Samples Classification Analysis Across DNN Layers with Fractal Curves

Adrien Halnaut, Romain Giot, Romain Bourqui, D. Auber

Samples Classification Analysis Across DNN Layers with Fractal Curves

Adrien Halnaut[1], Romain Giot[1][0000−0002−0638−7504], Romain Bourqui[1][0000−0002−1847−2589], and David Auber[1][0000−0002−1114−8612]

Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400 Talence, France
{firstname,lastname}@u-bordeaux.fr

Abstract. Deep Neural Networks are becoming the prominent solution when using machine learning models. However, they suffer from a black-box effect that renders complicated their inner workings interpretation and thus the understanding of their successes and failures. Information visualization is one way among others to help in their interpretability and hypothesis deduction. This paper presents a novel way to visualize a trained DNN to depict at the same time its architecture and its way of treating the classes of a test dataset at the layer level. In this way, it is possible to visually detect where the DNN starts to be able to discriminate the classes or where it could decrease its separation ability (and thus detect an oversized network). We have implemented the approach and validated it using several well-known datasets and networks. Results show the approach is promising and deserves further studies.

Keywords: Deep Learning · Visualization · Explainable Artificial Intelligence Glyph Definition.

1 Introduction

Deep-learning [20] based approaches are used in various contexts and dominate most historical methods, especially for classification problems. Even when datasets are not large enough to train Deep Neural Networks (DNN), it is possible to use transfer learning with a pre-trained DNN by fine-tuning [10] it, or by extracting features and feeding them to a conventional classifier [35]. Any DNN corresponds to a graph of computational blocks: each block processes the output of one or several previous ones through a simple function parametrized by many weights. Such weights are data-dependent and computed during the training phase.

The black-box feeling is one of the largest issues. Indeed, even if each block is individually well understood mathematically, its behavior depends mainly on the training data (*i.e.*, their impact on the learned weights). As a consequence, one cannot know what treatment these blocks are doing. However, it is well admitted that the first layers extract low-level features, while the latest ones extract high-level features specific to the application problem [8]. Two non-exclusive strategies can help to open this black box. (i) *Explainable deep-learning* where the architecture of the DNN emphasizes its explainability [40], even if

it could negatively impact its performance, and (ii) *Interpretable deep-learning* where additional processes extract information by computing a more explainable model [41], computing some saliency information [5,2] or using information visualization techniques [14].

This paper presents a new method for interpretable deep-learning based on information visualization techniques. The task to solve corresponds to the analysis of the data classification over layers. It aims at analyzing how all input samples of a dataset are globally treated by any part of the network. In opposite to most papers of the literature on attribution-based methods, the focus is not for a specific sample. The method *allows focusing on* successive layers that better (or worst) discriminate the samples. It also displays the complete network and the data behavior for each of its computational blocks.

It originality relies on the fact it focuses on both all samples and full architecture and has the advantages of (i) using less screen space than existing methods despite the amount of information to display; (ii) fitting to any network that can be represented as a directed acyclic graph; (iii) using the same encoding for input data, inner blocks and final result.

The remaining of the paper is organized as follows. Section 2 presents related works in visualization for CNN and space filling curves. Section 3 describes the proposed method. Section 4 provides the details of the experimental protocol. Section 5 discusses the results and provide directions for future work. We finally draw conclusions in Section 6.

2 Previous Works

Our proposed method aims at visually interpreting how DNN behave using a space-efficient method. For this reason, this section firstly presents previous works on deep neural networks visualization then focus on dense pixel oriented methods.

2.1 Visualization for the Interpretation of Deep Neural Networks

Visualization plays an important role in the tools used to help to explain or interpret how deep models work and to reduce their black-box feeling [14]. Different purposes have been achieved in the literature.

Some works focus on *single views* that can be reused in other works or embedded in more complex applications. GradCam [30] aims at generating a heatmap for a single input to highlight the spatial location that greatly supports the final decision. It is computed thanks to the gradients from the logit of a target class up to the latest convolutional layer and can be straightforwardly visualized and understood with a heatmap when the input feature is an image. Other methods rely on different concepts to achieve the same objective such as LRP [5] on the concept of relevance or other work [2] that only uses information collected during the forward pass. Instead of focusing on a single input sample, it is also possible to focus on the complete dataset. Some use Sankey-diagram analogy [11] to highlight the processing flows. Others project activations obtained

at a specific layer in a 2d space to verify how the network sees the data at this specific point [28]. Such an approach is also common in the literature using T-SNE projection [24]; however, an important drawback remains: the representation is not space efficient and there is no guarantee that overlap does not occur. Our proposed method solves these two issues.

Other works create *applications for educational purpose* to visually explain how some specific deep systems perform. For example, Tensorflow playground [33] focuses on simple DNN, CNN 101 [37] focuses on CNN, Ganlab [16] focuses on Gan system and Adversarial Playground [27] illustrates the concept of adversarial examples. Even if they are visually appealing, these systems can hardly be used for industrial scenarios.

In opposite, several complete tools treat *industrial problems*. Some of them are generalist enough to be used in almost any scenario, such as Activis [15] (that focuses on the visualization and comparison of activation of a single selected layer), while some others are restricted to some specific networks or evaluation scenarios. CNNVIS [23] is tailored for CNN and uses a visualization that relies on aggregation of layers (not all layers are depicted), filters (filters that behave similarly are grouped) and data (a subset of the samples are depicted). DQNVIZ [36] has been designed for Deep Q-Network explanation in the specific context of Atari play.

2.2 Hilbert Curve in Information Visualization

Dense pixel-oriented methods aim at improving both the data-ink ratio and the visualization size by displaying a unit of information on a single pixel while avoiding unused pixels. Keim reviewed various pixel-oriented visualizations [18] and asserts that space-filling curves, such as the Hilbert one [13], are among the bests to project ordered elements in a screen space while preserving the distance of the one-dimensional ordering in the two-dimensional arrangement. Blanchard *et al.* [6] have shown that to display images, reduced to a one pixel representation, on an Hilbert curve produces coherent and identifiable clusters. Auber *et al.* [3] have also shown the interest of such visualization, when complemented by tailored interaction techniques, to explore datacubes of several dozen of millions of elements. Since these previous successes, we have selected the Hilbert curve to project our data in a square; a curve of order n contains 4^n elements [3].

3 Proposed Method

Fig. 1 describes the proposal with the "Nested blocks and guidelines model" [25] among various description levels: *domain* (who is concerned by which problem), *abstraction* (which data is used or generated to solve which task), *technique* (which methods are used) and *algorithms* (how these methods are implemented).

Additionally, Fig. 2 lists the successive steps involved in the method. The requirements of the proposed method are: to be *space efficient* (R1) while displaying information from *all samples* (R2) in *all layers* (R3) of the network to solve the *task "classification quality analysis over layers"* (R4).

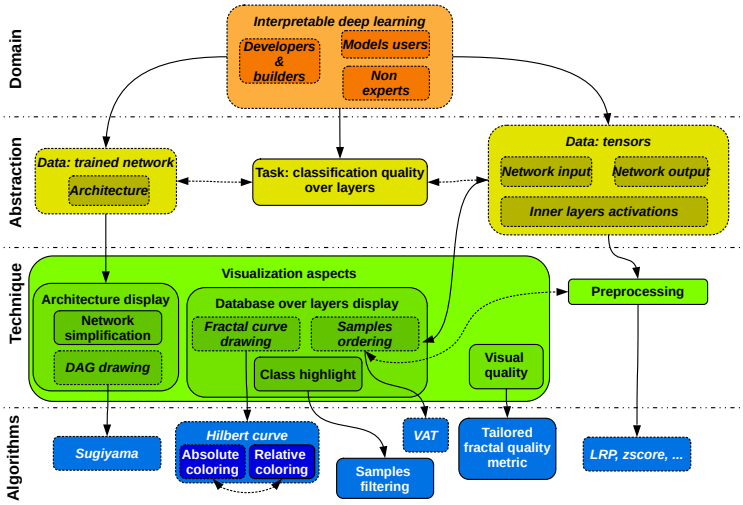


Fig. 1. Nested blocks and guidelines [25] representation of the proposed method. Dotted italic blocks corresponds to existing ones; plain straight blocks are defined in the work.

3.1 Domain Level

The proposed method fits the needs of *networks designers* and *trainers* that want to verify *how* the data is *grouped* by the various *layers* of their *classification network*. From the *analysis* of these groupings, they could infer *hypotheses* that aim at being verified with other techniques. Such hypotheses are related to input sample properties and network errors. *Non experts* would better understand how DNN work by looking at the representation of simple networks and datasets.

3.2 Abstraction Level

The proposed method considers an already trained DNN N with a compatible test dataset D_{test}. N is a network (*i.e.*, graph) of operations (*i.e.*, nodes) $N = (O, E)$. Its sources $s_\bullet \in O$ are the identity function on data input (*i.e.*, samples) and its sinks $t_\bullet \in O$ are its outputs (*i.e.*, classes probability). N has multiple sources for a multi-modal system, but always a single sink as we are restricted the use case of standard classification. The other nodes $o_\bullet \in O \setminus \{s_\bullet \cup t_\bullet\}$ correspond to any operations (*e.g.*, convolution, pooling, etc) that compose N; operations related to optimization (*e.g.*, dropout) are not included. The edges $E = O \times O$ model the flow of data over the operations of the network (*i.e.*, they link successive layers).

Each sample $d_i \in D_{test}$ is fed into the network and the output (*i.e.*, activations) of each operation o^j is stored in a_i^j; we assume operations are ordered depending on the execution flow. These activations consist of tensors whose order depends on the underlying operation and whose dimensions size depends on the input data of the network. Each operation o^j consumes at least one result $a_i^k | k < j$

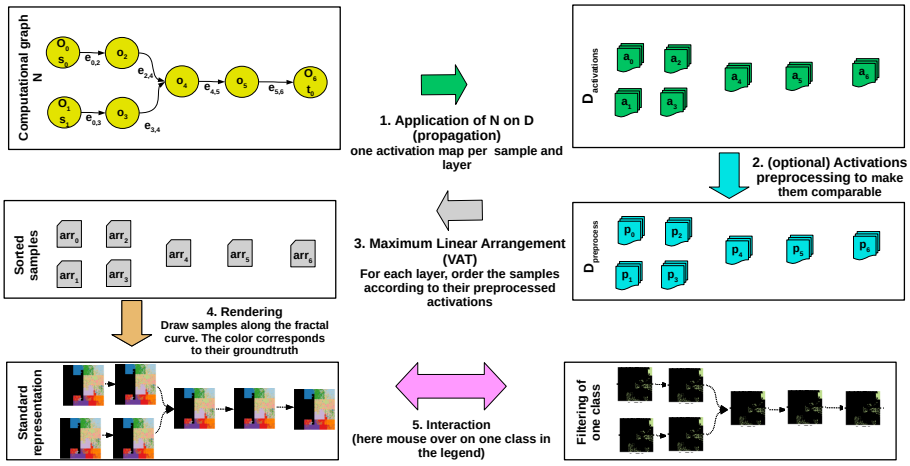


Fig. 2. Summary of the proposed method. Dataset D is fed to the network N. All activations are collected, eventually preprocessed, and finally ordered at each layer. The ordered samples are drawn along a fractal curve at each operation of the network that is placed on the screen using a graph drawing method.

computed by a previous operation except for the sources where $a_i^\bullet$ corresponds to the raw data (of the targeted modality in a multimodal scenario). Thus, a sample d_i is represented by a set of activations $A_i = \cup_j \{a_i^j\}$ and the complete dataset D_{test} is represented by an ensemble of sets of activations $D_{activations} = \cup_i \{A_i\}$.

The activations can be optionally preprocessed to fall within compatible domains as their domain is not controlled: $D_{preprocess} = \bigcup_i \bigcup_j \left\{ Preprocessed\left(a_i^j\right) \right\}$. This preprocessing method is a parameter of the workflow.

3.3 Technique Level

As the method aims at displaying (a) the dataset and its *groundtruth* (R2), (b) the *architecture of the network* (R3) and (c) its impact on the *complete dataset* (R4), we propose an encoding relying on both $D_{preprocess}$ and N.

Groundtruth encoding. The groundtruth of the dataset is depicted with a legend where each class is represented by a colored rectangle followed by a black text. Obviously, the text is the name of the class and the color represents this class in the samples encoding.

Network Encoding. It is straightforward to layout N operations with a graph-drawing algorithm tailored for Directed Acyclic Graphs (networks are always DAG). Such technique is common in the literature [15,38] and aims at computing

the coordinates of each node (operation) in a plane while emphasizing the order of operations in the computing flow. Each node is depicted by a glyph that represents the whole dataset as viewed by the network at this specific operation. Thus, a specific encoding is used to map the activations $\bigcup_i \{Preprocessed(a_i^j)\}$ of each node o^j in the screen space.

Like Ganlab [17], a dotted line is drawn between nodes that represents consecutive operations; the flow of data is revealed by the dots moving in flow direction. Some networks can be very deep with successive layers that do not bring additional information because they consist of data reordering. We allow the user to request the visualization of a simplified network where the corresponding nodes are removed (thus, their successors are linked to their predecessors), as such information brings noise to the representation. No special encoding is used to represent this information shrinking.

Samples encoding. As already explained, we have chosen a pixel-oriented technique that relies on fractal curves (R1). For a given node o^j, a maximal linear arrangement method is used to order the representation a_i^j of each sample d_i in such a way that close samples are positioned closely in the ordering according to a *distance* function. We assume close samples in the output space of o^j corresponds to samples treated similarly by the network (*i.e.*, considered to be similar). Once the samples are ordered, they are projected into a discrete pixel grid using a fractal curve that respects proximity relations. This way, screen space usage can be maximized (1 pixel per sample) and we are assured that close samples are drawn closely on the screen (however close pixels on the screen are not necessarily close in data space). Two visual encodings can represent this curve. The first one, *absolute coloring*, explicitly draws samples of each class with the same color. The second one, *relative coloring*, uses a gray-scale to emphasize label difference between adjacent nodes and identify zones where different labels are present. It can be used *de facto* when the number of classes is higher than the number of discernable colors by a human. When using the *absolute coloring* scheme, the user can choose to only visualize a specific class to analyze the spread of its samples over the layer. The name of the layer is written above its fractal representation, and a quality metric (see later) is written below it.

3.4 Algorithms Level

The model topology is drawn using the well-known Sugiyama [34] algorithm and each node is depicted with a specific fractal-based glyph that represents the ordered samples. The Euclidean distance is used to compare the activations generated for all the samples on the same operation. It reflects the dissimilarity between samples in the Euclidean space; we consider that each neuron activation has the same impact as others in the full network processing. These distances are then compiled into a $n \times n$ sized distance-matrix, n being the number of compared samples. In real use case, some neurons have more impact on the final prediction than others. Some pre- or post-processing methods, such as

the LRP [5] methods as done in [11], can be applied to the activation maps in order to reflect that behavior. However, we decided not to apply those methods because of the unsure interpretation on model topologies using branches, such as ResNet [12] or our chimeric *DoubleLeNet5* (section 4.1). Using dissimilarity matrix ordering methods [4], data can be ordered in a queue with similar elements placed next to each other using their dissimilarity matrix. By using the VAT algorithm [31] on the dissimilarity matrices, we found a progressive definition of the clusters (or "black squares" as shown in the original paper) reflecting the progressive recognition by the model over the layers we attempt to show. The order computed by this algorithm can then be applied on a 1d-space to display similar data indexes next to each other. Using a fractal curve, we transformed this 1d-space into a 2d-space which is more suitable for data visualization. The fractal curve chosen to map each sample into a pixel-grid is the Hilbert curve [13] because of its ability to place points in a discrete space (this is not the case of Gosper curve [9]) and the absence of "jumps" in the curve (this is not the case of the Z-order curve [26]) which ensures that two consecutive samples are adjacent. The order in which each sample is positioned is following the same order computed by VAT on the previous step. When the number of test samples is lower than the number of pixels available in the curve, we skip half of the missing positions in the beginning of the curve (and thus half of the missing positions at the end of the curve); which gives a hole in the curve.

In the absolute coloring, each pixel sample is being colored according to its ground-truth class, which is different for each class. In the relative coloring, the colors depend on the number of similar labels for the pixel of interest in its sample ordering. That gives three possible values (0 for an outlier with no neighbors of the same class, 1 for a previous or next label different, and 2 when the three successive samples are of the same class). The absolute colors come from a palette of diverging colors while the relative colors or black (0), gray (1) and white (2). Computing in the ordering space instead of the picture allows to no highlight the visual border inherent to the fractal curve. Placing the cursor on a class in the legend selects this specific class and draws only its samples with the appropriate absolute color.

The machine learning community provides various evaluation metrics (*e.g.*, accuracy or cross-entropy) to evaluate the quality of the network by comparing its output to a ground-truth. By definition, they cannot be applied at each layer, but we still need to provide hints to the user of their efficiency. We have defined a quality metric, based on the quality of the visual representation of a layer, which counts the number of neighbors of a given pixel that are of the same color (*i.e.*, the number of samples that belong to the same class). We normalized it between 0 and 1 to ease its comparison (however, as the normalization does not consider the mandatory borders, 1 is an unreachable value). We assume that to quantify the quality of the visualization is strongly related to the ability of the layer to separate data.

Table 1. Number of layers, parameters and activations per sample for each network.

Network	Layers	Parameters	Activations
LeNet5	10	1 182 006	26 378
DoubleLeNet5	18	1 646 370	54 570
VGG16	18	14 714 688	308 244

4 Experimental Protocol

Several scenarios, that rely on a *test dataset* and a *trained network* (Table 1), illustrate the efficiency of the proposed method.

4.1 Scenarios

Datasets.

- *Mnist* [19] is a standard dataset used in handwritten recognition from 28×28 grayscale images. Even simple networks are able to perform almost perfectly on this 10-class dataset. We use it to illustrate what happens with an easy dataset.
- *FashionMnist* [39] shares a similar distribution than *Mnist* and is composed of images of clothes instead of digits. Classification performance is usually lower than with *Mnist*. We use it to illustrate what happens with an average difficulty dataset.

Both datasets are composed of 60 000 samples to train the model and 10 000 samples to evaluate the model.

Networks.

- *LeNet5* [21] is a simple and historical CNN that provides good accuracy results on *Mnist*. Its topology is simple enough to get a grasp on how data is being transformed across the model. It is also easy to train with its low parameter count, but that simplicity comes at the cost of lower accuracy results in more complex recognition tasks.
- *DoubleLeNet5* is a chimeric network we have created to illustrate the ability of the system to handle networks with several branches. It consists of two *LeNet5*-like networks that process in parallel two versions of the given input data, one as-is and one with image rotation applied. Those sub-models are concatenated after the reduction to a 128-values long vector by a Dense-type layer from each branch. This model targets the same kinds of data as *LeNet5*, with a minor performance gain.
- *VGG16* [32] is a deep CNN usually used on complex datasets composed of large color images, with a thousand recognizable classes [29]. Its robustness allows it to reach fairly good accuracy results on target tasks, but comes with a heavy computation cost and cannot be trained in a reasonable amount of

time on standard computers. In this paper, the convolutional layers of the *VGG16* model are already pre-trained with the ImageNet dataset, and are not altered when training the prediction layers.

Couples of Network and Dataset. We have selected meaningful combinations of network and dataset.

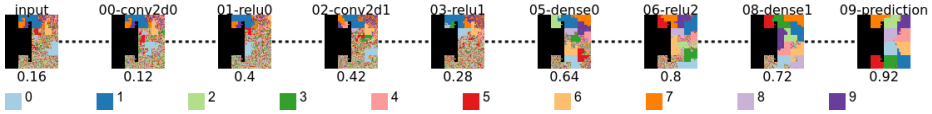
- *Easy scenario*: *LeNet5* uses *Mnist* which illustrates a well performing system.
- *Generalization scenario*: *LeNet5* uses *FashionMnist* which illustrates a system with more errors.
- *Problematic scenario*: *LeNet5* predicts *FashionMnist* but is trained with *Mnist* which illustrates a system that does not perform well.
- *Simplification scenario*: *VGG16* is processing *Mnist*. It illustrates the use of a complex network to solve a simple task.
- *Realistic scenario*: *DoubleLeNet5* predicts *FashionMnist* which illustrates a network with branches. It illustrates the correct usage of transfert learning for a simpler task.

4.2 Implementation and Execution Infrastructure

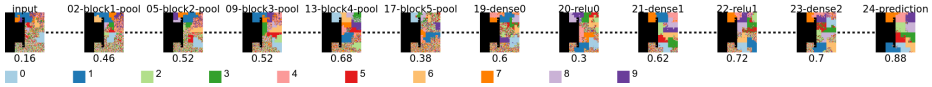
The TensorFlow framework is solicited along with Keras to train the studied models with said datasets. [7,1] The intermediate computations between each layer, seen as high-dimensional vectors leading to potentially very large data, are saved into a 52-machines large cluster handling the dissimilarity matrix computation, which makes use of a large pool of memory (around 2 Terabytes in our infrastructure). The resulting matrices are small enough (for our experiments) to fit and be processed on a recent laptop, equipped with a 9th generation Intel i9 CPU, a nVIDIA Quadro T1000 GPU and 16GB of RAM capacity. The matrix manipulations of the original VAT algorithm are implemented using the ArrayFire library for their efficient matrix computation abilities. This part of the process only produces the data for the visualization tool and thus can be seen as a backend infrastructure. Fractal images are then generated by relying on the Rust *hilbert* library. The visual and interactive part corresponds to an `HTML` application written in `Typescript` relying on `D3.js` for the visualization, `D3-dag` for the Sugiyama implementation and `webpack` for the build system.

5 Results and discussion

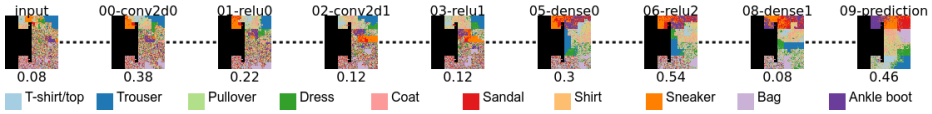
The complete results are accessible and best viewed online (images are strongly undersized in the paper where 1 pixel represents several samples) at the following address: `https://pivert.labri.fr/frac/index.html`. Fig 3 depicts still-resized representations of the proposed method for several scenarios, while their confusion matrices are presented in 4.



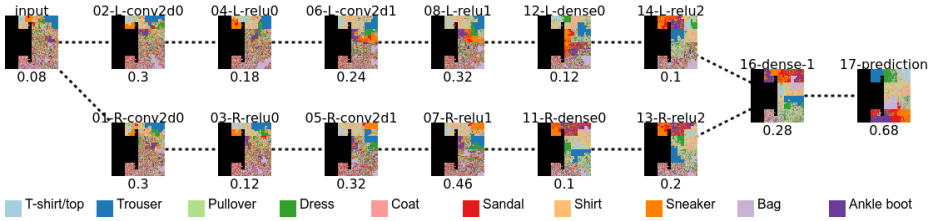
(a) Easy scenario: *LeNet5*, *Mnist* (accuracy=98.96%)



(b) Simplification scenario: *VGG16* + *Mnist* (accuracy=97%)



(c) Generalization scenario: *LeNet5* + *FashionMnist* (accuracy=73.44.5%)



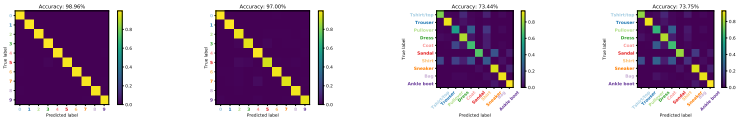
(d) Realistic scenario: *DoubleLeNet5* + *FashionMnist* (accuracy=73.75%)

Fig. 3. Illustration of results on some scenarios. The simplified version of the network is drawn. The confusion matrices are presented in 4 for comparison with a standard visual evaluation method. Larger images are available here: `https://pivert.labri.fr/frac/`.

Results. The accuracy for the system of the *easy scenario* is 98.96%. This appears clearly in the latest Hilbert curve where clusters appear neatly. Looking at the successive operations that correspond to activation functions (`01-relu0`, `03-relu1`, `06-relu2`, `09-prediction`), we observe an improvement in the quality of the representation, which means the network improves its discriminability ability over layers. Classes `1` and `0` seem to be highly differentiable since the beginning of the network, as well as a subset of `6`, `7`, `8`. Discriminability improves greatly at layer `05-dense0`.

The accuracy in the *generalization scenario* reduces to 73.44%. This performance reduction is observed with the latest curve that is saltier as well as the representation of the inner operations. `T-shirt` and `Trouser` classes start to be differentiable at the beginning of the network while `Ankle boots` or `Sandal` are differentiable only at the end of the network.

The *problematic scenario* provides noisy representations associated to the worst quality measures as the network is unable to classify properly. Cluster tendencies may appear because samples of the same class can be somehow



(a) *LeNet5*, *Mnist* (b) *VGG16* (base), *Mnist* (c) *LeNet5*, *FashionMnist* (d) *DoubleLeNet5*, *FashionMnist*

Fig. 4. Confusion matrices of the systems presented in Fig. 3.

transformed similarly. The *simplification scenario* has the accuracy of 97% (so lower than *LeNet5*). As a consequence, the cluster frontiers of the latest Hilbert curve is less neat. There is a succession of improvement/decreasing of drawing quality over time. We assume they come from several reasons: they are pre-trained and not specialized for the task, and they are too deep and redundant.

The *realistic scenario* illustrates the ability to draw networks with branches. We observe the same tendencies in both branches with an initial increase in the drawing quality then a decrease. They are also confident on the same classes.

Discussion. We observe easily the efficient use of the screen space (remember each sample is drawn once per layer) in comparison to a T-SNE projection [28]. Pixels usage is maximized; but not all points of the curves are used; black pixels correspond to unused pixels because test datasets are smaller than what is technically possible with such display size. It is possible to evaluate larger datasets without using more space on the screen. Another observation is the effectiveness of samples projection over the fractal curve to depict the classification performance over layers. Usually, dataset representation is less and less salty over layers which means the network is better and better at separating classes of samples. *FashionMnist* which is a problem more difficult than *Mnist* is also saltier than the later one: we are able to visually represent this fact.

By construction, the very first node corresponds to the projection of the raw dataset; the noisier it is, the more complex it is to distinguish its samples without extracting additional features. The representation clearly depicts this point and its quality metric is worst for *FashionMnist* than *Mnist*. The very last node corresponds to the projection of the softmax values; the noisier it is, the worst the accuracy is. The final representation is complementary of a confusion matrix (see Fig. 4) as it provides more information.

A labeled dataset is currently needed to color the pixels. It limits the use of the method to a test dataset and not a real world unlabeld dataset. However, it is still possible to use the predicted labels instead of the groundtruth ones to obtain a view of how the network interprets the data.

Fig 5 compares the absolute and relative color schemes for one operation able to differentiate the samples and another one yet not able to differentiate them. Thanks to the color, the absolute one allows to clearly see which classes is subject to more noise than the others, while the relative one allows to better see

(a) Good system, absolute coloring (b) Good system, relative coloring (c) Bad system, absolute coloring (d) Bad system, relative coloring

Fig. 5. Comparison of the absolute and relative color schemes. No data is depicted in black for absolute and light blue for relative color schemes.

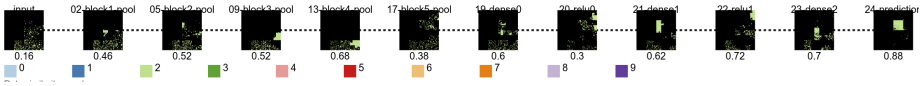


Fig. 6. Analysis of the spread of samples of class 2 over layers. Such representation could indicated an oversizing of the network by looking at the separation effect around layer 17.

the relative quantity of errors. The relative one is also able by construction to handle many classes.

Focusing on a specific class helps to track the evolution of the treatment of samples for that chosen class over layers. Fig. 6 illustrates a possible oversizing of the network by observing that samples of selected class tend to be considered similar around layer 9, whereas it is not the case anymore around layer 17.

Future Work. The method is resource consuming, mainly due to the need of storing dissimilarity matrices in memory. As a future work, it would be interesting to study whether one could use smaller part, or estimation, of the dissimilarities to approach a similar visualization. Additionally, the ordering of the samples highly depend on the Euclidean distance that is known to not be efficient in high dimensional spaces. Other metrics need to be compared.

The approach is satisfactory using interaction, but is not yet self-sufficient. Indeed, it provides a good overview of how the classification is handled but lacks of interactions to track the progression of a single sample or group of samples (in opposite to our previous work that specifically focus on this point [11]) in the network. Such investigations have to be held.

The Hilbert curve is very efficient to place the samples in its reserved space. However, there is a high probability that the number of elements in the dataset to visualize is lower than what is possible with the curve (in our experiment, $4^7 - 10\,000$ pixels are lost, which is roughly 39% of the picture for each layer). It would be interesting to implement additional interactions that use this additional space; or use grid-based projection methods instead of fractal ones. To subsample or sample with replacement the dataset with a number of samples equals to the curve length, and that follows data distribution, could also be interesting. The standard Sugiyama algorithm does not consider the screen space size; a modified

method should be used in order to project the graph on the screen in a way that does not necessitate to horizontally scroll the screen to see it [22].

6 Conclusion

Deep learning classifiers are progressively replacing handcrafted and understood standard classifiers for various fields. This performance gain is counterbalanced by a difficulty in understanding how and why they perform well. Information visualization is one solution to this lack of interpretability. We have presented a pipeline consuming a trained network and a dataset and producing an interactive representation depicting both the network's architecture and the behaviors of the test samples at each layer. Such system allows to visually analyze the classification quality over layers of a dataset and could be used to visually detect patterns in the data and propose a hypothesis about the performance of the network. Such hypothesis would need then to be verified by other means.

This approach has been validated on various scenarios and shows its interest and limits that will be overcome in the future. It will be extended with various specific interaction methods to also focus on individual data and, subsampling or dense pixel based method that do not "lost" screen space.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., *et al.*: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `https://www.tensorflow.org/`, software available from tensorflow.org
2. Asif Fuad, K.A., Martin, Pierre-Etienne andGiot, R., Bourqui, R., Benois-Pineau, J., Zemmari, A.: Features understanding in 3d cnns for actions recognition in video. In: The tenth International Conference on Image Processing Theory, Tools and Applications (IPTA 2020). p. 6 (2020)
3. Auber, D., Novelli, N., Melançon, G.: Visually mining the datacube using a pixel-oriented technique. In: 2007 11th International Conference Information Visualization (IV'07). pp. 3–10. IEEE (2007)
4. Behrisch, M., Bach, B., Henry Riche, N., Schreck, T., Fekete, J.D.: Matrix reordering methods for table and network visualization. Computer Graphics Forum **35**(3), 693–716 (2016)
5. Binder, A., Montavon, G., Lapuschkin, S., Müller, K.R., Samek, W.: Layer-wise relevance propagation for neural networks with local renormalization layers. Lecture Notes in Computer Science, vol. 9887, pp. 63–71. Springer Berlin / Heidelberg (2016)
6. Blanchard, F., Herbin, M., Lucas, L.: A new pixel-oriented visualization technique through color image. Information Visualization **4**(4), 257–265 (2005)
7. Chollet, F., et al.: Keras (2015), `https://keras.io`
8. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. Tech. rep., University of Montreal (2009)
9. Gardner, M.: Mathematical games–in which "monster" curves force redefinition of the word "curve". Scientific American **235**(6), 124–133 (1976)

10. Ghazi, M.M., Yanikoglu, B., Aptoula, E.: Plant identification using deep neural networks via optimization of transfer learning parameters. Neurocomputing **235**, 228–235 (2017)
11. Halnaut, A., Giot, R., Bourqui, R., Auber, D.: Deep dive into deep neural networks with flows. In: 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. pp. 231–239 (2020)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
13. Hilbert, D.: Über die stetige abbildung einer linie auf ein flächenstück. In: Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes, pp. 1–2. Springer (1935)
14. Hohman, F., 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 **25**(8), 2674–2693 (2018)
15. Kahng, M., Andrews, P.Y., Kalro, A., Chau, D.H.: Activis: Visual exploration of industry-scale deep neural network models. IEEE Transactions on Visualization and Computer Graphics **24**(1), 88–97 (2018)
16. Kahng, M., Thorat, N., Chau, D.H.P., Viégas, F.B., Wattenberg, M.: Gan lab: Understanding complex deep generative models using interactive visual experimentation. IEEE transactions on visualization and computer graphics **25**(1), 1–11 (2018)
17. Kahng, M., Thorat, N., Chau, D.H.P., Viégas, F.B., Wattenberg, M.: Gan lab: Understanding complex deep generative models using interactive visual experimentation. IEEE transactions on visualization and computer graphics **25**(1), 1–11 (2018)
18. Keim, D.A.: Designing pixel-oriented visualization techniques: Theory and applications. IEEE Transactions on visualization and computer graphics **6**(1), 59–78 (2000)
19. LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
20. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
21. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
22. Liu, M., Liu, S., Su, H., Cao, K., Zhu, J.: Analyzing the noise robustness of deep neural networks. In: 2018 IEEE Conference on Visual Analytics Science and Technology (VAST). pp. 60–71. IEEE (2018)
23. Liu, M., Shi, J., Li, Z., Li, C., Zhu, J., Liu, S.: Towards better analysis of deep convolutional neural networks. IEEE transactions on visualization and computer graphics **23**(1), 91–100 (2017)
24. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**, 2579–2605 (2008)
25. Meyer, M., Sedlmair, M., Quinan, P.S., Munzner, T.: The nested blocks and guidelines model. Information Visualization **14**(3), 234–249 (2015)
26. Morton, G.M.: A computer oriented geodetic data base and a new technique in file sequencing. Tech. rep., International Business Machines Company New York (1966)
27. Norton, A.P., Qi, Y.: Adversarial-playground: A visualization suite showing how adversarial examples fool deep learning. In: Visualization for Cyber Security (VizSec), 2017 IEEE Symposium on. pp. 1–4. IEEE (2017)

28. 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 (2017)
29. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
30. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: 2017 IEEE International Conference on Computer Vision (ICCV). p. 618–626 (2017)
31. d. Silva, L.E.B., Wunsch, D.C.: A study on exploiting vat to mitigate ordering effects in fuzzy art. In: 2018 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2018)
32. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. preprint arXiv:1409.1556 (2014)
33. Smilkov, D., Carter, S., Sculley, D., Viégas, F.B., Wattenberg, M.: Direct-manipulation visualization of deep networks. arXiv preprint arXiv:1708.03788 (2017)
34. Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. IEEE Transactions on Systems, Man, and Cybernetics **11**, 109–125 (1981)
35. Tang, Y.: Deep learning using linear support vector machines. In: In ICML (2013)
36. Wang, J., Gou, L., Shen, H.W., Yang, H.: Dqnviz: A visual analytics approach to understand deep q-networks. IEEE transactions on visualization and computer graphics **25**(1), 288–298 (2019)
37. Wang, Z.J., Turko, R., Shaikh, O., Park, H., Das, N., Hohman, F., Kahng, M., Chau, D.H.: Cnn 101: Interactive visual learning for convolutional neural networks. In: Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems. pp. 1–7 (2020)
38. Wongsuphasawat, K., Smilkov, D., Wexler, J., Wilson, J., Mane, D., Fritz, D., Krishnan, D., Viégas, F.B., Wattenberg, M.: Visualizing dataflow graphs of deep learning models in tensorflow. IEEE transactions on visualization and computer graphics **24**(1), 1–12 (2017)
39. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
40. Zhang, Q., Nian Wu, Y., Zhu, S.C.: Interpretable convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8827–8836 (2018)
41. Zhang, Q., Yang, Y., Ma, H., Wu, Y.N.: Interpreting cnns via decision trees. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6261–6270 (2019)