Grassmannian initialization: Neural network initialization using sub-space packing

Anonymous Authors[1]

Abstract

We recently observed that convolutional filters initialized *farthest apart* from each other using off-the-shelf pre-computed Grassmannian subspace packing codebooks performed surprisingly well across many datasets. Through this short paper, we'd like to disseminate some initial results in this regard in the hope that we stimulate the curiosity of the deep-learning community towards considering classical Grassmannian subspace packing results as a source of new ideas for more efficient initialization strategies.

1. Introduction

1.1. The setting

Standard initialization methods of neural networks are motivated primarily to prevent activations from vanishing or exploding.

Consider Fig 1 which focuses on the first convolutional layer of a standard CNN model for MNIST digit classification. As seen, $N = 32$ different convolutional filter weights of size 3×3, denoted by $\{\mathbf{w}_k\}_{k=1}^{N=32}$, are applied to an *image patch*, $\mathbf{x}_{ij}$, to yield the 32 channel values associated with the feature tensor extracted. The feature values computed for the k^{th} channel would be expressed as

$$\{y_{i,j,k} = f\left(\mathbf{x}_{i,j} * \mathbf{w}_k + b_k\right)\}_{k=1}^{N}; \mathbf{w}_k \in \mathbb{R}^m,$$

where $f(.)$ is a non-linear activation function, which is typically RELU or tanh, $b_k \in \mathbb{R}$ is the bias term, and in this specific case, $N = 32$ and $m = 3 \times 3 = 9$.

Initialization of these filter weights has been extensively studied in conjunction with the activation functions and state-of-the-art architectures. There now exists a plethora of resources[1] that detail the best practices to be followed with regards to the initialization strategy to be used for the specific architecture chosen.

1.2. Initialization strategies: A brief review

Most deep-learning frameworks now make available a standard repertoire of initialization strategies available for deep-learning researchers, of which the four most common ones we found available off-the-shelf in almost all frameworks were:

1. *Lecun_uniform/lecun_normal* (LeCun et al., 2012)
2. *He-uniform* (He et al., 2015)
3. *Glorot/Xavier-uniform* (Glorot & Bengio, 2010)
4. *Orthogonal* (Saxe et al., 2013)

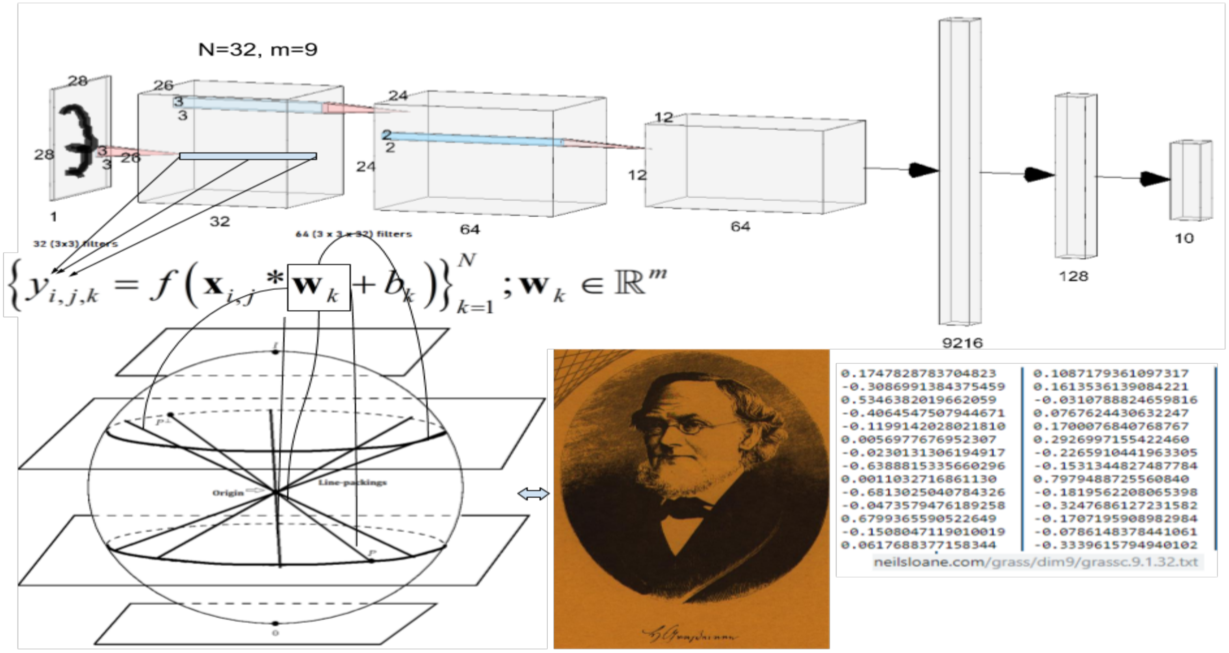
We'd posit that it's now part of machine learning folklore(Katanforoosh & Kunin, 2018) that a practitioner would feault to the choice of `Xavier initialization` if the activation function of the layer is $tanh()$ and `He-initialize` if the activation function is $RELU()$. It is to be noted that for specific architectures such as deep-residual networks, recent works have questioned on the efficacy of the above initialization strategies. In (Yang & Schoenholz, 2017) , the authors critique the dependence of the optimal init-variances on the depth of the network and propose a novel mean field residual networks framework. This 'random initialization on the edge of chaos' idea reoccurs in (Hayou et al., 2018) where they also re-emphasize the supposed efficacy of using *swish* activation functions over RELU-like functions. In the context of Binary neural networks, the authors in (Clark et al., 2017) has showcased the efficacy of *Hadamard initialization* over the other techniques. We note in passing that the default initializations for specific layers and activation-types vary from one deep learning framework to the other, and is often a point of much debate among the practitioners[2].

[1] Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1] http://www.deeplearning.ai/ai-notes/initialization/

[2] https://twitter.com/jeremyphoward/status/1113477414628106240, https://



$$\left\{y_{i,j,k} = f\left(\mathbf{x}_{i,j} * \boxed{\mathbf{w}_k} + b_k\right)\right\}_{k=1}^{N}; \mathbf{w}_k \in \mathbb{R}^m$$

```
0.1747828783704823     0.1087179361097317
-0.3086991384375459    0.1613536139084221
0.5346382019662059     -0.0310788824659816
-0.4064547507944671    0.0767624430632247
-0.1199142028021810    0.1700076840768767
0.0056977676952307     0.2926997155422460
-0.0230131306194917    -0.2265910441963305
-0.6388815335660296    -0.1531344827487784
0.0011032716861130     0.7979488725560840
-0.6813025040784326    -0.1819562208065398
-0.0473579476189258    -0.3247686127231582
0.6799365590522649     -0.1707195908982984
-0.1508047119010019    -0.0786148378441061
0.0617688377158344     -0.3339615794940102
```

neilsloane.com/grass/dim9/grassc.9.1.32.txt

Figure 1. Outline of the Grassmannian initialization idea

The continued efforts of researchers in this specific subfield of deep-learning highlights that much work needs to be done before unanimity can be achieved as to which is the optimal initialization strategy for what combination of architecture/activation function.

The initial set of observations that we'd like to disseminate through this paper sits squarely into this setting of finding that elusive optimal initialization strategy. We build on the **Grassmannian subspace-packing** body of work (Conway et al., 1996) in experimental mathematics that has also been successfully used in physical layer wireless communications (See (Love & Heath, 2005),(Prabhu et al., 2009)) for limited feedback codebook-based downlink beamforming schemes and sparse signal reconstruction (Malioutov et al., 2005).

1.3. Presenting the hunch: Tackling dead filters and achieve distinctive learning

The reason why we developed a hunch for trying out this technique is as follows. Getting the convolution filters to learn distinctive attributes so that we don't end up with a scenario where the learned filters post-training all *look the same* has an interesting and chequred history. Back in 2014, in the highly cited work on *Visualizing and Understanding Convolutional Networks* (Zeiler & Fergus, 2014), the authors propose a set of best-practices for improving upon *Alexnet* such as choosing a different kernel size, stride-length and use of feature-scale clipping. They focus on a set of visualizations that demonstrate how the learned features look like before and after applying their set of techniques. With regards to the feature scale clipping idea, they highlight how this prevents 'one feature (sic) Kernel' from dominating. They also showcase how the smaller stride and filter-sizes resulted in more "*distinctive features and fewer dead features*". This theme of wanting to ensure that each filter learns something different is rather intuitive as this alludes towards more efficient usage of the computational real estate that the conv-nets dispense at the classification problem. Our ansatz is that an intuitive way of achieving this inter-filter diversity is to ensure that they are initialized furthest apart upon initialization. Given that these filters reside in $\mathbb{R}^m$, this nows becomes the line-packing (and in general subspace packing) problem. Here we'd like to inherit the classical Neil Sloane example of the line-packing problem seeking to answer the question:

> *"How should N laser beams passing through a single point*
> *be arranged so as to make the angle between*
> *any two of the beams as large as possible?"*

`stackoverflow.com/questions/49433936/how-to-initialize-weights-in-pytorch/49433937`

2. Background on Grassmannian manifolds and subspace packing

Set theoretic definition: The real Grassmann manifold $G(m,k)$ is define as the set of all k-dimensional (linear) subspaces in $\mathbb{R}^m$ and the Grassmannian N-subspace packing problem is the problem of finding a set of N k-dimensional subspaces in $G(m,k)$ that maximize the minimum pairwise distance between the constituent subspaces in the set.

In this paper, we consider the special case of $k = 1$ (also termed as the line-packing scenario). Let Ω_m denote the set of unit vectors in $\mathbb{R}^m$. As shown in (Love & Heath, 2005), for a given (N, m), arranging N unit vectors, $\mathbf{w}_i \in \Omega_m$ such that the magnitude correlation between any two vectors is as small as possible yields the line-packings with regards to the *sine-distance* metric defined to be:

$$d(\mathbf{w}_1, \mathbf{w}_2) = sin(\theta_{1,2}) = \sqrt{1 - |\mathbf{w}_1^T \mathbf{w}_2|^2}.$$

The final packing is represented by a *codebook matrix*, $\mathbf{W} = [\mathbf{w}_1|\mathbf{w}_2|...|\mathbf{w}_1, \mathbf{w}_N]; \mathbf{w}_i \in \Omega_m$, characterized by the the minimum distance of packing $\delta(\mathbf{W})$, which is defined as,

$$\delta(\mathbf{W}) = \min_{iklN} \left\{ \sqrt{1 - \left|\mathbf{w}_k^T \mathbf{w}_l\right|^2} \right\} = \sin\left(\theta_{\min}\right).$$

The Rankin bound (Barg & Nogin, 2002) provides the upper bound for this minimum distance and is given by,

$$\delta(\mathbf{W}) \leq \sqrt{\frac{(m-1)N}{m(N-1)}}$$

A normalized invariant measure μ introduced on $\mathcal{G}(m,1)$ by the normalized Haar measure on Ω_m allows computation of volumes in $\mathcal{G}(m,1)$, which is in turn used to define the density of a given line-packing matrix $\mathbf{W}$. It was shown in (Love & Heath, 2005) that,

$$\Delta(\mathbf{W}) = N\left(\frac{\delta(\mathbf{W})}{2}\right)^{2(m-1)}.$$

There exists pre-computed repositories for the best known packings for both complex [3] and real scenarios. [4]

2.1. What if we do not have a packing available in Sloane's repository for the tuple: $(m, 1, N)$ that we desire to incorporate into our architecture?

If the mismatch is with regards to N, we suggest finding the largest N' such that the tuple $(m, 1, N')$ exists in the repository.

2.2. Shortcomings

Akin to orthogonal matrix initializations (Saxe et al., 2013), which requires square matrices, we are somewhat limited by the choice of the tuple: $(m, 1, N)$. The repository from where we sourced the codebooks is limited up to $(m = 16, 1, N = 45)$ (Sloane, 2004). One path ahead is to a priori construct packings in Grassmannian manifolds via the alternating projection method described in (Dhillon et al., 2008). In this paper, we explore only those architectures whose filter-sizes (m) and the number of filters (N)

3. Experiments

3.1. Grassmannians in First Filter Initializations

With the intuition that the first few convolutional kernel filters should capture as much diverse features as possible to prevent 'dead kernels' (Zeiler & Fergus, 2014), we experimented with both shallow CNNs with 2 to 4 convolutional layers, as well as standard ResNet-56 models (He et al., 2016) on different datasets.

3.2. Shallow CNNs

As our baseline, we have 2-layer CNNs and 4-layer CNNs as per 1 with standard default Xavier initialization. In our experiments, we only change the first layer of convolutional filter kernels, as we wish to capture diverse representations and features from the input image, and allow the rest of the network to learn the different combination and activation of the diverse first-layer features.

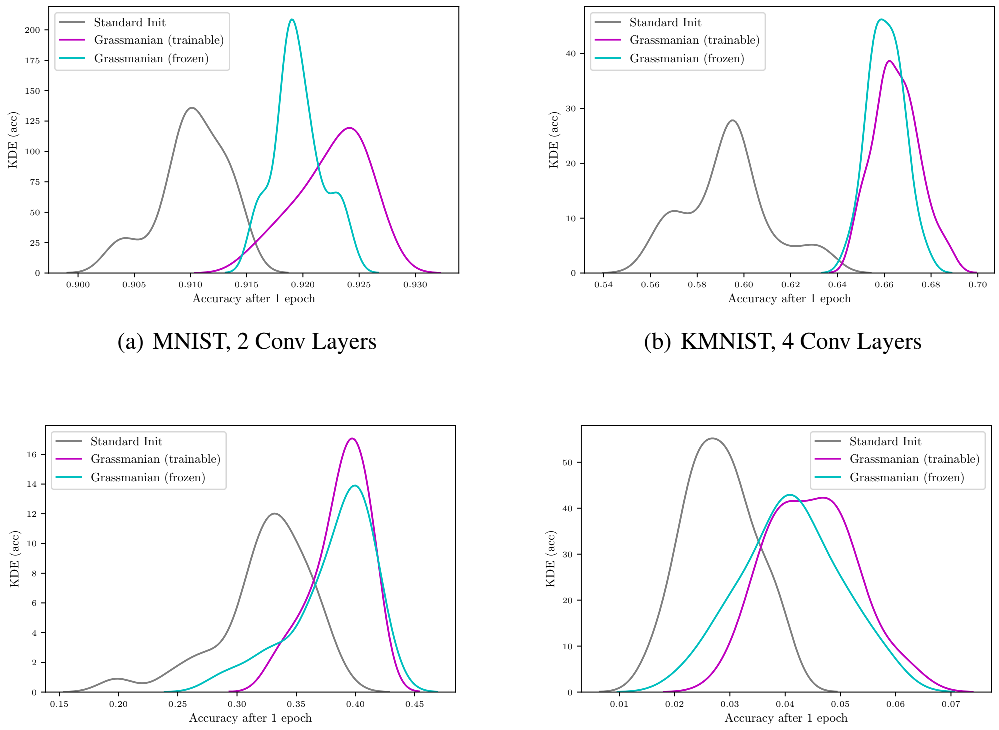
In comparison with the baselines, where we introduce Grassmannian initializations without biases as trainable and untrainable (fixed) parameters. For single-channel inputs such as MNIST, KMNIST and Fashion MNIST and assuming that the first convolutional kernels are of size $3 \times 3 = 9$, we use line packings of $(m, 1, N)$ where $m = 9$ and N is the number of output channels. This is equivalent to finding the best way of 'stabbing' N lines through a 9-D sphere such that the minimum distance between each line is maximized.

3.3. Deep CNNs

We also ran similar experiments on ResNets, where we use a standard ResNet-56 with standard Xavier initialization and Adam optimizer as baseline. Given images with 3 input channels and assuming that the first convolutional kernel is of size $3 \times 3 = 9$, we initialize the weights of the first convolutional kernel to be Grassmannian line packings of $(m, 1, N')$ where $m = 9$ is the kernel size squared, with N as the number of output channels and $N' = 3N$ since we have input channels of size 3. We then initialize the first layer using this line packing without biases, and train it under both fixed and trainable conditions.

[3] https://engineering.purdue.edu/~djlove/grass.html

[4] http://neilsloane.com.grass



(a) MNIST, 2 Conv Layers

(b) KMNIST, 4 Conv Layers

(c) CIFAR-10, 4 Conv Layers

(d) CIFAR-100, 4 Conv Layers

Figure 2. Distributions of 30 runs of first-epoch test accuracy at the first epoch with SGD with different datasets, comparing initialization of first layer using standard Xavier initialization, frozen Grassmannians, and trainable Grassmannians.

4. Results

We ran multiple trials on different datasets using with our approaches, and verified that in our runs we achieve better test accuracies on different datasets on shallow architectures, where both fixed and trainable Grassmannian first-layer initializaitons almost consistently achieve higher first-epoch accuracies.

The optimizer used also has a significant impact on the first-epoch test accuracy of initializations. While Adam and Adadelta with standard initialization outperforms frozen Grassmannian initialization in first-epoch accuracies, trainable Grassmannians still outperforms standard initializations in both cases. The improvement gained from Grassmannians are most pronounced with SGD as an optimizer.

Table 1. Final Accuracy of ResNet-56 on CIFAR-10

INITIALIZATION	TEST ACC
STANDARD, XAVIER	91.85
GRASSMANNIAN, FIXED	91.72
GRASSMANNIAN, TRAINABLE	**92.18**

For deeper networks such as ResNets, we see faster convergence even with Adam optimizer as per 4, and achieves slight improvement in accuracies on the final test-set classification score after training for 200 epochs. We also attach our code here for reproducibility purposes, while providing a framework for extracting Grassmannians. [5]

References

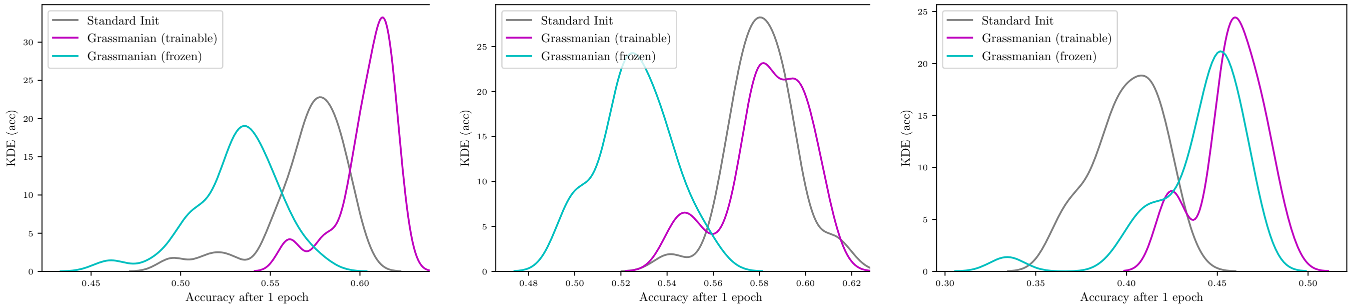
Barg, A. and Nogin, D. Y. Bounds on packings of spheres in the grassmann manifold. *IEEE Transactions on Information Theory*, 48(9):2450–2454, 2002.

Clark, A., Prabhu, V. U., and J, W. Weight initialization strategies for binary neural networks. *TinyML workshop, ICML*, 2017.

Conway, J. H., Hardin, R. H., and Sloane, N. J. Packing lines, planes, etc.: Packings in grassmannian spaces. *Experimental mathematics*, 5(2):139–159, 1996.

Dhillon, I. S., Heath, J. R., Strohmer, T., and Tropp, J. A. Constructing packings in grassmannian manifolds via

[5] https://anonymous.4open.science/r/98100837-225e-4765-8e45-0d11179c497b/



(a) Adadelta

(b) Adam

(c) SGD

Figure 3. Distributions of 30 runs of first-epoch test accuracy at the first epoch ***with different optimizers*** on the same shallow CNN architecture, comparing initialization of first layer using standard Xavier initialization, frozen Grassmannians, and trainable Grassmannians.

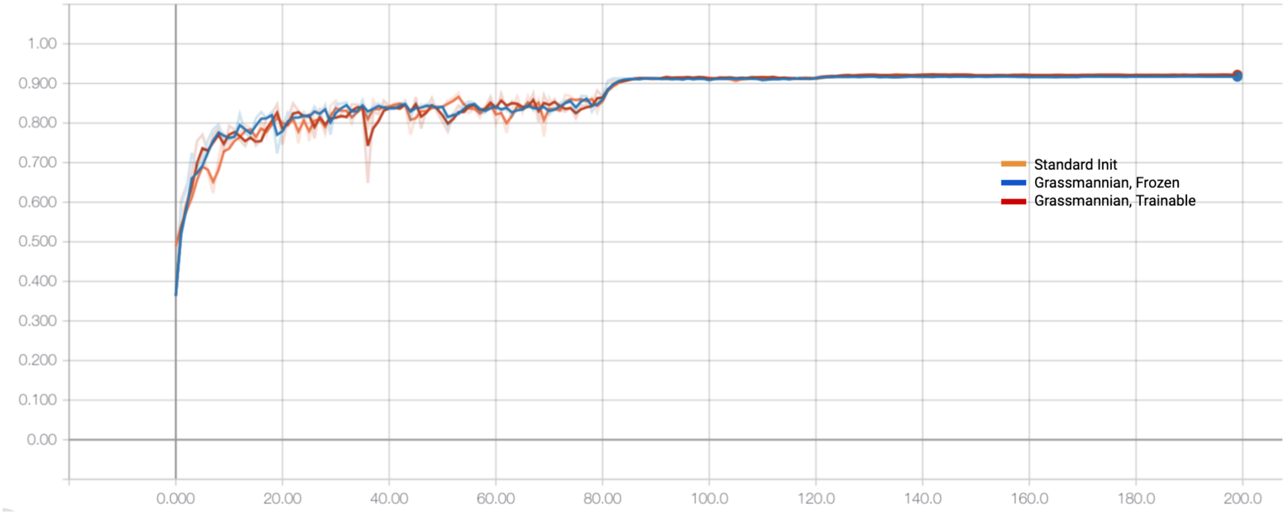


Figure 4. Comparison of standard initialization and Grassmannian initialization of first layer as both trainable and untrainable parameters on ResNet trained on CIFAR-10. Grassmannian approaches have a faster convergence with marginally better test accuracy with Adam optimizer used in all 3 cases.

alternating projection. *Experimental mathematics*, 17(1): 9–35, 2008.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Hayou, S., Doucet, A., and Rousseau, J. On the selection of initialization and activation function for deep neural networks. *arXiv preprint arXiv:1805.08266*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Katanforoosh and Kunin. Initializing neural networks. *deeplearning.ai*, 2018.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Love, D. J. and Heath, R. W. Limited feedback unitary precoding for spatial multiplexing systems. *IEEE Transactions on Information theory*, 51(8):2967–2976, 2005.

Malioutov, D., Cetin, M., and Willsky, A. S. A sparse signal reconstruction perspective for source localization with sensor arrays. *IEEE transactions on signal processing*, 53(8):3010–3022, 2005.

Prabhu, V. U., Karachontzitis, S., and Toumpakaris, D. Performance comparison of limited feedback codebook-based downlink beamforming schemes for distributed antenna systems. In *2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology*, pp. 171–176. IEEE, 2009.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Sloane, N. J. A. Table of best grassmannian packings. *In collaboration with A. R. Calderbank, J. H. Conway, R. H. Hardin, E. M. Rains, P. W. Shor and others. Published electronically*, 2004.

Yang, G. and Schoenholz, S. Mean field residual networks: On the edge of chaos. In *Advances in neural information processing systems*, pp. 7103–7114, 2017.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.