A Taxonomy and Library for Visualizing Learned Features in Convolutional Neural Networks

Felix Grün FELIX.GRUEN@TUM.DE
Technische Universität München, Germany

Christian Rupprecht CHRISTIAN.RUPPRECHT@IN.TUM.DE
Technische Universität München, Germany, Johns Hopkins University, Baltimore, USA

Nassir Navab NAVAB@CS.TUM.EDU
Technische Universität München, Germany, Johns Hopkins University, Baltimore, USA

Federico Tombari TOMBARI@IN.TUM.DE
Technische Universität München, Germany, University of Bologna, Italy

Abstract

Over the last decade, Convolutional Neural Networks (CNN) saw a tremendous surge in performance. However, understanding what a network has learned still proves to be a challenging task. To remedy this unsatisfactory situation, a number of groups have recently proposed different methods to visualize the learned models. In this work we suggest a general taxonomy to classify and compare these methods, subdividing the literature into three main categories and providing researchers with a terminology to base their works on. Furthermore, we introduce the FeatureVis library for MatConvNet: an extendable, easy to use open source library for visualizing CNNs. It contains implementations from each of the three main classes of visualization methods and serves as a useful tool for an enhanced understanding of the features learned by intermediate layers, as well as for the analysis of why a network might fail for certain examples.

1. Introduction

In recent years, research in the field of artificial neural networks (ANN) has made tremendous progress in many diverse areas (Taigman et al., 2014; Hannun et al., 2014; He et al., 2015). Fueled by GPU accelerated learning and the development of regularization techniques like dropout (Hinton et al., 2012; Srivastava et al., 2014), activation functions like rectified linear units (ReLU) (Glorot et al., 2011), LeakyReLUs (Maas et al., 2013), and parametric ReLUs (He et al., 2015), and larger labeled datasets (Deng et al., 2009; Lin et al., 2014), natural image classification with convolutional neural networks (CNN) has become a *flagship example* (Yosinski et al., 2015) for the advancements in supervised learning using neural networks.

Yet, understanding the learned models is still an unsolved problem. Even the researchers who developed new techniques to improve CNN performance relied largely on trial and error (Zeiler & Fergus, 2013). Historically, the inner units of a network were regarded as a black box (Yosinski et al., 2015), appropriately affording them the name *hidden layers*. It is therefore not surprising that feature visualization for neural networks is a very young area of research.

The first approaches aimed at visualizing CNNs were made in 2013, first by Zeiler & Fergus (2013) and only one month later by Simonyan et al. (2013). Since then, a number of visualization methods have been proposed. Gaining an overview of the differences and similarities between the methods and the advantages and disadvantages of each has become a time consuming exercise. However, our careful analysis of the different methods for feature visualization shows that they can be grouped into just three classes. Together, these three classes comprise a novel taxonomy for feature visualization methods, which will provide a terminology in which to discuss existing research and from which to develop new directions and techniques.

Furthermore, we introduce the FeatureVis library for the MatConvNet toolbox (Vedaldi & Lenc, 2015), which contains implementations for several state-of-the-art visualiza-

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

tion methods to provide researchers and practitioners with an easy way to use the different methods for visualization in their own projects. Upon acceptance, the library will be publicly accessible and open source. It works with all CNNs built from the layers available with the MatConvNet toolbox and can easily be extended to incorporate custom layer types. The library can aid in the improvement of existing network architectures and the search for better models, as well as in understanding their performance.

2. A Taxonomy for Feature Visualization Methods

Our proposal of a taxonomy for feature visualization methods consists of three classes which we refer to as *Input Modification* methods, *Deconvolutional* methods, and *Input Reconstruction* methods. These classes are characterized by both, the goals for which the methods have been developed and the algorithms which they use.

2.1. Input Modification Methods

Input Modification methods are visualization techniques which modify, as the name suggests, the input and measure the resulting changes in the output or intermediate layers of the network. These methods see the network (or all the layers before the layer of interest) as a black-box and they are designed to visualize properties of the function this black-box represents. The resulting visualizations can be used to analyze the importance and locality of features in the input space. A prime example for this class of visualization methods is the *Occlusion* method by Zeiler & Fergus (2013).

To find the areas of the input image which are most important to the classification result, Zeiler & Fergus (2013) systematically cover up portions of the input image with a gray square and measure the change in activations. If an area of the image which is vital to the classification is covered up, the activations change noticeably. Covering up parts of the image which are unimportant to the computed activations, on the other hand, results only in a small difference. If the image is occluded systematically by moving the gray square with a fixed step width from left to right and top to bottom across the whole image, the resulting changes in activation form a heat map showing the importance of different areas of the input image.

Zhou et al. (2014) extend the Occlusion method proposed by Zeiler & Fergus (2013) by using a patch of randomized pixel values for the occlusion instead of a mono colored gray square. Since CNNs are very sensitive to edges, a mono colored area might contribute in unpredictable ways to the network output. Randomizing the pixel values reduces the risk of accidentally introducing new features into the image, which could be picked up by the CNN.

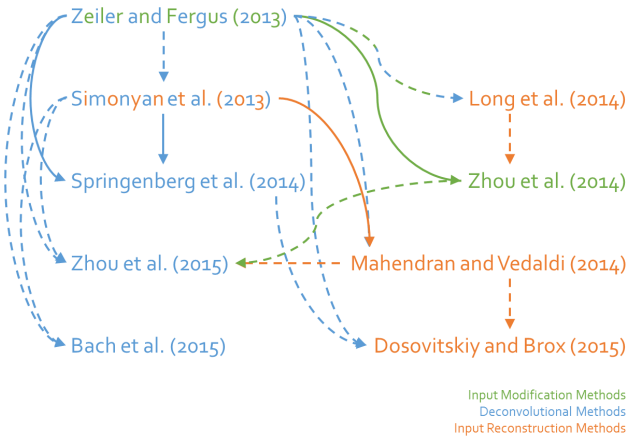

Figure 1. Influences between authors of papers on feature visualization for CNNs. Dashed lines indicate that one author cited another, while solid lines indicate that one author build upon and extended the work of another. The ordering is top down by publication year.

2.2. Deconvolutional Methods

In contrast to *Input Modification* methods, *Deconvolutional* methods see the network as a white-box and use the network structure itself for their visualizations. The common denominator amongst the different methods in this class is the idea to determine the contribution of one pixel of the input image by starting with the activation of interest and iteratively computing the contribution of each unit in the next lower layer to this activation. In this way, by moving backwards through the network until the input layer is reached, the contribution values for each pixel can be obtained, which together form a visualization of the features that are most relevant to the activation of interest.

The first methods of this class were proposed by Zeiler & Fergus (2013) and Simonyan et al. (2013). Zeiler & Fergus (2013) built on the work of Zeiler et al. (2011) and used a multi-layered Deconvolutional Network (*Deconvnet*) to project the activations from the feature space back to the input space. Deconvnets were originally proposed as a technique for unsupervised learning of CNNs, but can also be used to reverse the path of excitatory stimuli through the network to reveal which pixels and patterns of the input image are responsible for the observed activations.

Simonyan et al. (2013) build on work by Baehrens et al. (2010) in the context of Bayesian classification to present a variation of the Deconvnet approach. They reason that the derivative of the class score in the input space defines

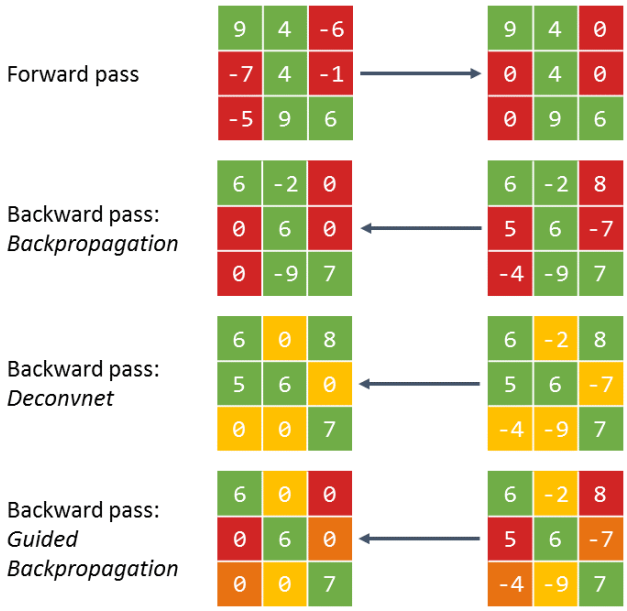

Figure 2. Different ways in which the pass through a ReLU layer affects contribution values for the Deconvnet method, Backpropagation, and Guided Backpropagation. The forward pass through the ReLU layer is shown for comparison.

the relative importance of the input pixels for the resulting class score. Intuitively, this is equal to finding those pixels that have to be changed the least for the greatest effect on the class score of interest. The underlying assumption is that the pixels that make up an object of a class in the input image have a greater contribution than pixels that constitute unrelated objects. To find the derivative of the class score with respect to the input image Simonyan et al. (2013) backpropagate the class scores from the last layer down through the network up to the input layer.

In the following year Springenberg et al. (2014) improved upon the previous two methods. They designed a CNN using only convolutional layers. To visualize the features learned by the resulting network they looked to the Deconvnet technique of Zeiler & Fergus (2013). Central to this approach of visualization are discriminative patterns encoded in the max pooling layers of the network. They contain the information on which pixels in the input space are of the highest importance for the resulting feature activations. Since the CNN developed by Springenberg et al. (2014) does not have max pooling layers, the original approach would not work.

Building on the work of Zeiler & Fergus (2013) and Simonyan et al. (2013), Springenberg et al. (2014) generalized the Deconvnet approach to work with networks that do not have max pooling layers and developed a new technique which enhanced the clarity and sharpness of the projection, even for networks that do. They named the resulting algorithm *Guided Backpropagation*.

Unlike the approaches presented so far, which specifically deal with the visualization of CNNs, Bach et al. (2015) aim at finding a general approach for visualizing classification decisions of nonlinear classifiers. To attain this goal they use pixel-wise decomposition to visualize the contributions of each pixel to the final classification decision. In their model, the inputs to the final classification layer have a certain relevance to that layer's activations and their inputs, in turn, have a certain relevance for their activations. This way, relevance flows from the final layer of the network down through the layers to the input image, where it can be mapped to pixels to show the relevance of each pixel for the classification result.

Despite the differences in the initial idea and the language used for description, careful analysis of this approach revealed that the algorithm which Bach et al. (2015) developed can be compared to the Deconvnet, Backpropagation of activations and Guided Backpropagation approaches described above. In fact, the main differences between the different approaches falling in the *Deconvolutional* methods class are the different ways in which they propagate the contribution values through ReLU and convolutional layers.

While Backpropagation uses the information from the original activations of the lower layer to decide which values to mask out during the pass through a ReLU layer, the Deconvnet approach uses the deconvolved activities from the higher layer. In other words, while Backpropagation uses information from the lower layers and the input image to determine which of the pixel values were important in computing the activations of the higher layers, the Deconvnet approach uses (gradient) information from the higher layers to determine which of the pixels from the input image had a beneficial effect on the activations of the units of interest. Guided Backpropagation combines the two approaches by masking out the value of any unit that had either a negative activation during the forward pass or a negative contribution value during the backward pass. Figure 2 provides an overview over how ReLU layers affect contribution values for the different methods.

The last contribution to the class of *Deconvolutional* methods is made by Zhou et al. (2015). Their approach requires a CNN which performs global average pooling on the convolutional feature maps just before the final output layer. By using the spatial information of the activation maps before the pooling operation and the values of the output layer weights they construct a *Class Activation Map* showing which areas of the input image were important for the output class of interest. Specifically, a Class Activation Map

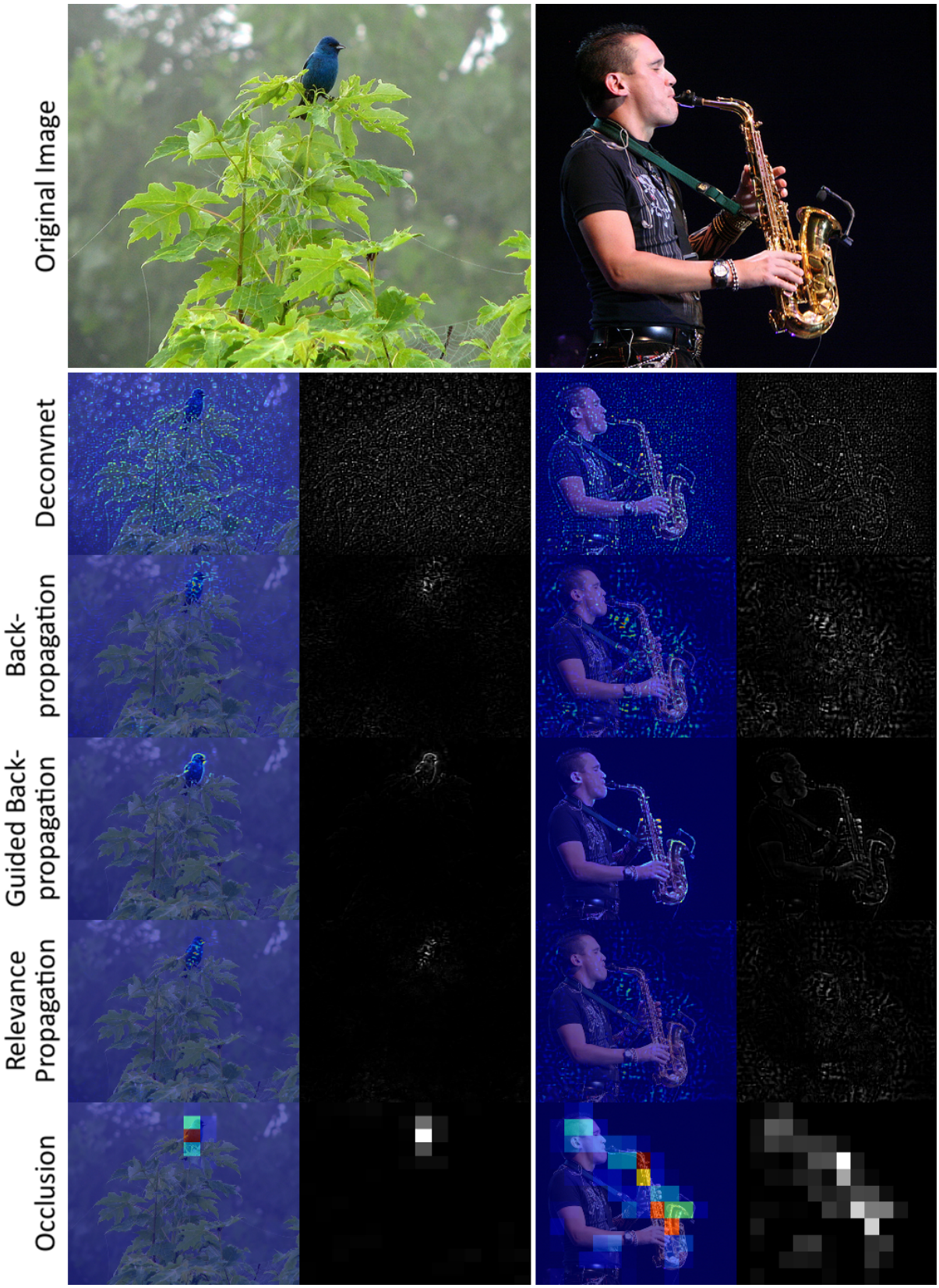

Figure 3. Comparing different visualization methods for two images. Both images where classified correctly with high confidence by the network. The image classes are: *indigo bunting, indigo finch, indigo bird, Passerina cyanea* and *sax, saxophone*. Visualizations for the respective image class using a VGG-16 network and an epsilon value of 0.001 for the Relevance Propagation method.

is the sum of all activation maps before the average pooling operation weighted by the output layer weights connecting the features to the class of interest.

2.3. Input Reconstruction Methods

The methods belonging to the third class, i.e. *Input Reconstruction* methods, follow the idea that reconstructing an input, which either maximally activates a unit of interest or leads to the same activations as a natural image prior, reveals which features are important to the associated filters. Here, we take a closer look at three of the proposed methods from this category.

Long et al. (2014) developed a new way of visualizing the feature detection abilities of a CNN to find out if CNNs learn interclass correspondence. They built on the *HOGgles* approach (Vondrick et al., 2013) of feature visualization for object detectors, but use replacement with the top-k nearest neighbors in feature space instead of paired dictionary learning. Intuitively, they replace patches of the original input image with patches from other images that lead to the same activations at the layer of interest. This allowed them to evaluate the extent to which CNNs are able to abstract from the input image and recognize higher level features at different layers in the network.

Simonyan et al. (2013) on the other hand propose a generative class model visualization method which builds upon work done by Erhan et al. (2009) in the context of visualizing deep belief networks (DBN) and deep unsupervised auto-encoders. They use gradient descent in the input space to iteratively find an image which maximally activates a unit of choice. To find the gradient, they backpropagate the difference between the actual activation and the desired activation to the input layer without changing the network itself. The resulting image contains all the features that the associated filters selects for.

In the following year, Mahendran & Vedaldi (2014) tried to invert the image representation encoded in the network output to obtain a reconstruction of the input image. Analyzing which features were present and absent in the reconstructed image would enable a better understanding of the information processed by the network. To this end, they built on the generative class model visualization method of Simonyan et al. (2013) and made a number of important contributions.

The *Input Reconstruction* method of Simonyan et al. (2013) and Mahendran & Vedaldi (2014) needs strong regularizers so that the resulting images and the features which they contain can be recognized and interpreted by the human observer. While Simonyan et al. (2013) used the L_2 norm as a regularizer, Mahendran & Vedaldi (2014) found out, that an L_p norm with higher values for p leads to clearer images. In addition to that, they introduce total variation as a second regularizer to be used in conjunction with the L_p norm. This markedly improved the quality of the end result and lead to more natural looking images.

In contrast to the two methods above using replacement and gradient descent, Dosovitskiy & Brox (2015) use generative networks, i.e. CNNs turned upside down (Dosovitskiy et al., 2014), to invert the image representation. A generative network receives the representation of an image, e.g. the output of a "normal" CNN, as input and computes a reconstruction of the original image. A drawback of this method is that each generative network has to be trained to invert a specific representation before it can be used. This is balanced by the benefit of computing reconstructions several orders of magnitude faster than using gradient descent, after training is complete.

3. The FeatureVis library

To facilitate the use of feature visualization methods for CNNs, we developed an open source library named FeatureVis, built on top of the popular MatConvNet toolbox. MatConvNet implements CNNs for MATLAB and provides different layer types as building blocks that allow for fast prototyping of new CNN architectures (Vedaldi & Lenc, 2015).

The FeatureVis library is continually growing and already implements functions for many popular feature visualization methods from all three classes. For the deconvolution approach three different methods for the propagation of activations through ReLUs, Deconvnet (Zeiler & Fergus, 2013), Backpropagation (Simonyan et al., 2013) and Guided Backpropagation (Springenberg et al., 2014), and two different methods for the propagation of activations through convolutional layers, Backpropagation and Relevance Propagation (Bach et al., 2015), have been implemented. For the occlusion approach the size of the occluding box and the stride rate can be specified independently for the two dimensions. Experiments with the FeatureVis library showed that the color of the occlusion box matters in determining the importance of occluded features. Therefore, as an improvement over Zeiler & Fergus (2013), the occluded area will either be filled with a color manually chosen by the user or random values as used by Zhou et al. (2014). For the input reconstruction approach using gradient descent the user can choose and combine the L_p norm regularizer (Mahendran & Vedaldi, 2014; Simonyan et al., 2013) and total variation (Mahendran & Vedaldi, 2014).

Figure 3 shows a comparison of different visualization methods included in the FeatureVis library applied to two example images. The visualizations were created using a VGG-16 network trained on the 1000 classes of

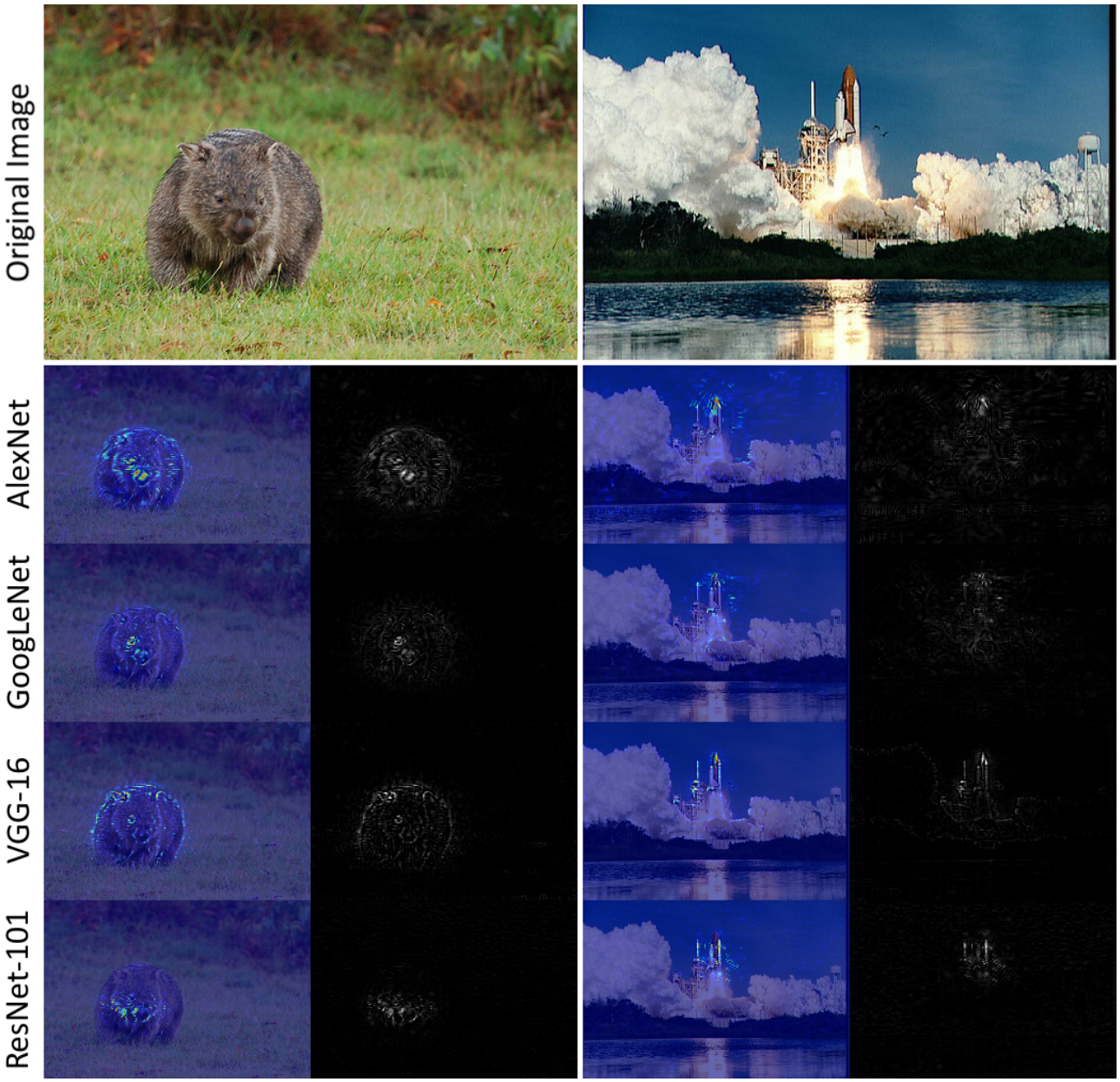

Figure 4. Comparative visualizations of different networks for two images. Both images where classified correctly with high confidence by all networks. Visualizations for the respective image class using Guided Backpropagation. The image classes are: *wombat* and *space shuttle*.

the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015). The images very clearly show the progress in *Deconvolutional* methods, with Guided Backpropagation offering the sharpest visualizations of the features important for the classification of the input images.

An interesting use-case of the FeatureVis library is the comparison of different network architectures. Figure 4 shows visualizations of two images for four different network architectures: AlexNet (ILSVRC top-1 error rate of 41.8%), GoogLeNet (34.2%), VGG-16 (28.5%), and ResNet-101 (23.4%). It is interesting to see that smaller error rates go hand in hand with more sharply focused features contributing to the final classification results. This can be seen best in the black and white visualizations of the pixel contributions for the space shuttle image.

This is a very good example of how visualization methods can facilitate the understanding of the performance of different network architectures. Additionally, it is noteworthy that most visualization techniques are not confined to classification tasks. FeatureVis is agnostic to the loss-layer of the network and can therefore also be used for networks with regression tasks such as human pose estimation, facial landmark detection, semantic segmentation, and depth prediction.

4. Conclusion

In this paper we have proposed a taxonomy for feature visualization techniques for deep networks, subdividing the literature into three main classes. For each class, we have described its defining characteristics and summarized the related literature. Together with the taxonomy we have introduced the open source library FeatureVis for MatConvNet. The library contains implementations of a number of popular feature visualization methods, to be used with any CNN designed using the standard MatConvNet layers with no further setup required. It thus provides a useful tool for the visual analysis of deep models and for direct improvements of a user's network architecture.

While the FeatureVis library can already be used to visualize CNNs using a variety of different methods, more remains to be done. We will focus our efforts on extending the FeatureVis library to incorporate more visualization methods and provide interactive real-time visualizations to assist the comparison of different methods and visualize the influence of different parameters.

References

Bach, Sebastian, Binder, Alexander, Montavon, Grégoire, Klauschen, Frederick, Müller, Klaus-Robert, and Samek, Wojciech. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):1–46, 07 2015. doi: 10.1371/journal.pone.0130140. URL http://dx.doi.org/10.1371%2Fjournal.pone.0130140.

Baehrens, David, Schroeter, Timon, Harmeling, Stefan, Kawanabe, Motoaki, Hansen, Katja, and Müller, Klaus-Robert. How to explain individual classification decisions. *J. Mach. Learn. Res.*, 11:1803–1831, August 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1859912.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848.

Dosovitskiy, Alexey and Brox, Thomas. Inverting convolutional networks with convolutional networks. *CoRR*, abs/1506.02753, 2015. URL http://arxiv.org/abs/1506.02753.

Dosovitskiy, Alexey, Springenberg, Jost Tobias, and Brox, Thomas. Learning to generate chairs with convolutional neural networks. *CoRR*, abs/1411.5928, 2014. URL http://arxiv.org/abs/1411.5928.

Erhan, Dumitru, Bengio, Yoshua, Courville, Aaron, and Vincent, Pascal. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341, 2009.

Glorot, Xavier, Bordes, Antoine, and Bengio, Yoshua. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

Hannun, Awni Y., Case, Carl, Casper, Jared, Catanzaro, Bryan C., Diamos, Greg, Elsen, Erich, Prenger, Ryan, Satheesh, Sanjeev, Sengupta, Shubho, Coates, Adam, and Ng, Andrew Y. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014. URL http://arxiv.org/abs/1412.5567.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. URL http://arxiv.org/abs/1502.01852.

Hinton, Geoffrey E., Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/abs/1207.0580.

Lin, Tsung-Yi, Maire, Michael, Belongie, Serge J., Bourdev, Lubomir D., Girshick, Ross B., Hays, James, Perona, Pietro, Ramanan, Deva, Dollár, Piotr, and Zitnick, C. Lawrence. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

Long, Jonathan, Zhang, Ning, and Darrell, Trevor. Do convnets learn correspondence? *CoRR*, abs/1411.1091, 2014. URL http://arxiv.org/abs/1411.1091.

Maas, Andrew L, Hannun, Awni Y, and Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, pp. 1, 2013.

Mahendran, Aravindh and Vedaldi, Andrea. Understanding deep image representations by inverting them. *CoRR*, abs/1412.0035, 2014. URL http://arxiv.org/abs/1412.0035.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Simonyan, Karen, Vedaldi, Andrea, and Zisserman, Andrew. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013. URL http://arxiv.org/abs/1312.6034.

Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, and Riedmiller, Martin A. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014. URL http://arxiv.org/abs/1412.6806.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Taigman, Yaniv, Yang, Ming, Ranzato, Marc'Aurelio, and Wolf, Lior. Deepface: Closing the gap to human-level performance in face verification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

Vedaldi, A. and Lenc, K. MatConvNet – convolutional neural networks for MATLAB. In *Proceedings of the ACM International Conference on Multimedia*, 2015.

Vondrick, Carl, Khosla, Aditya, Malisiewicz, Tomasz, and Torralba, Antonio. Hoggles: Visualizing object detection features. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1–8, 2013.

Yosinski, Jason, Clune, Jeff, Nguyen, Anh Mai, Fuchs, Thomas, and Lipson, Hod. Understanding neural networks through deep visualization. *CoRR*, abs/1506.06579, 2015. URL http://arxiv.org/abs/1506.06579.

Zeiler, M. D., Taylor, G. W., and Fergus, R. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2018–2025, Nov 2011. doi: 10.1109/ICCV.2011.6126474.

Zeiler, Matthew D. and Fergus, Rob. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. URL http://arxiv.org/abs/1311.2901.

Zhou, Bolei, Khosla, Aditya, Lapedriza, Àgata, Oliva, Aude, and Torralba, Antonio. Object detectors emerge in deep scene cnns. *CoRR*, abs/1412.6856, 2014. URL http://arxiv.org/abs/1412.6856.

Zhou, Bolei, Khosla, Aditya, Lapedriza, Àgata, Oliva, Aude, and Torralba, Antonio. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150, 2015. URL http://arxiv.org/abs/1512.04150.