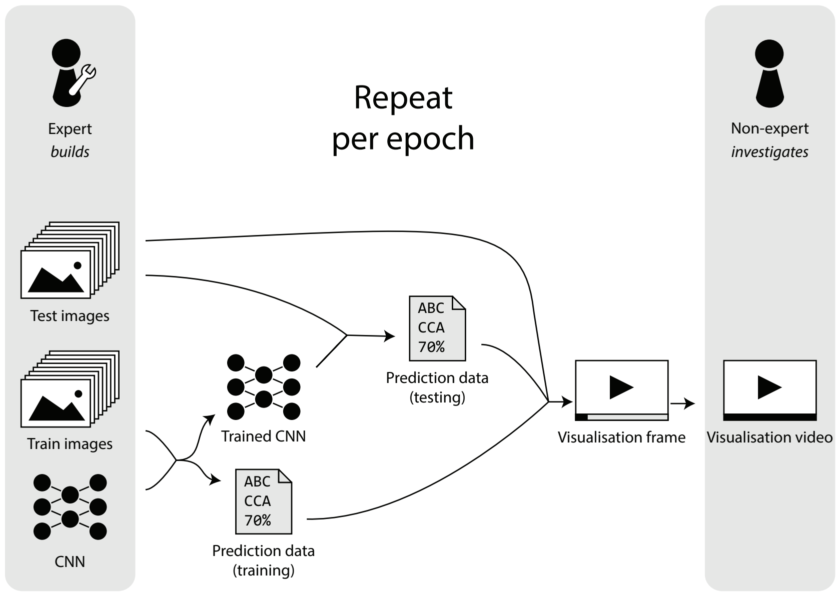

Fig. 1. Tool process and stakeholders

this dynamic visualisation for UMAP. The original images are included in the plots and they are colour-coded according to actual class. The tool is interactive but lacks certain information such as accuracy.

We conclude we will develop a tool that creates dynamic DR visualisations, including performance statistics, by creating videos with the inter-epoch DR plots as video frames. We use UMAP as DR method and include the input images in the plots. The user-friendliness and increased interpretability that the Embedding Projector offers with its rich interactivity could be an easy extension for our tool in the future.

3 Approach

In this section, we describe our approach for producing dynamic visualisations and interpreting them. To further facilitate the reproduction and expansion of this study, our source code is available on GitHub[1].

Simply put, our training visualiser takes in train and test images, runs them through a trainable CNN and outputs a video visualising the training process of the CNN. The main idea is to show how a CNN develops and improves on classifying unseen images after more training. This is why we require test data as well, even though we aim to visualise the *training* process. While a properly performing neural network is crucial to our approach, there are only a few further constraints. Our integrated CNN can be interchanged with any feedforward neural network with a fully connected layer. Note that the resulting visualization shows the original input images, thus if the network doesn't have images for input, only minor code tweaks are needed to show these as data points instead.

Figure 1 shows how the tool operates and how the stakeholders are involved. Per epoch of the CNN, it learns from the train data, classifies the test data and creates a visualisation frame. After all epochs, the tool creates a video from those frames.

4 Experimental Setup

For the experiment, we use our tool on one specific data set and one specific neural network architecture. We manually evaluate the resulting visualisations in detail.

4.1 Data Set

The data set we use was collected by having people sketch recognizable, specific subjects [5]. It consists of 250 categories with 80 sketches of 1111x1111 pixels each. They analyzed sketch recognition performance by humans, which was 73%, to compare to the computational sketch recognition.

We augmented the training data as follows: (i) horizontal flipping, (ii) random rotation between 3.5 and 20.5 degrees clockwise or counterclockwise, (iii) random shift between 20 and 100 pixels to right or left and to top or bottom (iv) random rescale between 0.75 and 0.9 or between 1.1 and 1.25. All test and train images are resized from 1111 x 1111 to 128 x 128, converted to tensor images and normalized according to a normal distribution with $\mu = 0.8$ and $\sigma = 0.2$ (determined empirically).

We limit our experiments to 10 classes to increase the CNN's performance and our visualisation's legibility. We created data sets, one corresponding to a difficult classification task (further denoted as *Hard*) and one corresponding to a simple classification task (further denoted as *Easy*). Table 1 shows our two data sets. *Easy* has arguably easily distinguishable images, e.g. apples and ants, while *Hard* contains very similar images: types of bears, birds and cars. This way, we can assess how our visualisation videos might provide understanding in different situations. Moreover, showing a non-expert both videos might give them additional insight into the CNN's training process. For every class, we split the data into 60 original train images (which amounts to 300 images after transformations) and 20 test images.

4.2 Neural network architecture

Previous work on the sketch data set yielded an accuracy of $\approx 70\%$ using 15 classes [3]. The authors used a CNN with two convolutional layers with ReLU activations, each followed by a max-pool layer, and two fully connected layers

Table 1. Overview of data sets. Similar classes are grouped for data set *Hard*

Easy **Easy to classify**	***Hard*** **Hard to classify**
airplane	bear (animal)
alarm clock	panda
angel	teddy-bear
ant	cloud
apple	sheep
banana	pigeon
basket	seagull
bed	car (sedan)
bell	suv
calculator	van

Table 2. CNN hyperparameters

Parameter	**Value**
# Classes	10
# Train images	3000
# Test images	200
Image width	128
Batch size	25
# Epochs	20
Learning rate	0.0001
FC2 size	50

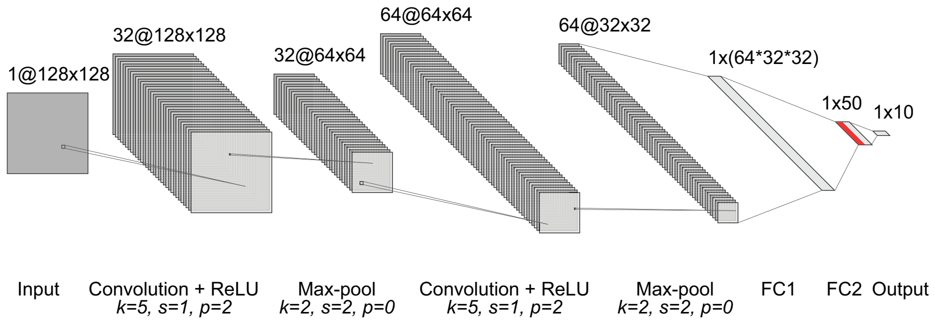

Fig. 2. Architecture of the CNN. The visualised component is red: the pre-activation of the second fully connected layer.

(FC1 and FC2). In this work we use a similar architecture, that is shown in Figure 2. The differences between our architecture and that of [3] are the amount of nodes in the FC layers, the amount of filters per convolutional layer, the stride in the last max-pool layer, and that their FC1 has ReLU activations, where ours does not. We apply dropout before FC1, and softmax on the 10 output nodes. All parameters are determined empirically, optimising test accuracy, speed and the visualisation (e.g., showing a clear clustering). Table 2 lists the final hyperparameters. We increased the number of epochs to show how the visualisation changes when the model overfits.

4.3 Visualisation

To make the training process visible and understandable for non-experts, we visualise the training data, testing data and accuracy of training and testing per epoch, as can be seen in Figure 5. For this, we take the pre-activation values of FC2 layer, as shown in Figure 2. To visualise all data points from an epoch, we save the original labels, predictions and pre-activation values of FC2 of each image after it passes the neural network.

For the test data plot, we want to display the images on the plot points, to make it visible why a certain image was perhaps misclassified, why specific classes are displayed as multiple clusters or why they appear close to specific other clusters. We add a mask to each of those images with the colour corresponding to the predicted label, and a red border if the image is misclassified, which is most useful when the actual class of the image is difficult to determine from the image and to give a quick overview of how many images within a certain cluster are misclassified. This way, the plot represents three important parts of information; (i) the actual class, which is represented by the image, (ii) the predicted class, represented by the image colour, and (iii) the neuron activity, represented by the location of the image in the plot figure.

The training data plot is almost the same as the test data plot, however, we only show simple dots instead of the images, since we want to focus on showing the effect of the training phase on the test data, and thus we want to keep the training data plot small.

We present the test accuracy and train accuracy, to see whether the CNN trains well, and to view the relation between the quality of the neural network and the UMAP visualisation.

To visualise the neuron activity, we use UMAP (with *#neighbours*=25) to reduce the pre-activation node values of the FC2 to two dimensions. To keep this dimensionality reduction consistent, we fix the UMAP seed. This way, when the values in the nodes are only slightly changed after an epoch, the x and y values of an image will also only slightly change, which makes for a visualisation video which is easy to follow.

However, since the absolute x and y values have no useful addition for non-experts when they already have the relative locations of the plot points, we decide to not show any values on the axis of the training and testing data plots. This way, we avoid confusing the user with useless information.

4.4 Evaluation

Our visualisation needs to give useful insights into the decisions that are made inside a neural network during the training process, while also considering the level of knowledge and understanding of a non-expert user. To evaluate whether our tool complies to these two conditions, we compare our dynamic visualisations with static visualisations, look at which conclusions can be drawn from our visualisations, and also look at which aspects can be confusing for non-experts and can lead to wrong conclusions.

5 Results

In this section we show and interpret our visualisations for *Easy* and *Hard*. We also examine whether the visualisations themselves, the observations and the

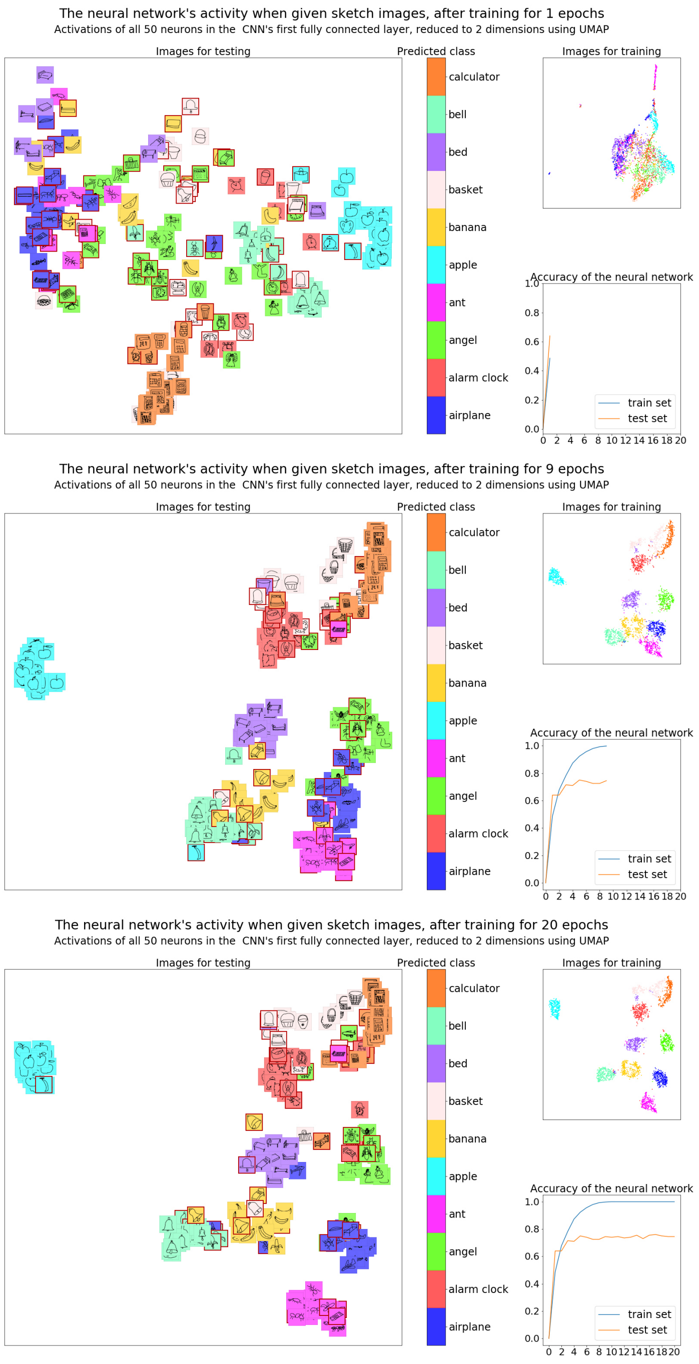

Fig. 3. Visualisation frames of epochs 1, 9 and 20 of *Easy*, with epoch 9 as the supposed start of overfitting.

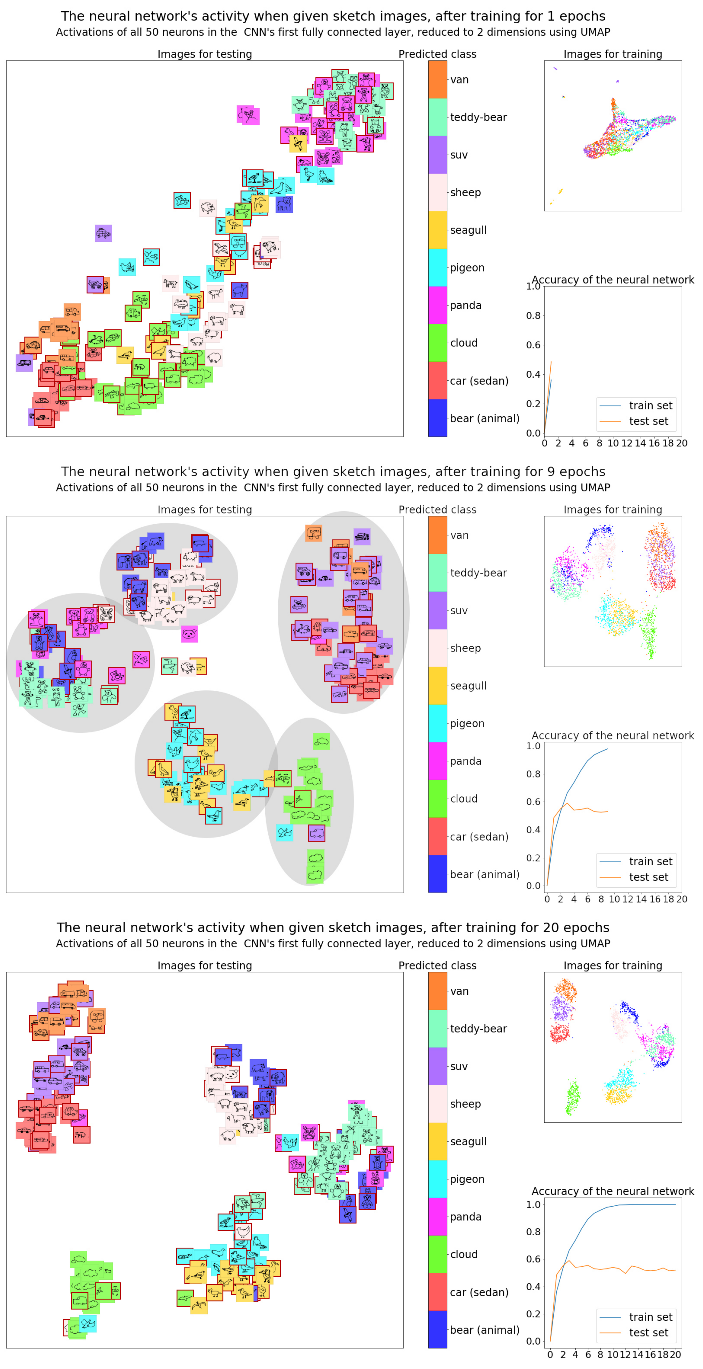

Fig. 4. Visualisation frames of epochs 1, 9 and 20 of *Hard*, with epoch 9 as the supposed start of overfitting. Clusters are manually highlighted for epoch 9.

interpretations are accessible for non-experts. While our resulting visualisation is a video, for clarity we will refer to video frames (Figures 3 and 4). We refer to a specific frame with a shorthand, e.g. the visualisation of *Easy* after 3 epochs is called "*Easy*-3". The two output videos and the raw frames are available online[2].

5.1 Accuracy of the model

We build and tweak the neural network based on *Easy*, which consists of only 10 classes. With this data subset, we reach over 70% accuracy. We also test the accuracy of our model on a data subset of 15 classes. For this, we use *Easy*, with the addition of the crab, pineapple, snail, sponge bob and squirrel classes. Before the model starts to overfit, at epoch 7, we achieve around 74% accuracy. However, despite our neural network being resistant to more than 10 classes, our visualisation becomes cluttered and less understandable when it has to present more classes. Table 3 gives an overview of the accuracies of our model.

Table 3. Accuracy of our model with the different data subsets.

Dataset	Number of classes	Accuracy
Easy	15	74%
Easy	10	70%
Hard	10	50%

5.2 Trends observed

We observe several trends in the visualisations for *Easy* and *Hard* after running for 20 epochs.

Throughout the whole timeline, *Easy* and *Hard* show clustering in the train and test plots. Every frame contains several clusters of mainly the same actual and/or predicted class. Over time, these clusters separate more. The video[2] shows relatively smooth transitions; it maintains the same data structure with some slight to moderate shifts. Unexpectedly, the visualisation seems to flip horizontally or vertically at a few points, but even then the relative positioning seems consistent. Interestingly, a specific structure is only present in the train plots for *Easy*-1 and *Hard*-1. We refer to the very distanced and closely knit groups of just several points. We found out through simple tests that this happens when there is too little coherence in the data to properly apply DR – logical to happen after just one epoch.

Generally, the image distancing in the test plot seems logical and easily interpretable. Firstly, in *Easy*-4 and later, the apple cluster moves far away from all the other points. This means the CNN has found a very specific way to predict apples: the activation behaviour for apples is very different than for the other

[2] https://doi.org/10.5281/zenodo.3525091

Table 4. Types of prediction-placement combinations

Prediction	Plot placement		
	Predicted class	Actual class	Seemingly random
Correct	A		B
Incorrect	C	D	E

classes. As almost the whole cluster is predicted correctly, the strong pattern the CNN learns is a useful one. By inspecting all the images, one can speculate what was used as the discriminating characteristic for apples. We hypothesize it is a combination of the round and simple form and the large whitespace.

Secondly, in *Hard*-7 and later we find clusters that contain two to three predicted classes. The partitioning almost completely matches our expectations (Figure 1), grouping cars and bears. However, the tool does not group sheep with clouds. To the contrary, the sheep are grouped with bears. This indicates the CNN has ease separating sheep from clouds, but not from bears. We believe the CNN focuses on legs which is a strong differentiator between sheep and clouds, but not for sheep and bears. The bears that are matched with the sheep do look similar as they are on all fours. Moreover, the CNN clearly distinguishes this bear variant from bears that are sitting, standing on hind legs or lack a body.

The train accuracy signals the model overfits around *Easy*-9 and *Hard*-9. The clusters in the test and train plots continue to separate, even for *Easy* where the visualisation seems to have stabilised. *Easy*-9 and *Easy*-20 are closely similar, but *Easy*-20 shows a bit more separation. For *Hard*, these differences are bigger. While *Hard*-20 has a significantly clearer separation of clusters than *Hard*-9, the test accuracy does not actually increase. Note that the clusters here concern images of the same *predicted* class. In reality, the model just polarizes its activation behaviour. It becomes stricter in enforcing the patterns it claims to see, but not the patterns that actually exist. It is important to keep in mind that more exaggerated clustering does not always indicate a better model.

5.3 Situation types

There are multiple ways an image can appear in our visualisations. The background colour, thus predicted class, can either be correct or wrong. And the position of the image can differ. Table 4 shows the different types of combinations, and in Figure 5 we point out some examples of these types. For A, we show a correctly classified apple, which is also placed correctly. For B, we point out an alarm clock which is classified correctly but not placed with the other alarm clocks. C is represented by a basket, classified as an apple, which is also placed with the apples by UMAP. For D, we show two examples; bananas classified as bells, and bells classified as bananas. However, UMAP places both with their actual class rather than their predicted class. That means that UMAP's DR seems to be better at identifying these images than the neural network's final layers, while they are given the same input: the pre-activation values of FC2. Lastly, E

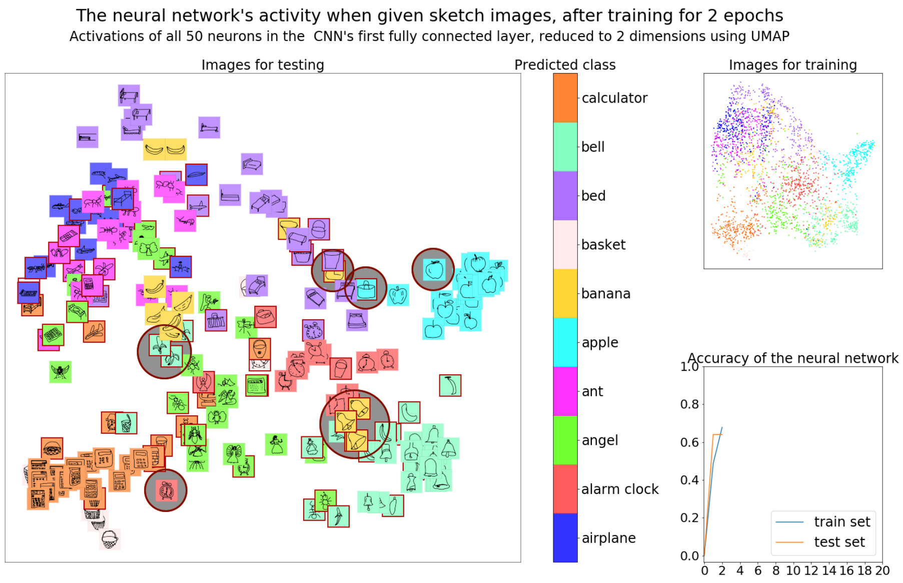

Fig. 5. Visualisation of *Easy* after 2 epochs. Several datapoints are highlighted with their situation type.

is represented by a misclassified alarm clock, which is neither placed with the other alarm clocks, nor its predicted class. There are also middle grounds of the above-mentioned situations. For example, a combination of C and D. When an image is classified incorrectly, but both the actual and predicted class are clustered together, the image can be placed with both its actual and its predicted class images. This means that the classes are probably really similar, such as the types of cars in *Hard*.

5.4 Approachability for the non-expert

Our tool visualises only information that is useful for a non-expert. For the test data, we visualise the actual class, the predicted class, the neuron activations (which estimates the class compatibility), and the overall accuracy. The same holds for the training data except for the actual classes. Since these are all features that are easy to understand, also for non-experts, the tool seems very approachable for people with little to none background knowledge about neural networks.

There are some aspect that might be confusing for non-experts, such as situation types B, D and E from Section 5.3. These situations are less intuitive since non-experts might not fully understand the difference between the output of the neural network, the activation values of the neurons, and the output of UMAP. When these do not agree on an image, it can become confusing for the user as to why an image is classified as one thing, but clustered with another. Other confusing factors are the occasional flips of the visualisations between epochs and the misleading clustering of a increasingly overfitting model (mentioned in Section 5.2).

Visualising the training process over time – rather than after it has finished – can give insights into which features are recognized first, and when certain images are clustered much quicker and further away from the rest of the classes. Also, seeing certain classes cluster together or next to each other can provide the knowledge that the neural network sees them as quite similar.

6 Discussion

In this section, we lay out factors that have influenced our research and the consequences for the tool's quality and our research's validity.

Ideally, our visualisation would be insightful for CNNs of various performance levels. We examined results for CNNs scoring around 50% and 70% test accuracy. Because of the challenging dataset, we were unable to get a higher accuracy and check if our visualisation would then be insightful. The data set we used is arguably hard to classify, as it only contains 80 instances per class and the classes can still contain quite different drawings, e.g., the bears mentioned in Section 5.2.

The CNN overfits rather quickly. It allows for only eight visualisation frames without overfitting. More frames would facilitate smoother transitions and might be more understandable and less overwhelming for non-experts. To add, detail might be lost by the lack of intermediate epochs.

Dimensionality reduction in itself is flawed. A datapoint can have false neighbours or missing neighbours due to the "compression" of dimensions. This could make our visualisation misleading. We have not tested other dimensionality reduction methods than UMAP and we have only roughly optimized the UMAP settings to minimize such mistakes. Also, we have not counted any of these mistakes in our current visualisation, but we did see UMAP struggling to apply DR in *Easy*-1 and *Hard*-1.

Because UMAP refits on the data for every epoch, the visualization frames sometimes appear to be flipped horizontally or vertically. This makes the transitions between our frames less smooth. A solution to this is applying DR only once, on the data of all epochs. We can then still visualise every epoch individually by only displaying the appropriate data points. This approach could however lower the DR quality as it has to fit significantly more data. Additionally, it would disable "live visualisations": visualising while running the CNN.

We have not used a standardized method to evaluate understandability, e.g. user studies. Our evaluation method concerned just two test cases and was not quantified. Visual inspection is not completely objective as well. Though tasks as determining clusters have no objective answer anyway: there's no silver bullet. All in all, our methodology could not produce statistical claims.

7 Summary and Future work

We have built a CNN for the data set of human-made sketches with an accuracy of over 70% for 10 classes when using the easily distinguishable data subset (*Easy*). For the data subset with classes that seem very similar (*Hard*), the accuracy is slightly lower, due to the classes being more similar. During the training of this convolutional neural network (CNN), we have created video visualisations which provide useful information such as the actual class, predicted class and similarity between classes. Despite some situations which can be difficult to understand by a non-expert, the videos are overall quite interpretable even by people without prior neural network knowledge.

Many options for improvement and addition have been left for the future. Most importantly, we would like to apply user studies with non-experts in our evaluation for a better scientific justification. That way, we can evaluate how understandable the visualizations are in practice. Another useful addition for the evaluation would be a thorough comparison of our tool with several other neural network visualisations. It would be interesting to at least include methods that are focused on the training process and/or DR, such as [2,4,8,9,14,17]. With user studies and tool comparisons, one can more properly assess the added value of both the DR and training process aspect of the visualizations. Furthermore, it could be interesting to see whether fitting the UMAP visualisation to the result of the first epoch, or to all epochs after training, makes the visualisation more easy to follow since clusters might not jump between epochs. Another useful addition could be to add an automatic cluster detector, which can detect the types in Table 3. Some methods can detect false or missing neighbours in DR [10], which would add a quick overview for the user of which data points are classified correctly but displayed within the wrong cluster. Moreover, a small but very useful tool to give concrete and objective insight into how distinguishable the classes are is a confusion matrix of the actual classes versus the predicted classes. Lastly, the tool could be made into an interactive viewer. By being able to switch certain options on or off, it could provide more information without too much clutter. It could then also enable a 3D visualisation.

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: A survey on explainable artificial intelligence (xai). IEEE Access **PP**, 1–1 (09 2018). https://doi.org/10.1109/ACCESS.2018.2870052
2. Becker, M., Lippel, J., Zielke, T.: Gradient descent analysis: On visualizing the training of deep neural networks:. In: Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. pp. 338–345. SCITEPRESS - Science and Technology Publications (2019). https://doi.org/10.5220/0007583403380345
3. Chandan, C.G., Deepika, M., Suraksha, S., Mamatha, H.R.: Identification and grading of freehand sketches using deep learning techniques.

In: 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI). pp. 1475–1480 (Sep 2018). https://doi.org/10.1109/ICACCI.2018.8554920

4. Chen, X., Guan, Q., Liang, X., Lo, L.T., Su, S., Estrada, T., Ahrens, J.: Tensorview: visualizing the training of convolutional neural network using paraview. In: Proceedings of the 1st Workshop on Distributed Infrastructures for Deep Learning - DIDL '17. pp. 11–16. ACM Press (2017). https://doi.org/10.1145/3154842.3154846
5. Eitz, M., Hays, J., Alexa, M.: How do humans sketch objects? ACM Trans. Graph. (Proc. SIGGRAPH) **31**(4), 44:1–44:10 (2012)
6. Hohman, F.M., Kahng, M., Pienta, R., Chau, D.H.: Visual analytics in deep learning: An interrogative survey for the next frontiers. IEEE transactions on visualization and computer graphics (2018)
7. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
8. Liu, D., Cui, W., Jin, K., Guo, Y., Qu, H.: Deeptracker: Visualizing the training process of convolutional neural networks. ACM Transactions on Intelligent Systems and Technology **10**(1), 1–25 (Nov 2018). https://doi.org/10.1145/3200489
9. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
10. Martins, R.M., Coimbra, D.B., Minghim, R., Telea, A.C.: Visual analysis of dimensionality reduction quality for parameterized projections. Computers & Graphics **41**, 26–42 (2014)
11. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426 (2018)
12. Müller, B., Reinhardt, J., Strickland, M.: Neural Networks: An Introduction. Springer, Berlin/Heidelberg, Germany (2012). https://doi.org/10.1007/978-3-642-57760-4
13. Nene, S.A., Nayar, S.K., Murase, H.: Columbia object image library (coil-20. Tech. rep., Columbia University (1996)
14. Rauber, P.E., Fadel, S.G., Falcao, A.X., Telea, A.C.: Visualizing the hidden activity of artificial neural networks. IEEE transactions on visualization and computer graphics **23**(1), 101–110 (2016)
15. Seddati, O., Dupont, S., Mahmoudi, S.: Deepsketch: Deep convolutional neural networks for sketch recognition and similarity search. In: 2015 13th International Workshop on Content-Based Multimedia Indexing (CBMI). p. 16 (Jun 2015). https://doi.org/10.1109/CBMI.2015.7153606
16. Sert, M., Boyac, E.: Sketch recognition using transfer learning. Multimedia Tools and Applications (Jan 2019). https://doi.org/10.1007/s11042-018-7067-1
17. Smilkov, D., Thorat, N., Nicholson, C., Reif, E., Viégas, F.B., Wattenberg, M.: Embedding projector: Interactive visualization and interpretation of embeddings. arXiv preprint arXiv:1611.05469 (2016)
18. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
19. Zhang, X., Huang, Y., Zou, Q., Pei, Y., Zhang, R., Wang, S.: A hybrid convolutional neural network for sketch recognition. Pattern Recognition Letters (2019). https://doi.org/10.1016/j.patrec.2019.01.006