Structuring Neural Networks for More Explainable Predictions*

LAURA RIEGER

Technical University of Denmark, DTU Compute, Kgs. Lyngby, Denmark

PATTARAWAT CHORMAI

Department of Electrical Engineering & Computer Science, Technische Universität Berlin,
Marchstr. 23, Berlin 10587, Germany

GRÉGOIRE MONTAVON

Department of Electrical Engineering & Computer Science, Technische Universität Berlin,
Marchstr. 23, Berlin 10587, Germany

KLAUS-ROBERT MÜLLER

Department of Electrical Engineering & Computer Science, Technische Universität Berlin,
Marchstr. 23, Berlin 10587, Germany
Department of Brain & Cognitive Engineering, Korea University,Anam-dong 5ga, Seongbuk-gu, Seoul 136-713, South Korea
Max Planck Institute for Informatics, Stuhlsatzenhausweg, Saarbrücken 66123, Germany

LARS KAI HANSEN

Technical University of Denmark, DTU Compute, Kgs. Lyngby, Denmark

Machine learning algorithms such as neural networks are more useful, when their predictions can be explained, e.g. in terms of input variables. Often simpler models are more interpretable than more complex models with higher performance. In practice, one can choose a readily interpretable (possibly less predictive) model. Another solution is to directly explain the original, highly predictive model. In this chapter, we present a middle-ground approach where the original neural network architecture is modified parsimoniously in order to reduce common biases observed in the explanations. Our approach leads to explanations that better separate classes in feed-forward networks, and that also better identify relevant time steps in recurrent neural networks.

I. INTRODUCTION

Neural networks are powerful learning machines that derive their power from the interconnection of a large number of elementary computational units (neurons). A significant body of work has focused on finding

*The final authenticated publication is available online at https://doi.org/10.1007/978-3-319-98131-4_5

appropriate neural network structures for specific problems [18, 13, 9]. For example, on image classification tasks, convolution-type architectures have proven to be highly efficient [16, 36]. Similar models are also being used increasingly in fields such as computational biology [1] or physics [29].

Motivated by these successes, there is a renewed interest in developing techniques to interpret how these highly predictive neural network models reach their decisions. Some explanation techniques choose the architecture in a way that it becomes interpretable, by defining the function as a simple sum over readily interpretable quantities [25, 8, 40]. Other methods seek to explain a more general set of deep neural network architectures [31, 38, 3, 27]. Three methods, sensitivity analysis [10, 4, 31], guided backprop [34] and deep Taylor decomposition [21], all of them applicable to sequences of linear and ReLU layers, will be considered in this paper.

The paper asks the question whether high prediction accuracy is a sufficient condition for high explanation quality, and what additional steps are then necessary to also reach high explainability. The role of regularization and the interplay between performance and robustness of global sensitivity maps has been investigated, e.g. in Rasmussen et al [26]. Here we focus on interpretability of the individual decisions. More precisely, we will test whether explanations exhibit a systematic bias, i.e. a constant divergence between the features identified by the explanation technique and the actual features used by the model to predict, and how the structure of the neural network can be adapted to reduce such bias. Section II introduces the explanation techniques considered in this paper. In Sections III and IV, we present examples of highly predictive models for which explanations are difficult to extract, and how simple and parsimonious structural modifications of the neural network allow to maintain high predictive accuracy, while improving the explanations.

II. Explanation techniques

This section reviews a set of techniques for explaining the decisions made by neural networks. It focuses on sequences of linear and ReLU layers. Highly predictive convolutional neural networks (CNNs) or recurrent neural networks (RNNs) can be built from these sequences of layers. Let $\vec{x} = (x_1, \ldots, x_d)$ be the d-dimensional input presented to the neural network, and $f(\vec{x})$ the value of some output neuron. We focus on explanation methods, that aim to score input relevance according to additive contributions to the function output. An explanation is defined as a vector of scores $(R_1, \ldots, R_d)$ identifying the contribution of each input variable to the function value $f(\vec{x})$.

i. Sensitivity Analysis

A common way of defining these scores is based on the locally evaluated gradient $\nabla_{\vec{x}} f(\vec{x})$. The gradient can be efficiently computed with the backpropagation algorithm. Consider a deep network composed of multiple layers, where each layer is composed of a linear transformation followed by an element-wise ReLU nonlinearity. Letting j and k index neurons of two consecutive layers, activations $(a_j)_j$ and $(a_k)_k$ in the respective layers can be related as $a_k = \max(0, z_k)$, where $z_k = \sum_j a_j w_{jk} + b_k$ is called the pre-activation.

The backpropagation algorithm transmits partial derivatives from the top of the network to the input by repeated application of the chain rule. Let δ_j and δ_k be a shortcut notation for the locally evaluated partial derivatives $\partial f / \partial z_j$ and $\partial f / \partial z_k$. In this network, the chain-rule equation for propagating derivatives is

$$\delta_j = 1_{z_j > 0} \cdot \sum_k w_{jk} \delta_k \tag{1}$$

in the hidden layers, and $\delta_i = \sum_j w_{ij} \delta_j$ for the first layer. The gradients are propagated until the input variables, where they can be converted to importance scores, e.g. by squaring ($R_i = \delta_i^2$). We refer to this way of setting importance scores as *sensitivity analysis* (SA). Explanation through sensitivity analysis has been used, e.g. by Gevrey et al [10], Baehrens et al [4], and Simonyan et al [31]. Sensitivity analysis as well as

other methods relying on the gradient assume that the function value is not varying too quickly in the input domain. This assumption usually does not hold for deep networks, where the function becomes steeper and higher-frequency with every added layer, leading to an uninformative gradient [5].

To remediate to this problem, alternate propagation rules can be applied, for example the *guided backprop* (GB) technique [34] applies the modified rule

$$\tilde{\delta}_j = 1_{z_j>0} \cdot \max\left(0, \sum_k w_{jk} \tilde{\delta}_k\right) \tag{2}$$

which rectifies the incoming gradient and therefore prevents inhibitory effects to propagate. The propagated signal is no longer a gradient, but still retains a rough interpretation as a local direction of variation. Like for the gradient, the result of the propagation procedure can be converted to importance scores by squaring, i.e. $R_i = \tilde{\delta}_i^2$.

In general, methods relying solely on the gradient or similar quantities are in essence closer to an explanation of the function's variation than of the function value itself: For example, sensitivity scores relate to the function as: $\sum_i R_i = \sum_i \delta_i^2 = \|\nabla_{\vec{x}} f(\vec{x})\|^2$, i.e. the scores are a decomposition of the function's local slope [22]. Stated otherwise, these methods explain why the function varies strongly locally, however, they do not explain why the function has high value locally.

ii. Deep Taylor Decomposition

To explain the function's value we aim for an importance score that directly relates to $f(\vec{x})$. A number of works have proposed to attribute importance scores subject to the conservation constraint $\sum_i R_i = f(\vec{x})$, and where these scores are computed using a specific graph propagation procedure [17, 3, 39, 21, 30]. Unlike gradient-based methods, the quantity propagated at each neuron is no longer the partial derivatives δ_j, δ_k or some variant of it, but importance scores R_j, R_k. In the following, we present the *deep Taylor decomposition* (DTD) approach [21] to explaining $f(\vec{x})$, for which rules specific to deep networks with ReLU nonlinearities were derived. The DTD propagation rule between two hidden layers is given by

$$R_j = \sum_k \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k, \tag{3}$$

where $w_{jk}^+ = \max(0, w_{jk})$. The intuition for this rule is to redistribute the function value based on the excitation incurred by neurons in the lower-layer onto neurons of the current layer. This rule also has an interpretation as a Taylor decomposition of relevance R_k in the space of positive activations $(a_j)_j \in \mathbb{R}_+$. Another DTD rule specific to the input layer receiving as input pixel intensities $x_i \in [l_i, h_i]$ is given by

$$R_i = \sum_j \frac{x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_i x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-} R_j, \tag{4}$$

with $w_{ij}^+ = \max(0, w_{ij})$ and $w_{ij}^- = \min(0, w_{ij})$. A strict application of the DTD method imposes as additional requirements the absence of positive bias parameters and the ability to represent the concept to explain as a top-layer ReLU neuron. More details and theoretical justification for the DTD propagation rules are given in the original paper [21].

iii. Theoretical Limitations

The methods presented above, namely sensitivity analysis, guided backprop, and deep Taylor decomposition, are in principle applicable to a broad range of architectures, including shallow or deep ones, fully or locally connected, as well as recurrent architectures.

However, despite this broad applicability, the quality of explanation can differ strongly depending on the structure of the neural network. Unlike predictions made by the deep network, explanations are not what the network is trained for and come as a by-product instead. The fact that the model is not optimized for explanation error implies a possibly strong divergence from a ground truth explanation. We identify two potential sources of divergence:

The first source of divergence is gradient noise, and affects SA: Although a function $f(\vec{x})$ may be close to the ground truth $f^\star(\vec{x})$ in terms of function value (i.e. $\forall_{\vec{x}} : \|f(\vec{x}) - f^\star(\vec{x})\| \leq \varepsilon$), the gradient of the function, on which sensitivity analysis is based, can still be made uncontrollably large [33, 32, 23, 5]. As a consequence, the resulting explanations are no longer selective of the target concept to explain. A corollary of this gradient noise in the context of RNN architecture is the exploding gradients problem [7, 24], where a finite variation in the output space can be accompanied by a very large gradient in the space representing the older time steps.

The second source of divergence arises from attempts by explanation techniques such as GB or DTD to reduce gradient noise: For example, the gradient rectification applied by GB makes the procedure more stable than the actual gradient, however, the rectification operation can bias the explanation towards certain types of features in a CNN or certain time steps in a RNN. DTD also strongly departs from the actual gradient by redistributing only based on positive weights and activations in the hidden layers.

In the next two sections, we characterize these sources of divergence in the context of CNNs and RNNs, and propose to modify the neural network architecture specifically for reducing them.

III. Convolutional Neural Networks

Convolutional neural networks (CNNs) are a special category of neural networks that have come to attention in the last years due to their great success in tasks such as image classification [16, 36]. The first layers extract simple features at various locations and build some translation invariance, and the last layers map these features to the final concepts (e.g. image categories). The explanation problem can here be defined as finding which pixels are responsible for a certain classification decision produced at the output of the network.

While evidence for some classes originates from the same pixels (e.g. these classes share some of the low-level features), other semantically less related classes correspond to distinct features in the image, and we would like the explanation to better capture these features.

As an example, an image of *trousers* from the FashionMNIST dataset in Fig. 4 has the flared outline of a dress but otherwise resembles *trousers*. We would expect a heatmap for *trousers* to focus more on the gap between the legs and a heatmap for *dress* to focus more on the flared outline.

An explanation method must therefore be able to identify pixels that are truly relevant for a specific class of interest, and that are not simply relevant in general.

Our hypothesis is that all classes share a common salient component of representation, and that discrimination between classes does not occur as the effect of building individual class-specific features but rather as measuring small differences on this salient component. While this strategy is perfectly viable for the purpose of prediction, any explanation technique that deviates too much from the function itself and that relies instead on the graph structure might be biased with respect to this salient component.

In the following, we analyze the quality of explanations with respect to the structure of CNNs, specifically the level of connectivity of the dense layers, which controls how fast the backpropagated signal mixes between classes. Specifically, we want to build a structure that encourages the use of separate features for different classes.

We consider three different levels of connectedness, depicted in Fig. 1:

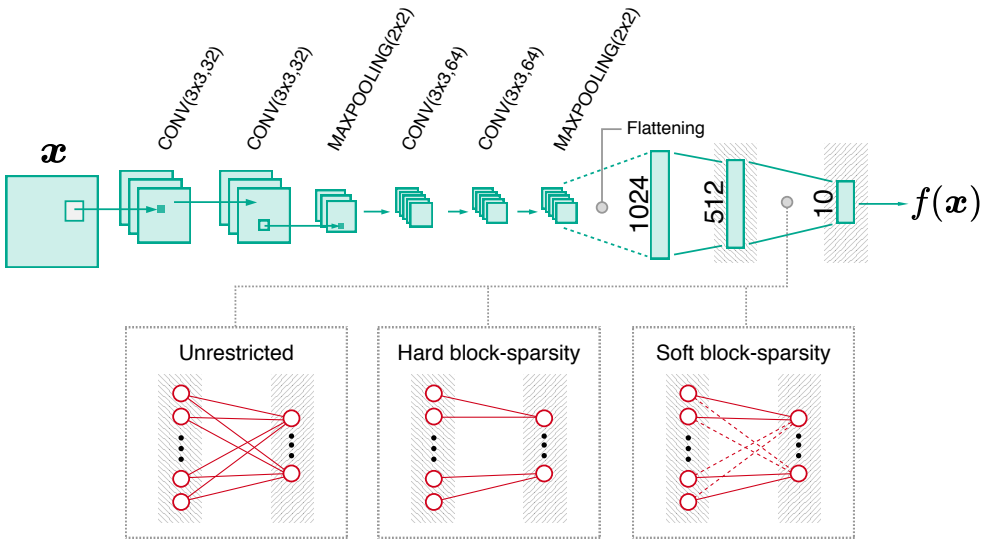

Figure 1: *CNN with various levels of connectedness for the dense layers.*

Structure 1: Unrestricted No restriction is applied to the dense layers of the neural network. That is, if f is the function implemented by the neural network, we simply solve

$$\min_{w,b} \; J_{\text{emp}}(f)$$

that is the standard neural network objective, by minimizing the cost $J_{\text{emp}}(f)$ over the network weights w and biases b. This is our baseline scenario.

Structure 2: Hard block-sparsity Here, we force the weight matrix of the dense layers to have block-diagonal structure so that the classes only recoup near the convolutions layers. That is, we solve the constrained optimization problem

$$\min_{w,b} \; J_{\text{emp}}(f) : \forall_l \forall_{i,j} : w_{ij}^{(l)} = 0 \text{ if } C(i) \neq C(j),$$

where $w_{ij}^{(l)}$ is the weight connecting the ith neuron in layer $l-1$ to the jth neuron in layer l, $\forall_l$ spans the last few dense layers, $\forall_{i,j}$ spans the input and output neurons of the current layer, and $C(i)$ and $C(j)$ are the classes for which neuron i and j are reserved respectively. Practically, the constraint can be enforced at each iteration by multiplying the weight matrix by a mask with block-diagonal structure, or can instead be implemented by splitting the neural network near the output into several pathways, each of which predicts a different class.

Structure 3: Soft block-sparsity In this last setting, the connectivity constraint is replaced by a L1-penalty on all weights that are outside the block-diagonal structure. The optimization problem is rewritten as

$$\min_{w,b} \; J_{\text{emp}}(f) + \lambda \sum_{l,i,j} |w_{ij}^{(l)}| \cdot 1_{C(i) \neq C(j)}, \tag{5}$$

with the same definitions as in Structure 2 and additionally λ controlling the level of sparsity. For DTD, because negative weights are not used in the backward propagation, we can further soften the regularization

constraint to only penalize positive weights, i.e. we replace $|w_{ij}^{(l)}|$ by $\max(0, w_{ij}^{(l)})$ in the equation above. We call these two variants L1 and L1+.

Experiments

We trained several CNNs on the MNIST, FashionMNIST, and CIFAR10 datasets [19, 37, 15]. The neural network used for CIFAR10 is shown in Fig. 1, and the neural networks used for the two other datasets have similar structure. The networks were pre-trained without regularization until the loss no longer improved for eight concurrent epochs, a heuristically chosen number. Due to the restriction of DTD, we constrained biases in all layers to be zero or negative. The trained network is our baseline. This network is fine-tuned by respectively applying L1 regularization, L1+ regularization or a block constraint and training until loss has again no longer improved for eight epochs. We heuristically chose $\lambda = 1.0$ for the regularization rate. The weight parameters of the last layer, to which the structuring penalty is applied, is visualized in Fig. 2.

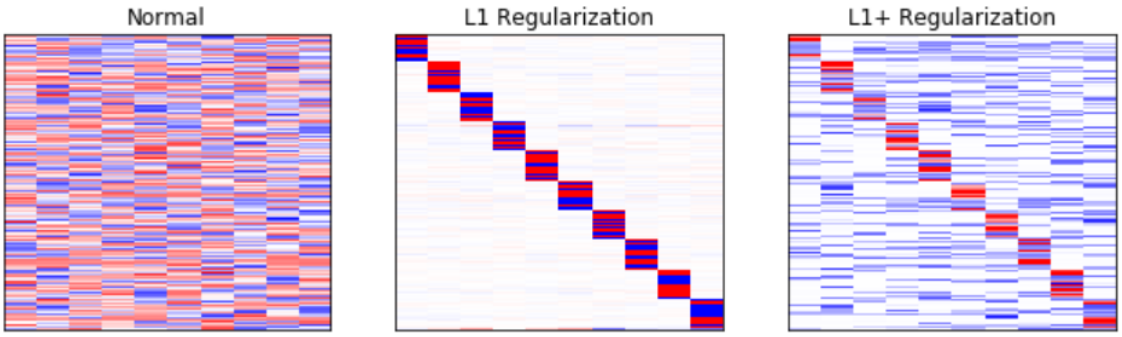

Figure 2: *Visualization of dense layer weights for baseline, L1, and L1+ regularized networks. Positive values are red, negative values are blue.*

Denoting by $R_A(\vec{x})$ and $R_B(\vec{x})$ the heatmaps for the true class and the class with the second highest output, we measure the effectiveness of the architecture at separating classes by the expected cosine distance (ECD):

$$\text{ECD} = \mathbb{E}_{\mathcal{D}}\left[1 - \frac{\langle R_A(\vec{x}), R_B(\vec{x})\rangle}{||R_A(\vec{x})||_2 \cdot ||R_B(\vec{x})||_2}\right], \tag{6}$$

where $\mathbb{E}_{\mathcal{D}}$ is the expectation over the set of test data points for which the neural networks build evidence for at least two classes. A high ECD reflects a strong ability of the neural network to produce class-specific heatmaps, and accordingly suffer less from the explanation bias.[1]

In Fig. 3, the ECD for regularized and normal networks is shown. We see that structuring the network with L1 regularization consistently helps with the disentanglement of class representations for GB and DTD. It does not have a significant effect for SA. This is likely due to the fact that SA is based on local variations of the prediction function and less dependent on the way the function structures itself in the neural network. The effects were consistently present when we repeated the experiments multiple times with different network configurations.

As shown in Table 1, the various structuration schemes do not impact the model accuracy with one exception for the block constraint on CIFAR10. They can therefore be considered as viable methods to decrease entanglement of explanations without trading in performance.

[1] Other quantitative ways of comparing the interpretability of different models, or different explanation techniques, are given in [6, 28].

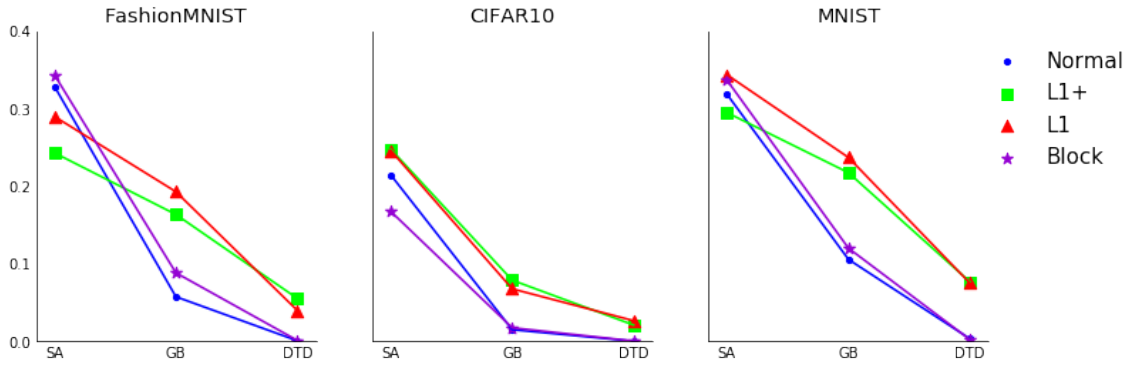

Figure 3: *Explanation separability as measured by the expected cosine distance (ECD), for different models, explanation techniques, and datasets.*

We can see in an example in Fig. 4 that the disentanglement of class representations is reflected in sensible differences between heatmaps. The structured model focuses more on the gap between the legs for *trousers* compared to the heatmap for *dress*. The heatmap for *dress* is spread more uniformly over the entire piece of clothing and focuses on the outline, which resembles a dress with a flared bottom. It is visible that the disentanglement of classes also improves the explanations for the correct class, as they now focus more on the relevant feature.

Table 1: *CNN model accuracy*

Structure	MNIST	FashionMNIST	CIFAR10
Unrestricted	99.16%	91.98%	83.02%
Block	99.29%	91.84%	71.75%
L1	99.20%	92.21%	84.20%
L1+	99.25%	92.55%	84.07%

IV. Recurrent Neural Networks

Recurrent neural networks (RNNs) are a class of machine learning models that can extract patterns of variable length from sequential data. A longstanding problem with RNN architectures has been the modeling of long-term dependencies. The problem is linked with the difficulty of propagating gradient over many time steps. Architectures, such as LSTM [12], or hierarchical RNNs [11], as well as improved optimization techniques [35] have been shown to address these difficulties remarkably well so that these techniques can now be applied to complex tasks such as speech recognition or machine translation. Some work has recently focused on explaining recurrent architectures in the context of text analysis [2].

In a similar way as for Section III, we will hypothesize that the recurrent structure forms a large salient component of representation and that the classes are predicted based on small variations of that component rather on class-specific features. Thus, explanation techniques that deviate from the prediction function itself might be biased towards that salient component.

To verify this, we consider various RNN architectures with different depths and connectivity. Each of these architectures can be expressed in terms of cells receiving the previous state and the current data, and producing the next state and the prediction. We use ReLU activations for every layer and softmax activation

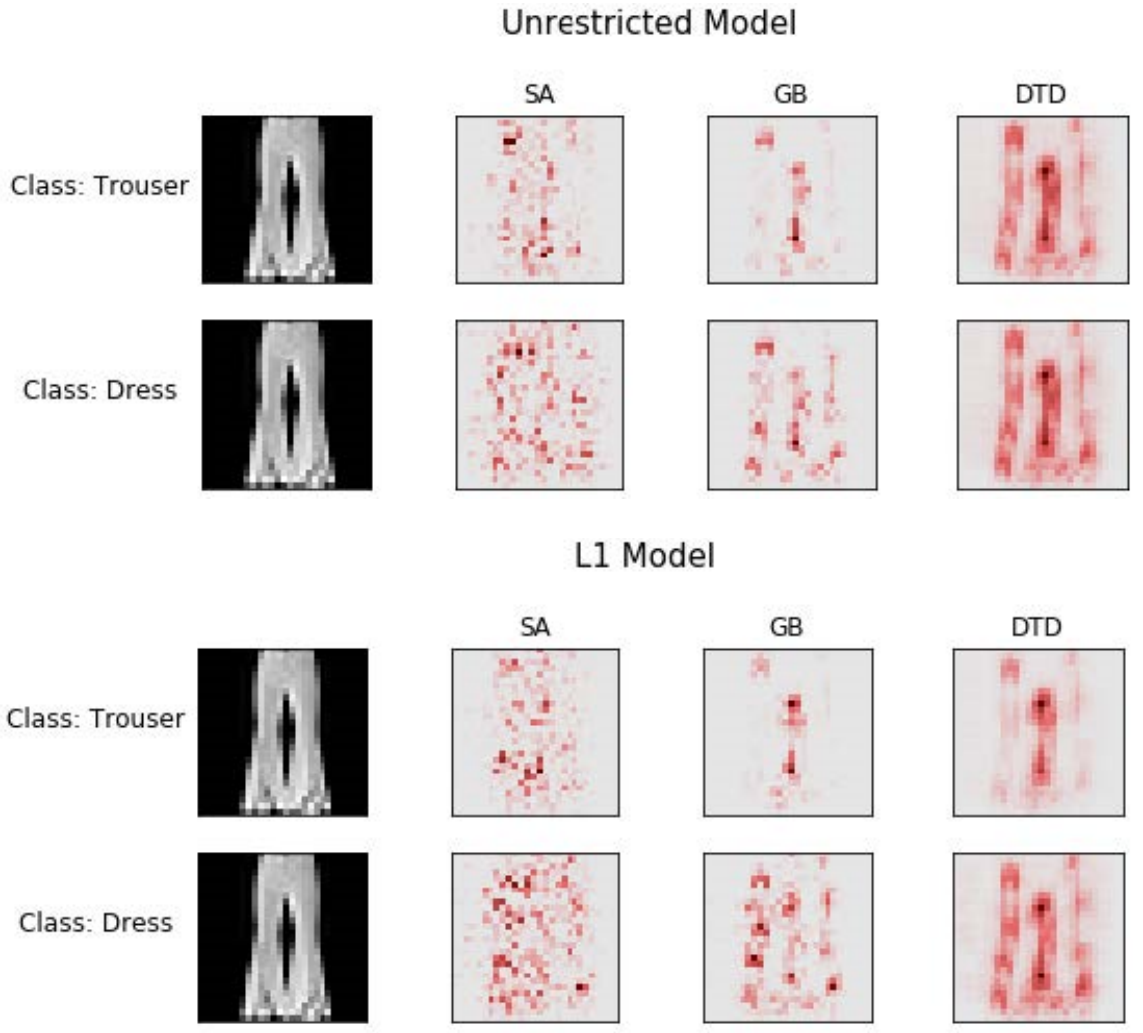

Figure 4: *Heatmaps on FashionMNIST produced by different explanation techniques applied to the basic unrestricted model (top) and the L1/L1+ model with soft block-sparsity (bottom). For the first model, there is nearly no differences between classes. For the second model, the explanations with GB and DTD identify the leg gap as relevant for trouser and flared outline for dress.*

to output the last cell to class probabilities. We consider the following five cell structures (two of them are shown in Fig. 5):

Structure 1: Shallow Cell The shallow cell performs a linear combination of the current state and current data, and computes the next state from it. This is our baseline scenario.

For applicability of deep Taylor decomposition to this architecture, we need an additional propagation rule to redistribute on two different modalities at the same time (hidden state and pixels). Denoting i and j the pixels and ReLU activations respectively forming the two cell modalities and k the hidden layer neuron, the propagation rule is redefined as $R_i = \sum_k (x_i w_{ik} - l_i w_{ik}^+ - h_i w_{ik}^-) \cdot (R_k / z_k)$ and $R_j = \sum_k r_j w_{jk}^+ (R_k / z_k)$, where $z_k = \sum_j r_j w_{jk}^+ + \sum_i x_i w_{ik} - l_i w_{ik}^+ - h_i w_{ik}^-$ is the normalization term.

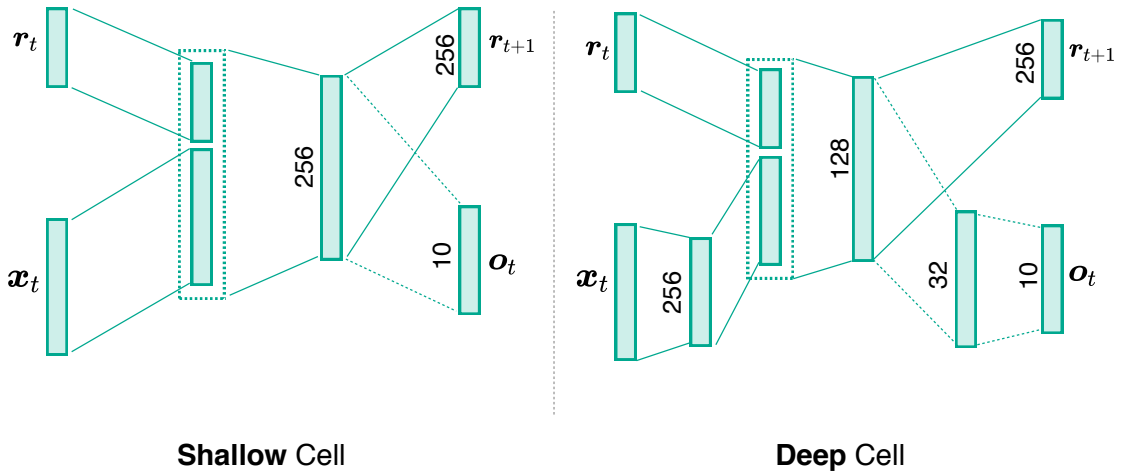

Figure 5: *Examples of RNN Cell Architectures.*

Structure 2: Deep Cell The deep cell nonlinearly combines the current state and the current data. This allows to build a data representation that can more meaningfully combine with the hidden state representation. It also makes explanation easier as the two modalities being merged are ReLU activations, and therefore, do not need a special propagation rule for DTD.

Structure 3: Convolutional-Deep Cell The convolutional-deep (ConvDeep) cell is an extension of the Deep cell in which a sequence of 2 convolution and pooling layers is applied to the input instead of a fully-connected layer. More precisely, we use 24 convolutional filters of size 5×5, followed by *sum* pooling with 2×2 receptive fields. The second convolutional layer has 32 filters of size 3×3, and the setting of the following pooling is the same. We use stride 1 for the two convolution layers, and stride 2 for the pooling layers. This allows to produce well-disentangled features that integrate better with the recurrent states.

Structure 4: R-LSTM Cell This cell is another variant of the Deep cell. It employs one fully-connected layer with 256 neurons connecting to 75 R-LSTM cells. The R-LSTM cell is a modified version of LSTM whose tanh activations are replaced by ReLU in order to satisfy the constraint of GB and DTD. We treat gate activations in the cell as constants when applying DTD as suggested by Arras et al [2], and set their gradients to zero for GB.

Structure 5: ConvR-LSTM Cell The last cell is an extension of R-LSTM where the first fully-connected layer is replaced by the convolution and pooling layers used in Structure 3.

Experiments

We construct an artificial problem consisting of three images concatenated horizontally (two of a given class, and one of another class), and where the goal is to predict the dominating class. We consider MNIST [19] or FashionMNIST [37] examples for this experiment. This leads to classification tasks where the input $\vec{x}$ is a mosaic of size 28×84, and where the output is a set of 10 possible classes. With this construction, we can easily estimate explainability by measuring which percentage of the explanation falls onto the correct tiles of the mosaic.

The problem above is mapped to the RNN architecture by horizontally splitting $\vec{x}$ into non-overlapping segments $\{\vec{x}_t \in \mathbb{R}^{28\times 7}\}_{t=1}^{12}$ and sequentially presenting these segments to the RNN. Fig. 6 illustrates the setting.

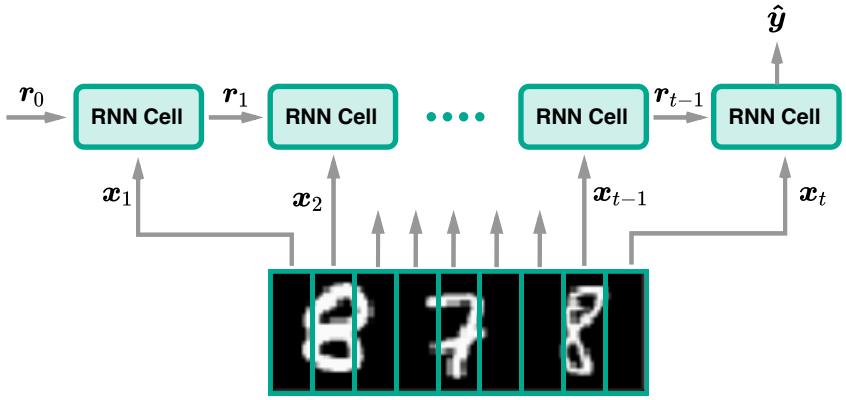

Figure 6: *RNN architecture scanning through a sequence of three digits and predicting the dominating class, here "8".*

Table 2: *Number of parameters of each RNN structure, and model accuracy*

		Accuracy	
Cell Architecture	**# Parameters**	**MNIST**	**FashionMNIST**
Shallow	184330	98.12%	90.00%
Deep	153578	98.16%	89.81%
ConvDeep	151802	99.22%	92.87%
R-LSTM	150570	98.50%	91.35%
ConvR-LSTM	152125	99.26%	93.33%

The number of neurons for each layer in each architecture is chosen such that these architectures have a similar number of training parameters. Table 2 summarizes the numbers. All models are trained using the backpropagation through time procedure and using the Adam optimizer [14]. We initialize weights with 2σ-truncated normal distribution with $\mu = 0$ and $\sigma = 1/\sqrt{|\vec{a}|}$ where $|\vec{a}|$ is the number of neurons from the previous layer as suggested in [20]. Biases are initialized to zero and constrained to be zero or negative during training. We train for 100 epochs using batch size 50. We apply dropout to every fully-connected layers, except neurons in input and output layers. Dropout probability is set to 0.2.

The learning rate is adjusted for each architecture to achieve good predictive performance. To use an architecture for experiments, we require that accuracy reaches approximately 98% and 90% on MNIST and FashionMNIST respectively. Lastly, we add one additional input with constant value zero to the softmax layer. This last modification forces the model to build positive evidence for predicting classes rather than relying on building counter-evidence for other classes.

Fig. 7 shows relevance heatmaps produced by various methods on the Shallow, Deep, ConvDeep, R-LSTM and ConvR-LSTM architectures. We observe that incorporating structure into the cell leads to a better

Figure 7: *Heatmaps obtained with each RNN structure, for different explanation techniques and datasets.*

allocation onto the relevant elements of the sequence. This is particularly noticeable for DTD, where heatmaps of the base model (Shallow) are strongly biased towards a *salient component* constituted of the rightmost pixels, whereas heatmaps for the structured models, especially LSTMs, are more balanced. ConvR-LSTM further improves R-LSTM's heatmaps by providing more resolution at the pixel level. Nevertheless, the presence of features from irrelevant input, such as "1" in Digit 0 example, suggests that cell design can be further improved for the purpose of explanation, beyond the modifications we have proposed here.

In the following, we provide quantitative measures of heatmap quality.[2] By construction, we know that

[2] see also [6, 28]

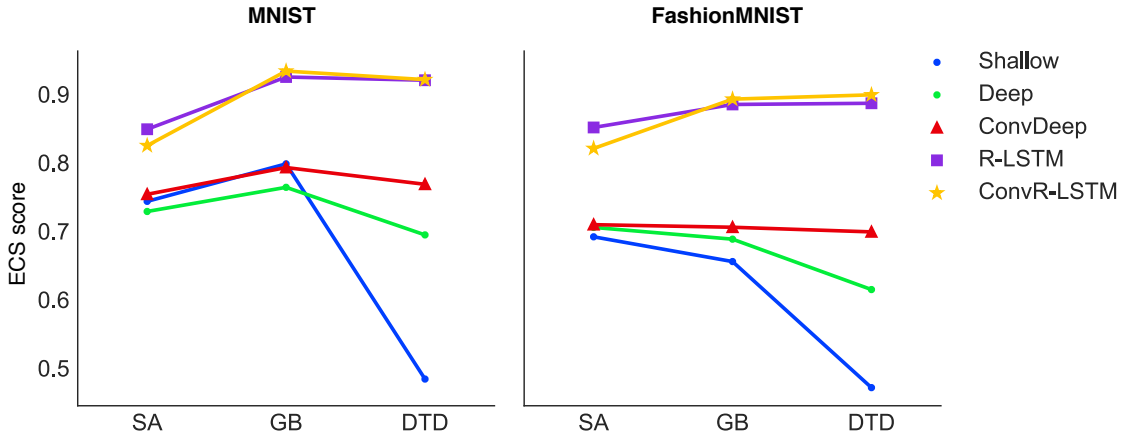

Figure 8: *Explanation quality as measured by the expected cosine similarity (ECS), for different models, explanation techniques, and datasets.*

relevance should be assigned to the two dominating items in the sequence (i.e. those that jointly determine the class). The degree to which heatmaps satisfy this property can be quantified by computing the cosine similarity between a binary vector $I(\vec{x}) \in \{(1,1,0),(1,0,1),(0,1,1)\}$, indicating what are the two items of the sequence $\vec{x}$ having the same class, and a vector of the same dimensions $R(\vec{x}) \in \mathbb{R}^3$ containing relevance scores pooled on each item of the sequence. Our metric for quantifying explanation power is the expected cosine similarity:

$$\text{ECS} = \mathbb{E}_{\mathcal{D}}\left[\frac{\langle I(\vec{x}), R(\vec{x})\rangle}{\|I(\vec{x})\|_2 \cdot \|R(\vec{x})\|_2}\right], \tag{7}$$

where $\mathbb{E}_{\mathcal{D}}[\cdot]$ computes an average over all sequences in the test set. The higher the ECS the better. Fig. 8 shows our ECS metric for various models and explanation techniques. Generally, we can see that more structured cells have higher ECS than the Shallow architecture. In particular, R-LSTM and ConvR-LSTM show significant improvements across all methods. Moreover, the large difference of the cosine similarity between Shallow and Deep architectures also corroborates the strong impact of cell structure on the DTD heatmaps as it was observed in Fig. 7.

V. Conclusion

The success of neural networks at learning functions that accurately predict complex data has fostered the development of techniques that explain how the network decides. While the training objective closely relates to the prediction task, the explanation of these predictions comes as a by-product and little guarantee is offered on their correctness.

In this paper, we have shown that different neural network structures, while offering similar prediction accuracy, can strongly influence the quality of explanations. Both for the baseline CNNs and RNNs, the explanations are biased towards a salient component. This salient component corresponds to general image features for the CNNs or the last time steps for the RNNs.

While the neural network is still able to solve the task based on capturing small variations of that salient component, the explanation technique, which departs from the function to predict, is much more sensitive to it. Therefore, when explanation of the prediction is needed, it is important to pay further attention to the

neural network architecture, in particular, by making sure that each class or concept to explain, builds its own features, and that these features are well-disentangled.

Acknowledgements

This work was supported by the Brain Korea 21 Plus Program through the National Research Foundation of Korea; the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government [No. 2017-0-01779]; the Deutsche Forschungsgemeinschaft (DFG) [grant MU 987/17-1]; the German Ministry for Education and Research as Berlin Big Data Center (BBDC) [01IS14013A]; and the Innovation Foundation Denmark through the DABAI project. This publication only reflects the authors views. Funding agencies are not liable for any use that may be made of the information contained herein.

References

[1] Angermueller C, Pärnamaa T, Parts L, Stegle O (2016) Deep learning for computational biology. Molecular Systems Biology 12(7)

[2] Arras L, Montavon G, Müller K, Samek W (2017) Explaining recurrent neural network predictions in sentiment analysis. In: Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@EMNLP 2017, Copenhagen, Denmark, September 8, 2017, pp 159–168

[3] Bach S, Binder A, Montavon G, Klauschen F, Müller KR, Samek W (2015) On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLOS ONE 10(7):e0130,140

[4] Baehrens D, Schroeter T, Harmeling S, Kawanabe M, Hansen K, Müller K (2010) How to explain individual classification decisions. Journal of Machine Learning Research 11:1803–1831

[5] Balduzzi D, Frean M, Leary L, Lewis JP, Ma KW, McWilliams B (2017) The shattered gradients problem: If resnets are the answer, then what is the question? In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, pp 342–350

[6] Bau D, Zhou B, Khosla A, Oliva A, Torralba A (2017) Network dissection: Quantifying interpretability of deep visual representations. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pp 3319–3327

[7] Bengio Y, Simard PY, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Networks 5(2):157–166

[8] Caruana R, Lou Y, Gehrke J, Koch P, Sturm M, Elhadad N (2015) Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015, pp 1721–1730

[9] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa PP (2011) Natural language processing (almost) from scratch. Journal of Machine Learning Research 12:2493–2537

[10] Gevrey M, Dimopoulos I, Lek S (2003) Review and comparison of methods to study the contribution of variables in artificial neural network models. Ecological Modelling 160(3):249–264

[11] Hihi SE, Bengio Y (1995) Hierarchical recurrent neural networks for long-term dependencies. In: Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, November 27-30, 1995, pp 493–499

[12] Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Computation 9(8):1735–1780

[13] Jarrett K, Kavukcuoglu K, Ranzato M, LeCun Y (2009) What is the best multi-stage architecture for object recognition? In: IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009, pp 2146–2153

[14] Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. CoRR abs/1412.6980

[15] Krizhevsky A (2009) Learning Multiple Layers of Features from Tiny Images. Tech. rep., University of Toronto

[16] Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States., pp 1106–1114

[17] Landecker W, Thomure MD, Bettencourt LMA, Mitchell M, Kenyon GT, Brumby SP (2013) Interpreting individual classifications of hierarchical networks. In: IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, 16-19 April, 2013, pp 32–38

[18] LeCun Y (1989) Generalization and network design strategies. In: Pfeifer R, Schreter Z, Fogelman F, Steels L (eds) Connectionism in perspective, Elsevier

[19] LeCun Y, Cortes C (2010) MNIST handwritten digit database URL `http://yann.lecun.com/exdb/mnist/`

[20] LeCun Y, Bottou L, Orr GB, Müller KR (2012) Efficient backprop. In: Neural networks: Tricks of the trade, Springer, pp 9–50

[21] Montavon G, Lapuschkin S, Binder A, Samek W, Müller K (2017) Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recognition 65:211–222

[22] Montavon G, Samek W, Müller K (2018) Methods for interpreting and understanding deep neural networks. Digital Signal Processing 73:1–15

[23] Montúfar GF, Pascanu R, Cho K, Bengio Y (2014) On the number of linear regions of deep neural networks. In: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pp 2924–2932

[24] Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013, pp 1310–1318

[25] Poulin B, Eisner R, Szafron D, Lu P, Greiner R, Wishart DS, Fyshe A, Pearcy B, Macdonell C, Anvik J (2006) Visual explanation of evidence with additive classifiers. In: Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA, pp 1822–1829

[26] Rasmussen PM, Hansen LK, Madsen KH, Churchill NW, Strother SC (2012) Model sparsity and brain pattern interpretation of classification models in neuroimaging. Pattern Recognition 45(6):2085–2100

[27] Ribeiro MT, Singh S, Guestrin C (2016) "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, San Francisco, CA, USA, August 13-17, 2016, pp 1135–1144

[28] Samek W, Binder A, Montavon G, Lapuschkin S, Müller KR (2017) Evaluating the visualization of what a deep neural network has learned. IEEE transactions on neural networks and learning systems 28(11):2660–2673

[29] Schütt KT, Arbabzadah F, Chmiela S, Müller KR, Tkatchenko A (2017) Quantum-chemical insights from deep tensor neural networks. Nature Communications 8:13,890

[30] Shrikumar A, Greenside P, Kundaje A (2017) Learning important features through propagating activation differences. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, pp 3145–3153

[31] Simonyan K, Vedaldi A, Zisserman A (2013) Deep inside convolutional networks: Visualising image classification models and saliency maps. CoRR abs/1312.6034

[32] Snyder JC, Rupp M, Hansen K, Müller KR, Burke K (2012) Finding density functionals with machine learning. Physical Review Letters 108(25)

[33] Snyder JC, Rupp M, Müller KR, Burke K (2015) Nonlinear gradient denoising: Finding accurate extrema from inaccurate functional derivatives. International Journal of Quantum Chemistry 115(16):1102–1114

[34] Springenberg JT, Dosovitskiy A, Brox T, Riedmiller MA (2014) Striving for simplicity: The all convolutional net. CoRR abs/1412.6806

[35] Sutskever I, Martens J, Dahl GE, Hinton GE (2013) On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013, pp 1139–1147

[36] Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pp 1–9

[37] Xiao H, Rasul K, Vollgraf R (2017) Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. CoRR abs/1708.07747

[38] Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I, pp 818–833

[39] Zhang J, Lin ZL, Brandt J, Shen X, Sclaroff S (2016) Top-down neural attention by excitation backprop. In: Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV, pp 543–559

[40] Zhou B, Khosla A, Lapedriza À, Oliva A, Torralba A (2016) Learning deep features for discriminative localization. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pp 2921–2929