Grad-LAM: Visualization of Deep Neural Networks for Unsupervised Learning

Alexander Bartler, Darius Hinderer and Bin Yang
Institute of Signal Processing and System Theory, University of Stuttgart
Stuttgart, Germany
Email: {alexander.bartler, bin.yang}@iss.uni-stuttgart.de
darius.hinderer@gmx.de

***Abstract*—Nowadays, the explainability of deep neural networks is an essential part of machine learning. In the last years, many methods were developed to visualize important regions of an input image for the decision of the deep neural network. Since almost all methods are designed for supervised trained models, we propose in this work a visualization technique for unsupervised trained autoencoders called Gradient-weighted Latent Activation Mapping (Grad-LAM). We adapt the idea of Grad-CAM and propose a novel weighting based on the knowledge of the autoencoder's decoder. Our method will help to get insights into the highly nonlinear mapping of an input image to a latent space. We show that the visualization maps of Grad-LAM are meaningful on simple datasets like MNIST and the method is even applicable to real-world datasets like ImageNet.**

***Index Terms*—deep visualization, transparency, unsupervised learning, explainable artificial intelligence, Grad-LAM**

I. Introduction

In the last years, the interest in understanding deep neural networks has grown significantly in the context of explainable artificial intelligence. Especially in supervised learned deep networks, many methods were developed [1]–[4], e.g. the Gradient-weighted Class Activation Mapping (Grad-CAM). They try to visualize which regions or pixels of the input image cause the predicted result and are widely used today. Besides supervised learned models, methods for learning representations in an unsupervised or self-supervised manner also show promising results [5]–[10]. For those methods, it would be even more interesting to understand which regions or pixels of the input are producing the respective latent representation.

Previous works on analyzing unsupervised trained models use mainly the weights, the activations of the model or a weighted sum of activations based on the weights of the previous layer [11, 12]. We propose in this work an adaptation of Grad-CAM, which is able to generate visualization maps for the encoder of a fully unsupervised trained autoencoder. We call our algorithm Gradient-weighted Latent Activation Mapping (Grad-LAM) since it uses the latent variables instead of the class-specific output. In a first step, this results in one gradient-weighted map for each latent variable. We further propose a way to combine those maps into one saliency map using the knowledge of the decoder. Our contributions are as follows:

- We adapt the idea of Grad-CAM [2] to visualize unsupervised trained autoencoders.
- We propose a decoder-based weighting technique to get a single saliency map of the encoder.
- We analyze the results and show the feasibility of Grad-LAM with three datasets.

II. Related Work

We first outline the idea of Grad-CAM. It is based on the activation maps of one intermediate layer of a supervised trained network for e.g. a classification task. Those intermediate activation maps are weighted and summed up to a single saliency map. To calculate the weights, the authors propose a gradient-based approach. They take the class or in general the desired target of the network and calculate its gradients with respect to the pixels of the respective activation maps. After a global average pooling of those gradients, the weights are used for the weighted sum of activation maps followed by cutting off negative values. This results in the Grad-CAM, where high values in the saliency map indicate a positive impact on the desired decision of the network. Usually, the calculated map is rescaled to the input size of the network and overlayed with the corresponding input image.

III. Grad-LAM

Compared to Grad-CAM for supervised trained models, the desired visualization result for unsupervised trained models is in general difficult to specify. The problem formulation is ill-posed because unsupervised trained models are not trained for solving specific tasks and therefore are not forced to focus on specific parts of the input image compared to problems like classification. Hence, we do not expect similar results as for Grad-CAM. For an autoencoder trained to reconstruct the input image based on a latent representation, the information of all input pixels could be important to encode the latent variables. Especially for real-world images, we expect that the visualization will probably highlight almost all image pixels as their information should be encoded into the latent variables in order to provide a good reconstruction. For simple datasets like MNIST, the visualization should mainly highlight the pixels of the digits because there is no big need for focusing on the homogeneous background to encode the complete image. If not all necessary input pixels are highlighted by a saliency map, this could indicate a loss of information in the encoded latent representation. Furthermore, a visualization map for

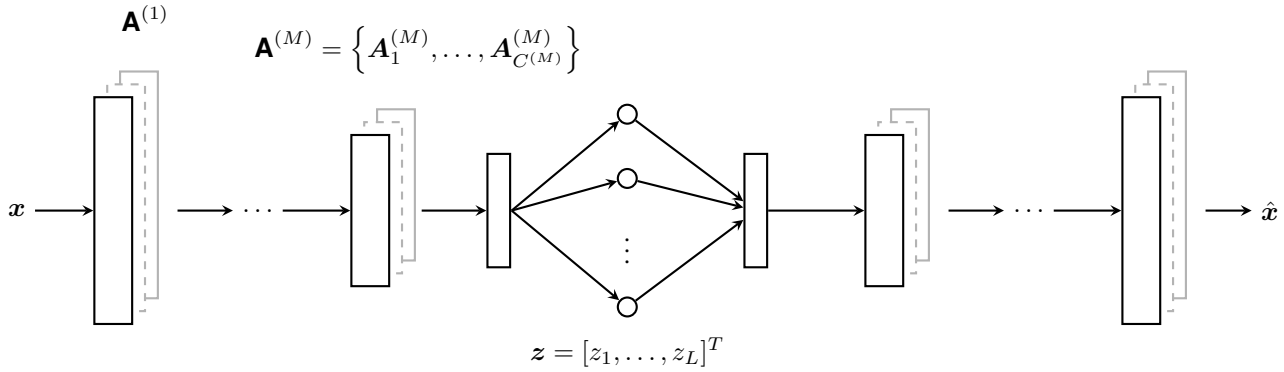


Fig. 1: Overview of the autoencoder $\mathcal{A}$ with the respective notations

each latent variable could help to show which information is encoded into which individual latent variable, thus providing a new technique to analyze the latent variables of an autoencoder.

For a better understanding, the notations are first introduced.

A. Notations

Let $\boldsymbol{x} \in \mathbb{R}^{H^{(0)} \times W^{(0)} \times C^{(0)}}$ be an arbitrary input image with height $H^{(0)}$, width $W^{(0)}$ and the number of channels $C^{(0)}$. The corresponding reconstruction of the autoencoder $\mathcal{A}$ is denoted by $\hat{\boldsymbol{x}}$. The autoencoder $\mathcal{A}$ consists of two parts, the encoder $\mathcal{E}$ which encodes a latent vector $\boldsymbol{z} \in \mathbb{R}^L$ and the decoder $\mathcal{D}$ which reconstructs the image from the latent variables. Thus, $\hat{\boldsymbol{x}} = \mathcal{A}(\boldsymbol{x}) = \mathcal{D}(\mathcal{E}(\boldsymbol{x}))$ and $\boldsymbol{z} = \mathcal{E}(\boldsymbol{x})$. An overview of the autoencoder $\mathcal{A}$ is shown in Fig. 1.

The output activation maps after the m-th convolution block of the encoder with M blocks is denoted as $\mathbf{A}^{(m)} \in \mathbb{R}^{H^{(m)} \times W^{(m)} \times C^{(m)}}$, where $H^{(m)}$ and $W^{(m)}$ are the height and width of the respective activation maps. The number of activation maps is denoted by $C^{(m)}$.

B. Visualization of latent variables

For the visualization at the m-th convolution block, we want to weight and sum up the $C^{(m)}$ activation maps $\mathbf{A}^{(m)}$. Therefore, we calculate similar to Grad-CAM the weights with the help of the gradient of the latent variables z_l w.r.t the activation maps. The weights $\alpha_{c,l}^{(m)}$ are calculated by Eq. (1) with $c \in \{1, \ldots, C^{(m)}\}$ and $l \in \{1, \ldots L\}$.

$$\alpha_{c,l}^{(m)} = \frac{1}{H^{(m)} \cdot W^{(m)}} \sum_{h=1}^{H^{(m)}} \sum_{w=1}^{W^{(m)}} \frac{\partial z_l}{\partial \boldsymbol{A}_c^{(m)}} \tag{1}$$

In a second step, we calculate the visualization map $\boldsymbol{M}_l^{(m)}$ for the latent variable z_l by

$$\boldsymbol{M}_l^{(m)} = \sum_{c=1}^{C^{(m)}} \alpha_{c,l}^{(m)} \boldsymbol{A}_c^{(m)}. \tag{2}$$

There are two differences to Grad-CAM. First, we do not restrict the values of $\boldsymbol{M}_l^{(m)}$ to be positive because the latent variables are real-valued in contrast to the non-negative softmax values for classification. Second, Grad-CAM was mainly developed for supervised classification and the weights are calculated starting from the output neuron of the true class. Grad-LAM on the other hand performs a visualization for unsupervised representation learning. Before the learned representation is used in any supervised task, all latent variables $z_1, \ldots, z_L$ might be important. In fact, a visual inspection of the L maps $\boldsymbol{M}_l^{(m)}, 1 \leq l \leq L$, may help to better understand which latent variable does encode which part of the input image. Nevertheless, there are situations where a single map is desired to visualize the learned overall latent representation. For this purpose, we propose two methods to combine the L maps $\boldsymbol{M}_l^{(m)}$ into one saliency map.

C. Uniform Weighting

One trivial approach is a uniform weighting of all L maps:

$$\boldsymbol{M}_{\text{Uniform}}^{(m)} = \frac{1}{L} \sum_{l=1}^{L} \boldsymbol{M}_l^{(m)}. \tag{3}$$

As we will show in the experiments, the assumption of equal contribution of each latent variable is in general not applicable. In the uniform weighting, maps with less visual information are treated equally as those with rich content. This leads to a bad visualization.

D. Decoder Importance Weighting

To overcome the problem mentioned above, we propose an improved weighting by using the information of the decoder $\mathcal{D}$. The motivation is that the decoder knows the importance of each latent variable. If a latent variable contains important information about the input image, the decoder will focus more on it to reduce the reconstruction loss during training. We calculate the decoder weights based on the impact of the latent variables on the average of the reconstructed pixels. Thus, the weights for different latent variables are calculated using the derivative of the sum of the reconstruction output w.r.t. to each latent variable as shown below.

$$\tilde{\beta}_l = \frac{\partial}{\partial z_l} \sum_{h=1}^{H^{(0)}} \sum_{w=1}^{W^{(0)}} \sum_{c=1}^{C^{(0)}} \hat{x}_{h,w,c} \tag{4}$$

The absolute value of the weights is large if a small change of the latent variable causes large changes on many output pixels. Before applying the weights $\tilde{\beta}_l$, we normalize them to absolutely sum up to 1 to have a fair comparison to the uniform weighting method:

$$\beta_l = \frac{\tilde{\beta}_l}{\sum_{l=1}^{L} |\tilde{\beta}_l|}. \tag{5}$$

The final decoder weighted visualization map for the m-th layer is

$$\boldsymbol{M}_{\text{Decoder}}^{(m)} = \sum_{l=1}^{L} \beta_l \boldsymbol{M}_l^{(m)}. \tag{6}$$

An alternative to calculate the weights $\tilde{\beta}_l$ could be to use the sum over the absolute value of the gradient of each output pixel w.r.t. each latent variable $|\frac{\partial}{\partial z_l}\hat{x}_{h,w,c}|$. Preliminary experiments showed that this leads to worse visual explanations and a larger computational complexity since not only a gradient of one scalar has to be calculated as in Eq. (4).

IV. Experimental Setup

In this section, we describe the experiment setup including model, training and the used datasets to evaluate Grad-LAM.

A. Architecture and Training

We use a simple architecture for our autoencoder $\mathcal{A}$. Depending on the input size of the used dataset, we build our model with M downsampling convolutional blocks (DownBlock) in the encoder $\mathcal{E}$ and M upsampling convolutional blocks (UpBlock) in the decoder $\mathcal{D}$. Each DownBlock consists of two LeakyReLU [13] (with $\alpha = 0.2$) convolutional layers, each followed by batch normalization [14]. The first convolution of each block uses a stride of 2 and therefore performs a spatial downsampling by the factor of 2. The first block uses $C^{(1)}$ filters and the number is doubled after each block. After the last DownBlock, the output is reshaped and followed by a linear fully connected layer to the dimension of the latent vector $\boldsymbol{z} \in \mathbb{R}^L$. For decoding the latent code at the input of $\mathcal{D}$, there is one dense layer followed by reshaping. Afterwards, there are M UpBlocks each consisting of two LeakyReLU convolutional layers with batch normalization. The first convolution of each block is a transposed convolution with a stride of 2 to perform a spatial upsampling by the factor of 2. The number of filters of the first UpBlock is $C^{(m)}$ and is divided by 2 after each block. An overview about the architecture is shown in Tab. I.

In all experiments, we train our model for 50 epochs with a batch size of 256 except for ImageNet, where we use a batch size of 64. For optimizing the model parameters, we use mean squared error as reconstruction loss and update the parameters with Adam [15] with a learning rate of 0.001.

B. Datasets and Parameter

We use the datasets MNIST [16], Fashion MNIST [17] and ImageNet [18]. As shown in Tab. II, we adapted the number of blocks M and the number of latent variables L to fit the model capacity to the respective dataset. We resized the MNIST and Fashion MNIST dataset to $32 \times 32 \times 1$ for convenience.

TABLE I: Autoencoder $\mathcal{A}$ architecture

Layer	Repeat	Output size
Input		$H^{(0)} \times W^{(0)} \times C^{(0)}$
DownBlock	$\times M$	$H^{(m)} \times W^{(m)} \times C^{(m)}$
Latent		L
Dense/Reshape		$4 \times 4 \times C^{(M)}$
UpBlock	$\times M$	$H^{(M-m)} \times W^{(M-m)} \times C^{(M-m)}$
Output		$H^{(0)} \times W^{(0)} \times C^{(0)}$

TABLE II: Datasets and the used parameters

Dataset	Dimension	M	L	$C^{(1)}$
MNIST	$32 \times 32 \times 1$	3	10	32
Fashion MNIST	$32 \times 32 \times 1$	3	20	32
ImageNet	$256 \times 256 \times 3$	6	50	32

C. Visualization

To show the visualization maps $\boldsymbol{M}_l^{(m)}$ of the latent variables, we normalize all maps by the largest absolute value of all maps. The combined maps $\boldsymbol{M}_{\text{Uniform}}^{(m)}$ or $\boldsymbol{M}_{\text{Decoder}}^{(m)}$ are normalized by its largest absolute value. Therefore, the shown maps are in range of -1 (blue) to 1 (red) and are resized to the size of the corresponding input image with bilinear interpolation. In general, Grad-LAM could be applied to all layers of the encoder. In preliminary experiments we have found that the visualization for the shallow layers results in maps without a semantic meaning and for the deepest layer results in coarse maps. Therefore, we compute and study the visualization maps at a spatial resolution of 8×8, i.e. for layer $m = 2$ for MNIST and Fasion MNIST and layer $m = 5$ for ImageNet.

V. Evaluation of Visualization Maps

A. MNIST

In the first experiment, we apply Grad-LAM to the autoencoder trained on MNIST. As already mentioned, we select the second block $\mathbf{A}^{(2)}$ of $\mathcal{E}$ for applying our proposed method. The resulting 10 latent visualization maps corresponding to 10 latent variables for an example digit 0 (top row) and 9 (bottom row) of the test set are shown in Fig. 2.

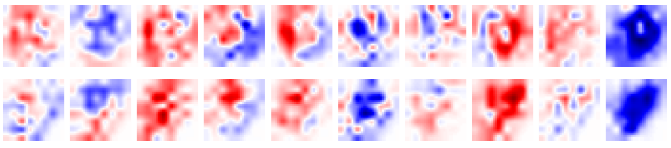
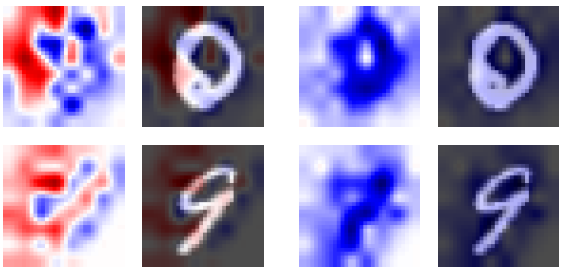


Fig. 2: $\boldsymbol{M}_l^{(2)}$ for all latent variables for digit 0 (top row) and digit 9 (bottom row)

As it can be seen in Fig. 2, the maps of the individual latent variables are not easy to interpret. Some of them seem to visualize the area where the digit is located but others highlight areas in the background or only some small regions of the input image. This implies, some latent variables may not encode any meaningful content while others contain most of the image content. For example, the 8th and 10th map in the first row show that those latent variables are generated with the focus mainly on the digit itself. Fig. 3 shows the result of uniform and decoder weighted saliency map in column 1 and 3 and the overlay with their corresponding input image $\boldsymbol{x}$ in column 2 and 4.



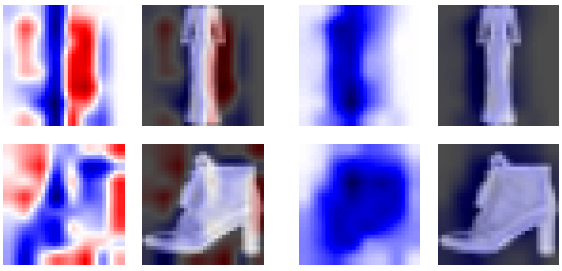
(a) $\boldsymbol{M}^{(2)}_{\text{Uniform}}$ (b) $\boldsymbol{M}^{(2)}_{\text{Decoder}}$

Fig. 3: Uniform weighting compared to decoder weighting for digit 0 (top row) and 9 (bottom row)

The uniform weighting results in a visualization map which focuses not only on the digit but also on the background. Thus, this weighting is not suited. In Fig. 3(b) the results of the decoder weighted map are shown. Compared to the uniform weighting, the encoder focuses mainly on the digit area and the background is mainly unimportant. The highlighted area of the decoder weighted map is much more smooth and consistent. This underlines the feasibility of the decoder weighting in this experiment.

B. Fashion MNIST

To further investigate Grad-LAM, we apply it to the Fashion MNIST dataset. This dataset contains more detailed structures, but also homogeneous black background. Because we trained an autoencoder with 20 latent variables, we only discuss the saliency maps $\boldsymbol{M}^{(2)}_{\text{Uniform}}$ and $\boldsymbol{M}^{(2)}_{\text{Decoder}}$.



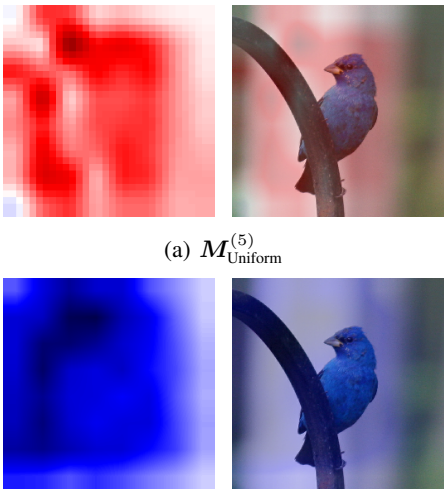
(a) $\boldsymbol{M}^{(2)}_{\text{Uniform}}$ (b) $\boldsymbol{M}^{(2)}_{\text{Decoder}}$

Fig. 4: Uniform weighting compared to decoder weighting for dress (top row) and ankle boot (bottom row)

In Fig. 4, the results of both weightings are shown for one example of the class dress and one of ankle boot. In the first row, we see that the uniform weighting focuses more on the vertical edges of the dress and on the background as well. This is not the expected result since the background contains no important information which should be encoded into the latent variables in order to reconstruct the input image. In comparison, the decoder weighted map visualizes mainly the image area of the dress itself and not the background pixels. This is expected because for the reconstruction task only the information and structure of the pixels of the dress should be relevant. The second sample shows similar results for the class ankle boot. Here, the uniform weighting highlights almost all pixels of the ankle boot and the background, whereas the decoder weighted map again highlights a continuous area in the image where the ankle boot is located. The two examples show that the weighting with the knowledge of the decoder improves the visual explanation and combines the visualization maps of the latent variables in a much better way than uniform weighting.

C. ImageNet

So far, we analyzed our proposed method on simple datasets. In our last experiment, we train the autoencoder with more layers ($M = 6$) on the whole ImageNet dataset with an image resolution of 256×256. We expect that the information of almost all pixels should be encoded into the latent variables since this is needed to reconstruct the input image. Furthermore, the model encodes the image in a much larger latent space $\boldsymbol{z} \in \mathbb{R}^{50}$.



(a) $\boldsymbol{M}^{(5)}_{\text{Uniform}}$

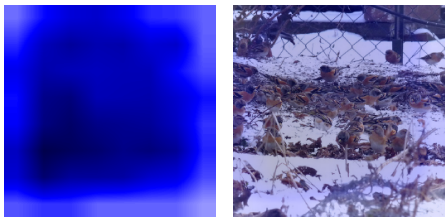
(b) $\boldsymbol{M}^{(5)}_{\text{Decoder}}$

Fig. 5: Uniform weighting compared to decoder weighting for the class indigo bunting of the ImageNet dataset

In Fig. 5 we show the visualization results for one sample of the class indigo bunting. The result of the uniform weighting in Fig. 5(a) does not highlight all image pixels and seems to focus on specific parts. Since the task of the autoencoder is reconstruction and not classification, the other pixels should be

highlighted as well. The result of the decoder weighted map in Fig. 5(b) highlights almost all pixels of the input image. Only on the right border of the saliency map, the result shows less importance on those pixels which could indicate that this information is not encoded into the latent variables.

(a) $\boldsymbol{M}^{(5)}_{\text{Uniform}}$



(b) $\boldsymbol{M}^{(5)}_{\text{Decoder}}$

Fig. 6: Uniform weighting compared to decoder weighting for the class brambling of the ImageNet dataset

Similar results can be seen in Fig. 6 for the class brambling. Here, the uniform weighting results in a bad map compared to the decoder weighting since a lot of the pixels are not highlighted. If all pixels are highlighted and therefore the complete information is encoded into the latent variables, the individual visualization maps of the latent variables could help here to further analyze the model and its encoding. Both examples of a real-world dataset show that our proposed method is suited for calculating meaningful maps for the visualization of unsupervised trained autoencoders and is also applicable to large images and deep models.

VI. Conclusion

We proposed in our work a new method for the visualization of unsupervised trained autoencoder. We first adapted the idea of Grad-CAM for the usage on the latent space of an autoencoder. In the second step we proposed two weighting techniques to generate one saliency map. The usage of the decoder knowledge for weighting the single maps resulted in interpretable visualization maps. In experiments on different datasets we showed the success of our proposed method. Grad-LAM can help to understand the highly nonlinear mapping of the image into a low-dimensional representation learned by a deep neural network in a unsupervised manner. In the future, Grad-LAM could be applied to other representation learning methods like β-VAE [7]. Furthermore, the proposed decoder-based weighting step could also be applied to other deep visualization methods, e.g. Integrated Gradients [4].

References

[1] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[2] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[3] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2018, pp. 839–847.

[4] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3319–3328.

[5] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[6] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[7] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework." *Iclr*, vol. 2, no. 5, p. 6, 2017.

[8] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[9] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 509–15 519.

[10] J. Donahue and K. Simonyan, "Large scale adversarial representation learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 10 541–10 551.

[11] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area v2," in *Advances in neural information processing systems*, 2008, pp. 873–880.

[12] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[13] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[15] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[17] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.