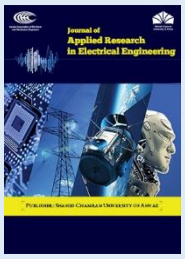
Shahid Chamran University of Ahvaz

Iranian Association of Electrical and Electronics Engineers

Journal of Applied Research in Electrical Engineering

E-ISSN: 2783-2864
P-ISSN: 2717-414X
Homepage: https://jaree.scu.ac.ir/



Research Article

Analyzing the Inference Process in Deep Convolutional Neural Networks using Principal Eigenfeatures, Saturation and Logistic Regression Probes

Mats Leon Richter, Leila Malihi*, Anne-Kathrin Patricia Windler, and Ulf Krumnack

Institute of Cognitive Science, University of Osnabrück, Osnabrück, Germany

** Corresponding Author: lemalihi@uni-osnabrueck.de*

Abstract: The predictive performance of a neural network depends on the one hand on the difficulty of a problem, defined by the number of classes and complexity of the visual domain, and on the other hand on the capacity of the model, determined by the number of parameters and its structure. By applying layer saturation and logistic regression probes, we confirm that these factors influence the inference process in an antagonistic manner. This analysis allows the detection of over- and under-parameterization of convolutional neural networks. We show that the observed effects are independent of previously reported pathological patterns, like the "tail pattern". In addition, we study the emergence of saturation patterns during training, showing that saturation patterns emerge early in the optimization process. This allows for quick detection of problems and potentially decreased cycle time during experiments. We also demonstrate that the emergence of tail patterns is independent of the capacity of the networks. Finally, we show that information processing within a tail of unproductive layers is different, depending on the topology of the neural network architecture.

Keywords: Convolutional neural networks, logistic regression probes, saturation, eigenfeatures, tail pattern.

Article history
Received 16 February 2022; Revised 13 June 2022; Accepted 29 June 2022; Published online 9 July 2022.

How to cite this article
M. L. Richter, L. Malihi, A. -K. P. Windler, and U. Krumnack, "Analyzing the inference process in deep convolutional neural networks using principal eigenfeatures, saturation and logistic regression probes," *J. Appl. Res. Electr. Eng.*, vol. 2, no. 1, pp. 1-10, 2023. DOI: 10.22055/jaree.2022.40073.1049

1. INTRODUCTION

The problem of the opaqueness of neural networks is one of the key challenges in deep learning. This has led to a primarily trial-and-error driven mode of development, based on the comparison of abstract metrics that capture model-agnostic concepts like predictive performance, demand in computational resources, and capacity [1-7]. To move towards a more efficient, principle-based design process, a more profound understanding of the model's state is required. This understanding does not have to be necessarily complete in regard to fully understanding the relation of the input and output of the model. Comparative analysis methods based on singular vector canonical correlation analysis (SVCCA) [8] are good examples of such non-holistic approaches. The information extracted from the model using SVCCA is highly aggregated, condensing each layer pair to a single value, but allows for useful insights into the converged model and the training process, by comparing different layers within that model, as well as the states of one layer at different training epochs. Another example close to our approach is the intrinsic dimensionality used for PCA-based pruning by [9, 10]. This method studies the inference process with spectral methods to determine unproductive layers and unnecessary filters. This method, originally only operable for simple sequential architectures, was later expanded on in [11] to ResNet-style architectures. Logistic regression probes [12] and saturation [13] aggregate a single layer to a number, which allows for easy and intuitive analysis, similar to measuring with a thermometer. While logistic regression probes measure the intermediate solution quality very directly by training logistic regressions on the output of a layer, saturation is more task agnostic. Richter et al. [13, 14] have shown that for visual classification tasks, the dimensionality of the subspace of features responsible for data processing varies significantly depending on the input resolution, leading to model and training inefficiencies. While saturation is easy to define, the exact properties of this metric are yet to be discovered. In this paper, we will consider the following questions to gain a better understanding of hidden layer saturation:

- How do model capacity and problem difficulty influence the saturation value?

- Is the organization of the inference process influenced by the capacity of individual layers or the capacity of the entire network?
- How do saturation patterns evolve in the training phase?

We address these questions in a series of experiments. This paper is an extended version of the conference contribution [15], introducing new results and adding some details. It is structured as follows: After introducing the concepts relevant for our work (Section 2), we demonstrate the idea of principal eigenfeatures using an autoencoder (Section 3). We then present further experiments conducted to address the questions raised above (Sections 4, 5, and 6). The paper concludes with a summary of the results (Section 7).

2. Concepts and Related Work

2.1. Logistic Regression Probes

Logistic regression probes, in this work abbreviated as probes, are a tool proposed by Alain and Bengio [12] used as a "thermometer-style"-scalar metric for analyzing the intermediate solution quality during the forward pass. They are obtained by training a simple logistic regression model on the same task as the original neural network, however, using the layer's output values as input data for training. Hence, the probe performance can be considered as a measure of the linear separability of the target classes in the layer's output representations. As the neural network's softmax layer and the logistic regression probe both minimize cross-entropy, both solve effectively the same task. Therefore, we can use the test accuracy of the logistic regression relative to the model's predictive performance to judge the intermediate solution quality. The logistic regression probe performance should increase monotonically from early to later layers of the network, approaching the predictive performance of the model towards the final layers. Such a development implies that all layers contribute qualitatively to the inference process. Logistic regression probe performance is visualized as a curve with individual measuring points (network layers) arranged in the same order as the data flows through the network during a forward pass. An example of this can be seen in Fig. 1, which displays the probe performances measured on the convolutional layers of VGG19 trained at a low resolution on the ImageNette dataset. From the example we can observe how the intermediate solution quality measured by the probes improve from layer to layer until reaching the same level of accuracy as the model.

2.2. Saturation

The saturation s_l of a layer l is a simple scalar metric that was first introduced by Shenk et al. [16] and Richter et al. [13]. It can be computed for any layer in a neural network based on the layer's output values. It measures how many of the available dimensions in the output space Z_l of the layer l are relevant for the inference process:

$$s_l = \frac{\dim E_l^K}{\dim Z_l} \quad (1)$$

Saturation is computed by approximating the ratio of the dimensionality of the relevant eigenspace *dim* E_l^k of layer l and the extrinsic dimensionality of that layer's activation

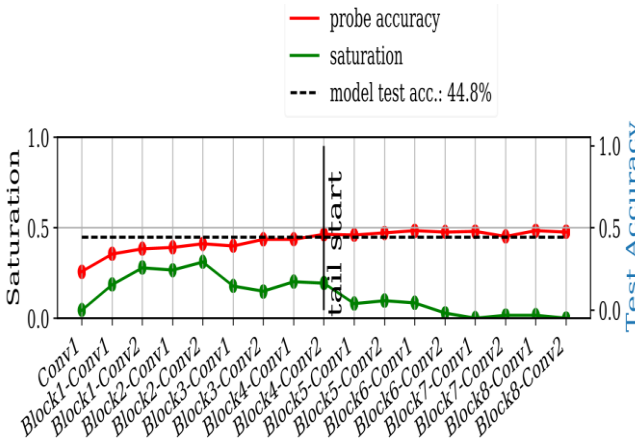


Fig. 1: An example of a tail pattern on a trained ResNet18 model. The tail is starting on the layer with the black border. Tail patterns can be identified by low saturation and stagnation of the logistic regression probe performance. Layers of the tail are no longer improving on the intermediate solution quality. For this reason, these layers can be considered a parameter-inefficiency. The model is trained on ImageNette at 32×32 pixel input resolution.

values *dim* Z_l. The relevant eigenspace E_l^k is a subspace of Z_l in which the information is processed. This space is referred to as "relevant" because a projection of the data into the relevant eigenspace will not lead to a loss of predictive performance [13]. The relevant eigenspace can be considered the subspace in which the information processing is happening. The approximation of E_l^k is done using principal component analysis (PCA) [17], where the largest eigendirections are kept in order to explain 99% of the data's variance in the output of layer l. This technique allows computing saturation on-line during training. In contrast, evaluating logistic regression probes may take significant extra time, as it requires the additional training of the probes from a complete set of activation values, which can easily take more time than training the network. In this work, we use our implementation prepared in the context of the Delve-Framework [18].

2.3. The Semantics of Saturation

A sequence of low saturated layers (< 50% of the average saturation of all other layers) is referred to as a "tail pattern" and indicates that these layers are not contributing qualitatively to the prediction. The example in Fig. 1 displays the saturation values of VGG19 trained on ImageNette alongside the logistic regression probe performances extracted from the same layers. We observe that layers improving the logistic regression probe performances are significantly higher saturated than layers that do not improve the probe accuracy relative to the previous layer.

This suggests that solving a problem saturates the layer more than simply passing through information. However, this does not mean that the absolute saturation value is indicative of the activity within a layer. So far, saturation has been only explored as a quicker on-line computable alternative for logistic regression probes. As such, saturation has always been viewed relative to other layers within a neural network.

In this work, we will explore how the absolute saturation value changes in different scenarios. We further explore how saturation evolves during training, and we will explain the low intrinsic dimensionality observed in the tail pattern and we will explain the low intrinsic dimensionality observed in the tail pattern.

3. Experiment I: Principal Eigenfeatures

The results and experimental work of this paper heavily rely on the analysis of the eigenspace of neural network layers. Since neural networks are feature extractors, we refer to a principal eigendirection inside the feature space of a neural network layer as principal eigenfeature. We first demonstrate the effectiveness of principal eigenfeatures and their relation to the orthogonal feature space using an autoencoder.

3.1. Methods

We choose a convolutional autoencoder since the output is easy to visualize and differences in predictive performance are intuitive to understand with the human eye. The exact architecture of the autoencoder is depicted in Table 1. We train the autoencoder for 30 epochs using the Adam optimizer [19] and a batch size of 128 images on the Food101 dataset [20]; the hyperparameters can be seen in Table 2. We also use random cropping, horizontal flipping, and random rotations for data augmentation purposes, to increase the difficulty of the reconstruction.

Table 1: Convolutional Autoencoder.

Encoder	*Decoder*
512 × 512 × 3 Input	(3 × 3) conv, 8 ReLU
(3 × 3) conv, 16 filters, ReLU	upsampling, nearest, scale-factor 2
(2 × 2) max pooling, strides 2	(3 × 3) conv, 8 filters, ReLU
(3 × 3) conv, 8 filters, ReLU	upsampling, nearest, scale-factor 2
(2 × 2) max pooling, strides 2	(3 × 3) conv, 16 filters, ReLU
(3 × 3) conv, 8 filters, ReLU	upsampling, nearest, scale-factor 2
(2 × 2) max pooling, strides 2	(3 × 3) conv, 3 filters, ReLU

Table 2: Hyperparameters for the convolutional autoencoder.

Parameter	*Parameter*
Input Resolution	*224 × 224*
Epoch	*50*
Batch size	*128*
Optimizer	*Adam*
Adam: beta1	*0.9*
Adam: beta2	*0.999*
Adam: epsilon	*1e-8*
Adam: learning rate	*0.0001*

3.2. Results

In Fig. 2 we visualize a randomly chosen example from the test set. During training time, we evaluate the autoencoder as normal. However, during inference time, we only keep the k largest eigenfeatures that are needed to explain a percentage δ of the data's variance in that layer by using a linear retraction generated from the reduced k-dimensional eigenspace E_l^k using the following formula: $P_{El}{}^k=(E_l^k)^T E_l^k$. By choosing various values for δ, we can observe the ablation caused by the removal of eigenfeatures. As we can see in Fig. 2, the images are recognizable until 99% explained variance. However, it is worth noting that even at 99.99% variance, 4374 of 8192 eigenfeatures were used in the bottleneck of the autoencoder. This is apparent by the dimensionality of the reduced eigenfeature space $E^K{}_{enc}$ of the encoding layer. At 99% variance, the principal eigenfeatures of the bottleneck layer are only 597-dimensional, demonstrating that over-parameterization results in underutilization of the feature space, even in the bottleneck of an autoencoder.

4. Experimnent II: Capacity and Problem Difficulty Behave Proportionality

In this section, we analyse the relationship between problem difficulty and model capacity in two experiments, exploring how this relationship is reflected in the saturation values. In our experiments, we train the entire VGG-network family (VGG11, 13, 16 and, 19) on Cifar10 [21] and reduce their capacity evenly over the entire architecture to observe how these reductions affect the saturation values. Our first hypothesis states that the average saturation s_μ increases proportionally with a reduction in capacity while the model performance decreases. We then move on to investigate how the problem difficulty changes the saturation emerging in a neural architecture. Since the relevant eigenspace is generally larger when the layer is contributing to the quality of the solution [13], we further hypothesize that more processing in a layer requires a larger relevant eigenspace. If this assumption holds true, the overall saturation level should increase with an increase in the difficulty of the task. If both working hypotheses are true, we can conclude that the difficulty of the problem and the capacity of the layers influence saturation in an antagonistic way.

4.1. Methodology

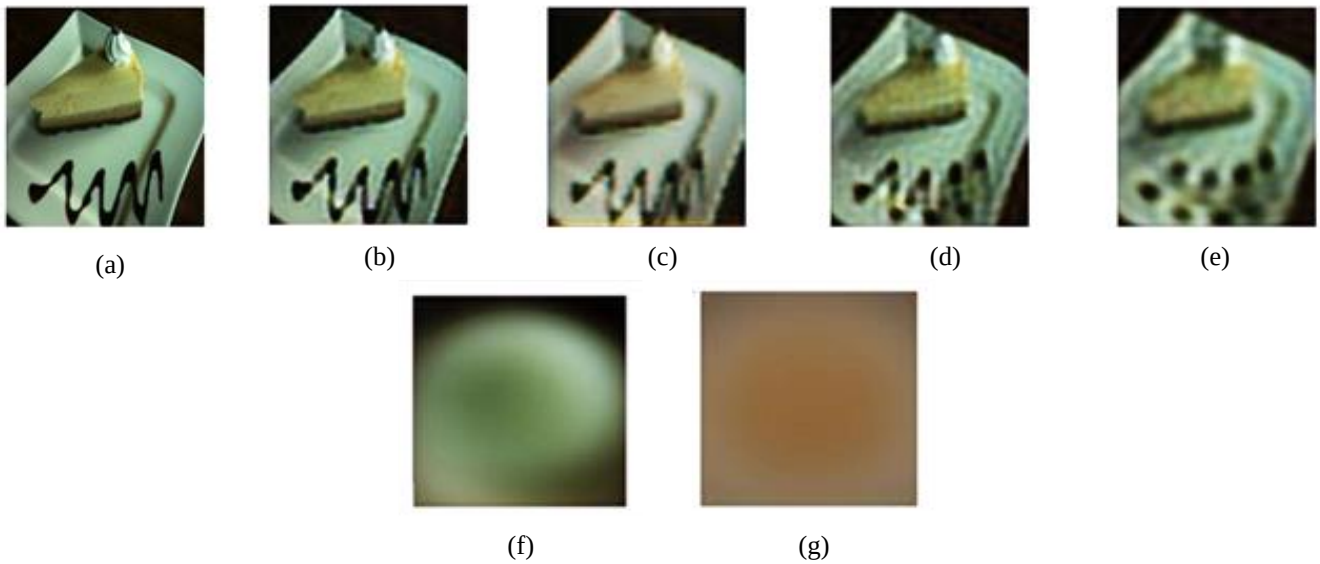
We test our working hypotheses by conducting two experiments. We first train the VGG-family of networks on Cifar10. We further train 4 additional variants of each model, which have the respective number of filters (and thus capacity) reduced by a factor of 12, 14, 18 and 116. We choose Cifar10 for its manageable size, which allows for a larger number of model training runs to be conducted with our available resources, which is necessary for this experiment. We choose the VGG-family of networks for its architectural simplicity and because we can test different depths of convolutional neural networks by experimenting on the entire family of networks. The training itself is conducted using a stochastic gradient descent (SGD) optimizer with a learning rate of 0.1, which is decaying after 10 epochs with a decay factor of 0.1. The models are trained on a batch size of 64 for 30 epochs in total.

The second experiment is conducted on ResNet18. However, we are using a standardized input resolution of

224×224 pixels, to avoid artifacts caused by the input resolution. We train the model on multiple datasets of different difficulties (in ascending order of complexity): MNIST, Cifar10, TinyImageNet, and the ImageNet dataset [21-24]. While it is hard to precisely define the complexity of the task, we think that the selected datasets can be regarded as increasingly difficult based on the number of classes and the complexity of the visual information provided as data points to the model. The resolution of MNIST binary images is 28×28 pixel. That is suitable for the 10-class classification problem. Cifar10 comprises RGB images with a 32×32 resolution. That is suitable for the 10-class classification problem as well. TinyImageNet consists of RGB images of size 64×64 with 200 classes, and ImageNet is made up of RGB images of various sizes belonging to 1,000 classes.

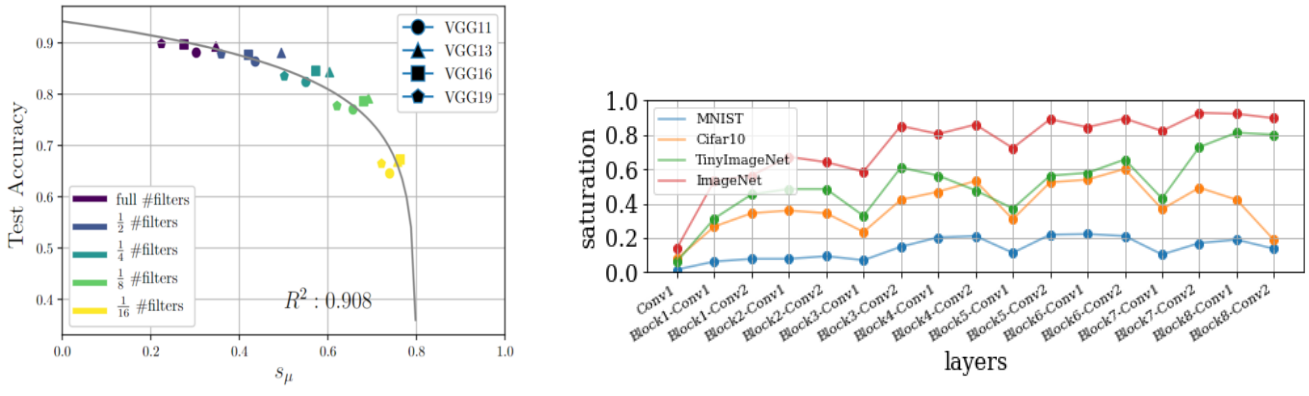
4.2. Results

When the capacity of the model is reduced, the average saturation s_μ increases, and the predictive performance decreases. The exponential reduction in capacity is reflected in a logarithmic relation between the increasing s_μ and predictive accuracy measured on the test set (see Fig. 3). From these observations, we can conclude that reducing the capacity of the architecture results in an increase in saturation. We further observe in Fig. 3 that saturation also increases with problem complexity. The saturation levels of all layers increase when the model is trained on a more difficult problem. The overall shape of the saturation curve only deviates slightly, with no tail pattern or similar anomalous shapes emerging. Since we know from the works of [13] that a resolution of 224×224 pixels results in an even



(a) (b) (c) (d) (e) (f) (g)

Fig. 2: Reconstructions of a single sample image, with the network being restricted to a percentage of its eigenfeatures: (a) Original: dimEncoding = 8192,(b) δ = 99.99%: $\dim E^K_{enc}$=4374, (c) δ = 99.9%: $\dim E^K_{enc}$=1626, (d) δ = 99.5%: $\dim E^K_{enc}$=1332, (e) δ = 99.0%: $\dim E^K_{enc}$=59, (f) δ = 95.0%: $\dim E^K_{enc}$=17, and (g) δ = 90.0%: $\dim E^K_{enc}$=1. Note how the visualized reconstructions degenerate with decreased explained variance,



(a) (b)

Fig. 3: Relationship of network saturation to model capacity and data complexity, (a) Accuracy and saturation with varying model capacity, and (b) ResNet18 saturation curves for different datasets. Reducing the number of filters and thus reducing the model capacity leads to an increase in the average saturation and a decrease in performance. Training a model on more difficult datasets also increases the overall saturation level. This indicates that saturation can measure the load on a ResNet18 model.

distribution of the inference process for the trained model, we can conclude that less processing is required for less complex problems. Combined with the insights gained from training the VGG-variants on Cifar10, we can conclude that for the pairs of dataset and model in our experiments, a saturation "sweet spot" exists between s_μ = 0.2 and s_μ=0.4, which yields good predictive performance without being too excessively over-parameterized. This sweet spot allows us to empirically formulate the algorithm for optimizing the network width proposed in our previous work [13]. This algorithm is depicted in Fig. 4. Since the experiment suggests a roughly linear relationship between the saturation values and the width scaling, the scaling parameter can be approximated from the average saturation.

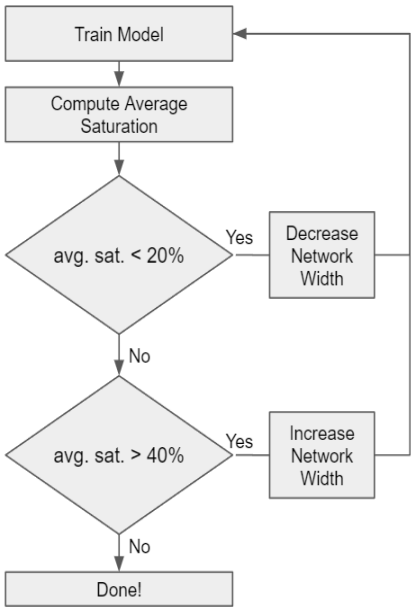


Fig. 4: This flow-chart depicts the basic procedure of optimizing the width of a neural architecture based on the average saturation of the model. The width of the network is increased to decrease saturation and vice versa until the model has an average saturation in the "sweet-spot"-range of 20-40%.

5. Experiment III: On the Emergence of Saturation Patterns

The tail pattern that we discussed earlier in this work allows for the identification of inefficiencies caused by mismatches between the neural architecture and the input resolution. However, since saturation can be computed life during training with little overhead [13], we think that it might be interesting to see how these patterns emerge during the training process.

5.1. Methodology

We first examine how the saturation levels evolve in a layer under different conditions. We train a set of multilayer perceptrons (MLP) with 3 fully connected layers. The first layer has 256 units, and the size of the second layer varies for each network, being in the range of 8, 16, 32, 64, 128, 256, 512, and 1024 units. We train these networks using the Adam optimizer and a batch size of 128 on Cifar10 using the native resolution of the dataset. The training is conducted twice. Once using 8 epochs, which is enough for all models to converge, whereas the second experiment is run for 30 epochs, which results in the loss increasing again due to overfitting. The hidden layer saturation is calculated after each epoch for observing the evolution of the architecture. We also calculate the cross-entropy loss of the model to observe a possible relationship between loss and saturation convergence. Based on these observations, we repeat the experiment on VGG11 and VGG19 as well as on sparse (low capacity) versions of these models with 1/8 of the original number of filters. We do this to understand if saturation patterns depend on the depth, architecture, and capacity of the network.

5.2. Results

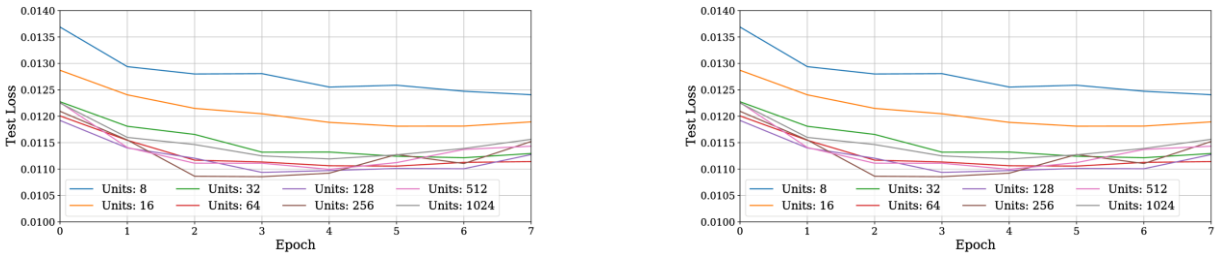
In Fig. 5, we observe that an increase in the number of units in the fully connected layer will result in a decreased saturation. However, the saturation does not change substantially during training, indicating that the inference process is not changed or shifted substantially inside the layer.

In Fig. 6, we can see that the increase in validation loss does not affect saturation. The fact that overfitting is not reflected in saturation values indicates that the changes to the way the data is processed when the model starts to overfit are subtle and thus are not reflected in changes to the relevant eigenspace and therefore saturation. This also means that saturation patterns in fully connected networks are independent of the training progress, which could allow for early detection of over- and under-parameterization during training. However, it also means that overfitting and convergence of the model need to be taken into consideration when analyzing saturation on fully connected neural networks, as these are not reflected by the saturation patterns.

In Fig. 7, we can see that saturation behaves substantially differently in convolutional neural networks, which exhibit a converging behavior towards a final pattern. This converging behavior is independent of the position of the layer in the network, the number of layers, and the capacity of the network, as Fig. 7 illustrates. Another interesting observation is that the tail pattern seems to be observable rather early during training, which indicates that an online analysis during training allows the data scientist to detect inefficiencies early, before the training has concluded.

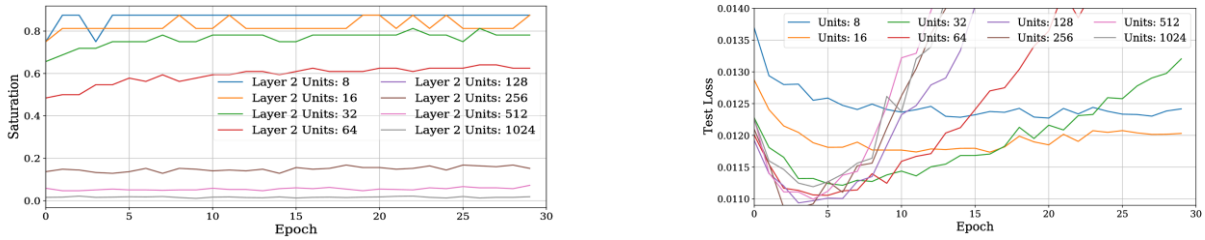
6. Experiments IV: Predictability of Tail Patterns Regarding Complexity

In the following, we examine how overall saturation affects the predictability of tail patterns. Richter et al. [14] show that the tail patterns in sequential convolutional neural networks can be predicted by computing the receptive field of all convolutional layers. The receptive field can be considered the field of view of a convolutional layer. Everything contained in the area spanned by the receptive field can hypothetically influence the value on a single position on the output feature map. In Section 5, we showed that changing the number of filters in a convolutional layer results in the changing of the global saturation level.



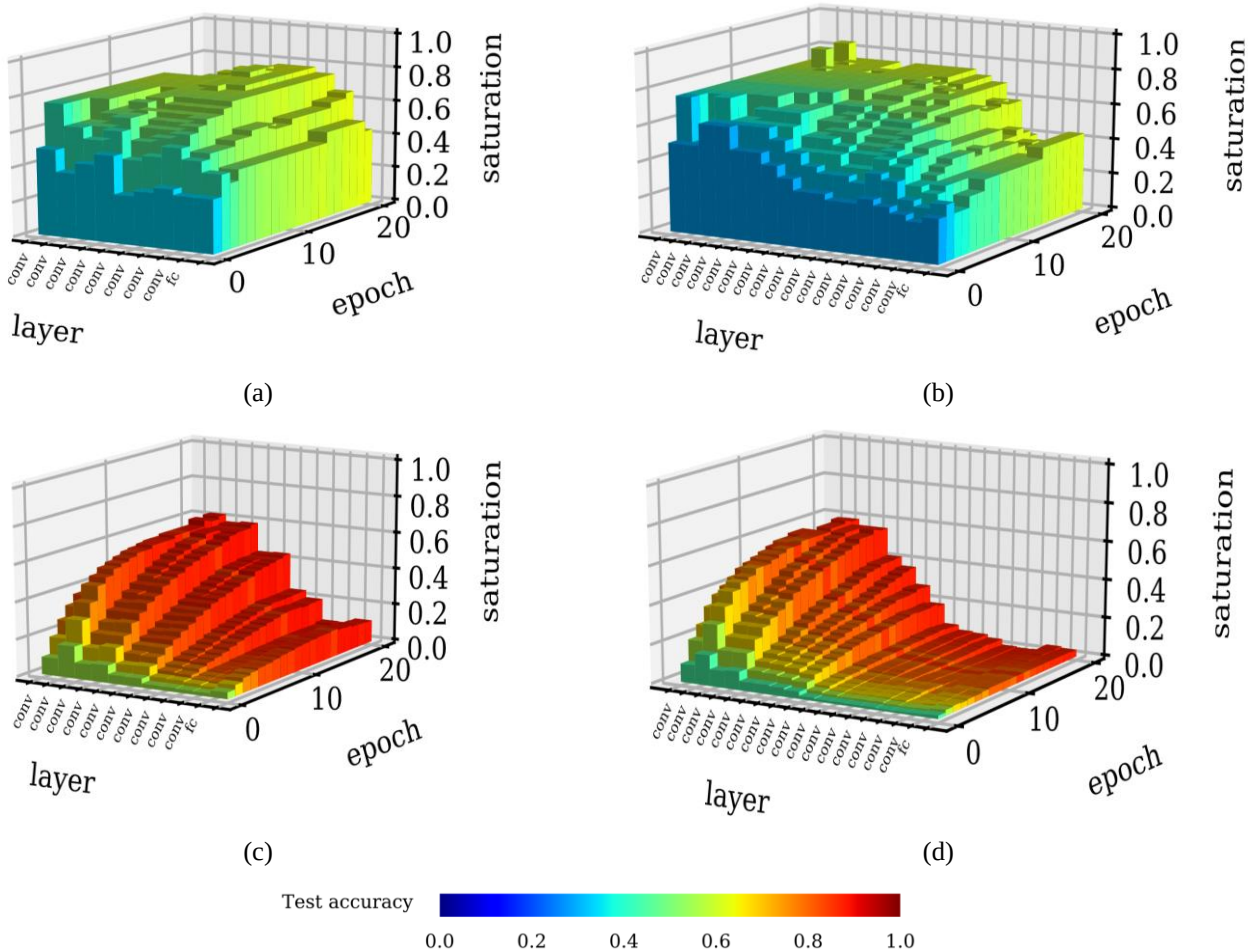
(a) (b)

Fig. 5: (a) Saturation of layer 2 during training, and (b) Validation loss during training. Saturation of a 3-layer MLP does not change substantially during training while the loss is converging.



(a) (b)

Fig. 6: (a) Saturation of layer 2 while overfitting, and (b) Saturation of layer 2 while overfitting. Saturation patterns of a 3-layer MLP are unaffected by overfitting, which indicates that overfitting is a process not affecting the overall dimensionality of the data inside the feature space.



(a) (b)

(c) (d)

Fig. 7: Saturations of convolutional neural networks, (a) VGG 11, (b) VGG 19 (Sparse), (c) VGG 13, and (d) VGG 19. There is a converging behaviour regarding saturation in contrast to previous observations in Fig. 5.

However, if the receptive field expansion is determining the number of unproductive layers, we will observe a tail pattern of unproductive layers starting at the 'border layer' [14].

6.1. Methodology

We test the hypothesis by repeating the experiments conducted by [14] regarding the prediction of unproductive layers. The authors of [14] were able to predict unproductive layers by computing the border layer for VGG11, 13, 16, and 19 on Cifar10. We reduce the capacity of these models by reducing the filter size to 1/8 of the original size to see whether a drastic loss in capacity changes how the inference is distributed. The models are trained for 30 epochs using the SGD-optimizer with a learning rate of 0.1, decaying by a factor of 0.1 every 10 epochs. The batch size is 64, each batch is channel-wise normalized, each image is randomly cropped during inference time as well as randomly horizontally flipped with a probability of 50%. The receptive field and the border layer are computed using the formulas provided by [14].

6.2. Results

Even though the capacity of the networks has been significantly reduced in every layer, the networks do not spread the inference process among significantly more layers (see Fig. 8). Based on these results, we conclude that the inference dynamics of the tested networks did not change substantially by reducing their capacity. This means that the capacity of layers primarily interacts with the difficulty of the problem, while the presence and absence of tail patterns interact with the receptive field, as exemplified by [14].

6.3. Different Types of Tail Patterns - A Brief Explanation

We find that saturation is subject to noise induced by certain features of the neural architecture. The increase or decrease in a number of filters from layer to layer, the use of 11 convolutions and downsampling layers are common culprits for zig-zag-like behavior or sudden dips and spikes in saturation. An example for the latter is DenseNet18 in Fig. 9b. It has to be stressed that these factors are not random nor create non-reproducible perturbations. Instead, they usually result in anomalous patterns that are very stable over multiple runs (which is exemplified by [13]). Logistic regression probes are considerably more robust against the aforementioned properties. However, they are influenced by the path that the information takes during the forward pass, revealing different types of tail patterns that can be differentiated based on the processing in the tail-layers. The three examples found commonly are exemplified in Fig. 9. These examples also give insights into how neural networks process information differently, which is the main reason why we dedicate an additional section to these findings in this paper. All the networks are trained on Cifar10 using a 3232 pixel input resolution. In Fig. 9a we find a pass-through tail, where the layers process the information but do not advance the quality of the intermediate solution. The second type of tail, depicted in Fig. 9b, is caused by the multiple pathways inside the DenseBlock of DenseNet. Information can pass from any previous layer to the current layer within the DenseBlock, effectively allowing the information to skip layers. When layers are skipped, the intermediate solution quality degrades and instantaneously recovers after the skipped section is over. The latter is apparent in the depicted example by the high model performance relative to the probe performance of the last DenseBlock layers. This phenomenon

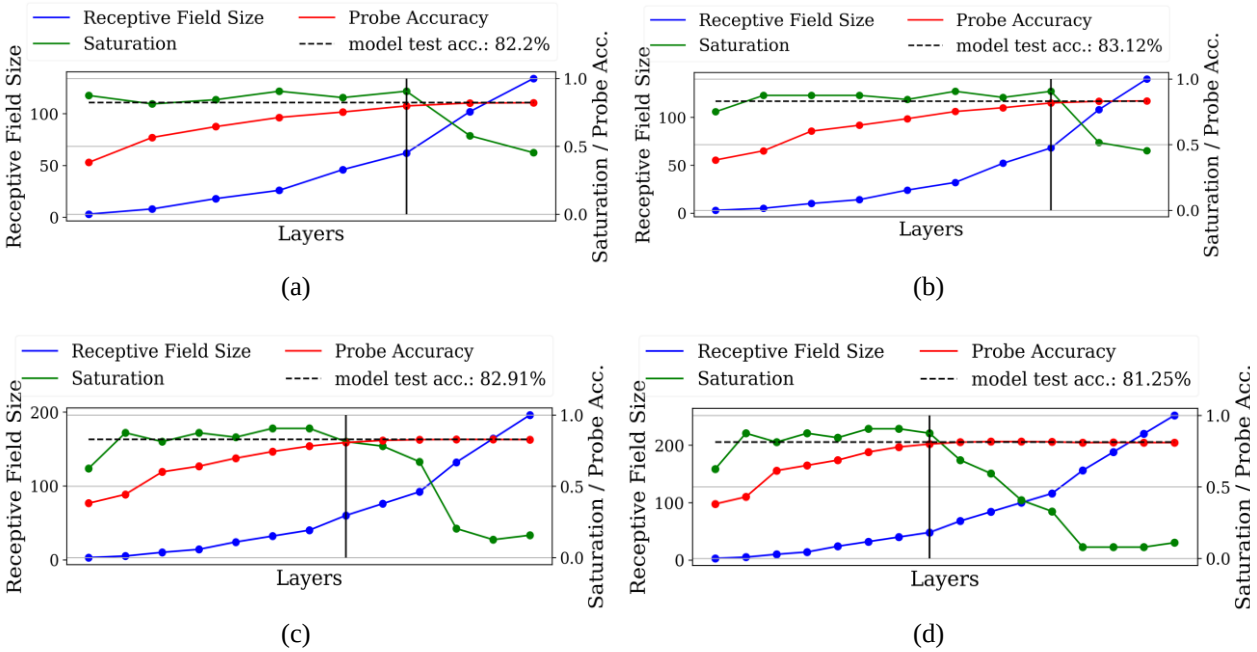


Fig. 8: Performance of the logistic regression probes past the border layer are miniscule, (a) VGG11 with ⅛ filters per layer, (b) VGG11 with ⅛ filters per layer, (c) VGG16 with ⅛ filters per layer, and (d) VGG19 with ⅛ filters per layer. The performance is improved even though the capacity of each layer is reduced to ⅛ of the original capacity. This indicates that the networks are unable to shift processing to otherwise unused layers even if the capacity is limited. This is consistent with observations made by Richter et al. [14].

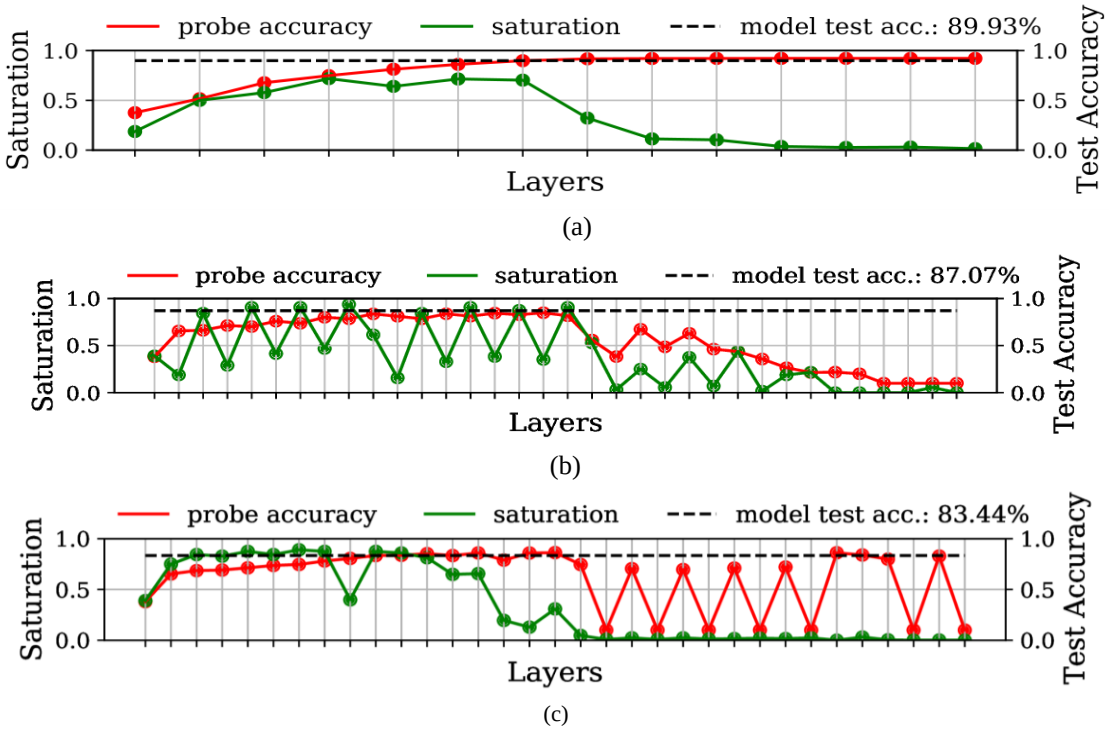


Fig. 9: (a) VGG16 tail layers maintain the quality of the intermediate solution, (b) The tail of DenseNet18 shows decay in probe performance, indicating that the last DenseBlock is skipped entirely [12], and (c) ResNet34 skips most residual blocks in the tail, which is apparent by the zig-zag pattern in probe performances caused by the starts and ends of skip-connections [12].

Depending on the neural architecture, tail patterns may deviate in their appearance in probe performance. In sequential architectures (a) the layers maintain the quality of the intermediate solution. If shortcut connections exist in the architecture, layers may be skipped. Skipped layers are apparent by their decaying probe performance [12]. This is apparent in DenseNet18 (b).

was initially observed in a simple MLP example by Alain and Bengio [12]. If necessary, the signal may jump more than a single building block in the architecture. An example of this can be seen in Fig. 9c on a ResNet34 architecture. This jumping is indicated by the zig-zag-pattern in the probe performance, where the higher performing layer resembles the first and the lower performing layer the second layer of a residual block. This shows that architecture decisions, which are influencing the potential pathways that the information can take from input to output, can have a significant influence on the way the model processes (or chooses not to process) information. In any case, the semantic of the tail-pattern remains unchanged, since a skipped layer and an unproductive layer can both be considered a parameter and computational inefficiency.

7. Conclusion

In this work, we explored the properties of the saturation metric in more detail and integrated this knowledge with insights from [13] and [14]. We have shown that model capacity and problem difficulty have opposite effects on the saturation value, as could be expected. A more surprising observation concerns the influence of individual layer capacity on the inference process: the tested models seem to be unable to shift processing to other layers, when some layers have substantially higher or lower capacity. An analysis of the evolution of saturation patterns during training revealed that they converge at a similar pace as the loss of the model, with saturation increasing during training. The way saturation evolves also gives hints on the properties of the dataset, but it is not influenced by the model over-fitting.

These insights allow us to expand upon the optimization strategies for neural architectures, proposed in our previous work. We demonstrate quantitatively that the average saturation of a model is indicative of over- and under-parameterization. This allows us to adjust the width of the model effectively. We further show that this property is independent of the tail pattern. The tail pattern is a symptom of a different design flaw, related to the depth of the neural network. Hence, we show that multiple axes of neural network design (depth and width) can be optimized in an informed manner using saturation. Our optimization strategies still require one or multiple training runs of the model, which could be seen as a disadvantage compared to pruning techniques like PCA-pruning [9-11]. On the other hand, our approach is architecture-independent, which we demonstrate on multiple experiments, while also being able to detect and resolve under-parameterization. The latter cannot be addressed by pruning algorithms. Furthermore, we demonstrate that saturation converges early during training, greatly reducing the cycle time of experiments, since pathological inefficiencies can be diagnosed before training has finished. However, the current approach is still too noisy to allow narrowing design decisions on a layer-by-layer basis.

While we demonstrate that tail-patterns are similar for different types of architectures, some architectural properties like down sampling and skip-connections induce artefacts

into the saturation values, making a layer-by-layer analysis harder to read. Therefore, we seek to combine this approach with the analysis of the receptive field, which was shown to greatly impact the presence of tail patterns [14], to make the diagnosis of inefficiencies more precise and robust.

Credit Authorship Contribution Statement

Mats Leon Richter: Software, Writing - original draft. **Leila Malihi:** Writing - review & editing. **Anne-Kathrin Patricia Windler:** Writing - review & editing. **Ulf Krumnack:** Writing - review & editing.

Declaration Of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. The ethical issues; including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, redundancy have been completely observed by the authors.

References

[1] M. Tan *et al.*, "MnasNet: Platform-aware neural architecture search for mobile," CoRR, vol. abs/1807.11626, 2018. Available: https://arxiv.org/abs/1807.11626

[2] M. Tan, and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in Proceedings of the 36th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 09–15 Jun 2019, pp. 6105–6114. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[4] K. Simonyan, and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[5] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016. [Online]. Available: https://arxiv.org/abs/1602.07261

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," CoRR, vol. abs/1512.00567, 2015. [Online]. Available: https://arxiv.org/abs/1512.00567

[7] C. Szegedy *et al.*, "Going deeper with convolutions," in Computer Vision and Pattern Recognition (CVPR), 2015. [Online]. Available: https://arxiv.org/abs/1409.4842

[8] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein, "SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[9] I. Garg, P. Panda, and K. Roy, "A Low Effort Approach to Structured CNN Design Using PCA," *IEEE Access*, vol. 8, pp. 1347-1360, 2020.

[10] W. Ahmed, S. Ansari, M. Hanif, A. Khalil, "PCA driven mixed filter pruning for efficient convNets," *PLoS ONE*, vol. 17, no. 1, article e0262386, 2022.

[11] I. Chakraborty, D. Roy, I. Garg Constructing, A. Ankit, and K. Roy "Energy-efficient mixed-precision neural networks through principal component analysis for edge intelligence", *Nature Machine Intelligence*, vol. 2, pp. 43–55, 2020.

[12] G. Alain, and Y. Bengio, "Understanding intermediate layers using linear classifier probes," ICLR 2017 workshop submission. [Online]. Available: https://arxiv.org/abs/1610.01644v4

[13] M. L. Richter, J. Shenk, W. Byttner, A. Arpteg, and M. Huss, "Feature space saturation during training," in *32st British Machine Vision Conference (BMVC)*, 2021.

[14] M. L. Richter, W. Byttner, U. Krumnack, L. Schallner, and J. Shenk, "(Input) size matters for CNN classifiers," in *Artificial Neural Networks and Machine Learning – ICANN 2021*. Springer International Publishing, 2021.

[15] M. L. Richter, L. Malihi, A.-K. P. Windler, and U. Krumnack, "Exploring the properties and evolution of neural network eigenspaces during training," in *2022 International Conference on Machine Vision and Image Processing*, 2022.

[16] J. Shenk, M. L. Richter, A. Arpteg, and M. Huss, "Spectral analysis of latent representations," 2019. [Online]. Available: http://arxiv.org/abs/1907.08589

[17] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, no. 6, pp.417–441, 1933.

[18] J. Shenk, M. L. Richter, W. Byttner, M. Marcinkiewicz "Delve: Neural Network Feature Variance Analysis", *Journal of Open Source Software*, vol. 7, no. 69, article 3992, 2022.

[19] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*, San Diego, CA, USA, May 7-9, 2015.

[20] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – mining discriminative components with random forests," in *Computer Vision– ECCV 2014*, Springer International Publishing, 2014, pp. 446–461.

[21] A. Krizhevsky, "Learning multiple layers of features from tiny images," MIT and NYU, Tech. Rep., 2009.

[22] Y. LeCun, and C. Cortes, "MNIST hand written digit database,", 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[23] Y. Le and X. Yang, "Tiny ImageNet Visual Recognition Challenge," 2015. [Online]. Available: http://cs231n.stanford.edu/reports/2015/pdfs/yle_project.pdf

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009.

BIOGRAPHY

Mats Leon Richter studied in the University of Osnabrück, where he received a Bachelor in Cognitive Science in 2017, a Bachelor in Computer Science in 2018 and a Master in Cognitive Science in 2019 with distinction. He worked full-time in the industry between 2017 and 2020 while pursuing his respective studies. Since 2019 Mats is enrolled in the PhD program in the University of Osnabrück, his main topics are Explainable AI and the Design of Convolutional Neural Network Classifiers. Since 2021 he also conducts research on Face Recognition Systems and Datasets as part of the KLIX Project.

Leila Malihi studied at the University of Shahid Chamran, Khuzestan, Iran, where she received a Bachelor and Master in Electrical engineering. Malihi's research interests include deep learning, transfer learning, and feature visualization. She has published 10 papers in Journals and conferences. She has 4 years of industrial experience and collaborations. Since 2021 she has also conducted research on Face Recognition Systems and Datasets as part of the KLIX Project.

Anne-Kathrin Patricia Windler studies at the University of Osnabrück, where she received a Bachelor in Cognitive Science and is finishing her Master in Cognitive Science in 2022. Her research interests include deep learning, face recognition and transfer learning.

Ulf Krumnack studied computational linguistics, artificial intelligence, and mathematics and received his PhD for work on the logical formalization of analogical reasoning. He currently works as a postdoc in the group of biologically inspired computer vision at the institute of cognitive science, University of Osnabrück, Germany. His research aims at the analysis of representations in deep networks, with a special focus on the visual domain.