Proseminar - *Convolutional Neural Network*

Convolutional Neural Network

Author

Simon Schupp

Matrikelnummer: *2412882*

Date:

Karlsruhe, July 24, 2023

Abstract

In recent years, significant advancements have been made in the field of Machine Learning, particularly in the domain of image-related tasks. This proseminar aims to provide a comprehensive overview of Convolutional Neural Networks (CNNs), a powerful class of neural networks widely employed for image analysis and processing. In addition to that, this paper presents a CNN model that is trained from scratch, specifically for digit classification. The model achieves solid performance and can accurately classify hand-drawn digits. All codes are available at: https://github.com/LuposX/SeminareCNN

Contents

1 Introduction

A study revealed that over 24,000 GB of data is uploaded to the internet every second, from which approximately 80% consists of video data (CISCO, 2022). Consequently, a vast amount of visual data is generated each second, highlighting the significance of algorithms that are capable of comprehending visual information.

Algorithms that can interpret visual data have various applications, including autonomous vehicles (Rao & Frtunikj, 2018) that rely on cameras to perceive the environment, analyze it, and make informed decisions based on the gathered data. Additionally, these algorithms are instrumental in fields such as skin cancer detection (Gouda, Sama, Al-Waakid, Humayun, & Jhanjhi, 2022), crop disease detection (Boulent, Foucher, Théau, & St-Charles, 2019) and more. Therefore, the development of algorithms that possess the ability to understand visual data is important.

In computer systems, an image is represented as a matrix, where the size of the matrix corresponds to the dimensions of the image. This matrix representation poses several challenges for algorithms aiming to understand images, including viewpoint variations, illumination changes, occlusions, background clutter, and more. For instance, when viewing an image from different perspectives or under different lighting conditions, the semantic meaning of the image remains the same, but the values within the matrix can change drastically.

This brings us to the fundamental question: How can we develop algorithms that can effectively understand images despite the inherent variability in their representations? One idea is instead of using rigid rules and handcrafted features, we let the computer learn the semantic meaning of images.

2 Convolutional Neural Network

2.1 Introduction

Neural Network can serve as universal function approximator (Hornik, Stinchcombe, & White, 1989). This property makes them a suitable candidate for training algorithms that can semantically understand images. However, as highlighted in (Lecun, Bottou, Bengio, & Haffner, 1998), conventional neural networks are not optimally designed for learning visual data due to the following reasons:

- A fully connected Neural Network would take too many parameters, thus being computationally expensive to train.
- A fully connected Neural Network has no built-in invariance to image transformations like already named in subsection 1.
- Fully connected Neural Networks(FCNN) ignore how the input is ordered. We could randomly map (fixed) input pixels to the first layer without affecting the training.

A more effective approach is to utilize specialized architectures known as convolutional neural networks (CNNs), like introduced in (Rumelhart, Hinton, & Williams, 1985), (LeCun et al., 1989), (Lecun et al., 1998) and (Krizhevsky, Sutskever, & Hinton, 2012). CNNs address the limitations faced by traditional neural networks and provide improved performance in handling visual data.

CNNs can learn to semantically understand images, instead of relying on handcrafted features or explicit rules. CNNs do that by learning a hierarchical representation from the raw pixel data, as seen in figure 1.

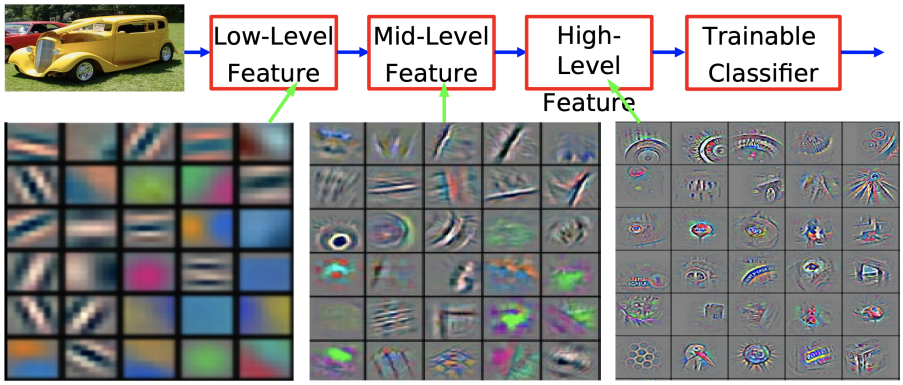


Figure 1: Feature visualization of convolutional net trained on ImageNet from (Zeiler & Fergus, 2014). Images to the left are features extracted in earlier layers, while images to the right are features extracted in the later layers. We can see earlier layer look for more simple patterns while the later layers look for more complex pattern. Source: `https://atcold.github.io/pytorch-Deep-Learning/en/week03/03-1/`

2.2 Convolutional Neural Network

Convolutional neural networks (CNNs) combine several key ideas:

1. Local Receptive Fields: Each neuron in the Neural Network receives input from neighboring neurons in a small local region of the previous layer.

2. Weights Sharing: Neurons within the same layer share the same set of weights, as opposed to having individual weights for each neuron.

3. Spatial Subsampling: Subsampling is performed based on region, reducing the dimension of the neural network.

The underlying concept of CNNs is inspired by the functioning of neurons in the visual cortex of animals and humans. This idea dates back to a paper by Hubel and Wiesel (Hubel & Wiesel, 1962). Where they demonstrated that neurons in the brain of a cat are hierarchically organized and possess local

receptive fields. These local receptive fields enable the detection of elementary patterns such as edges, vertices, corners, and more.

To implement this concept, weight sharing is employed. Neurons within the same layer share an identical set of weights, leading to a reduction in the number of parameters in the neural network. This set of weights is also called a filter/kernel. See figure 2 for an example.

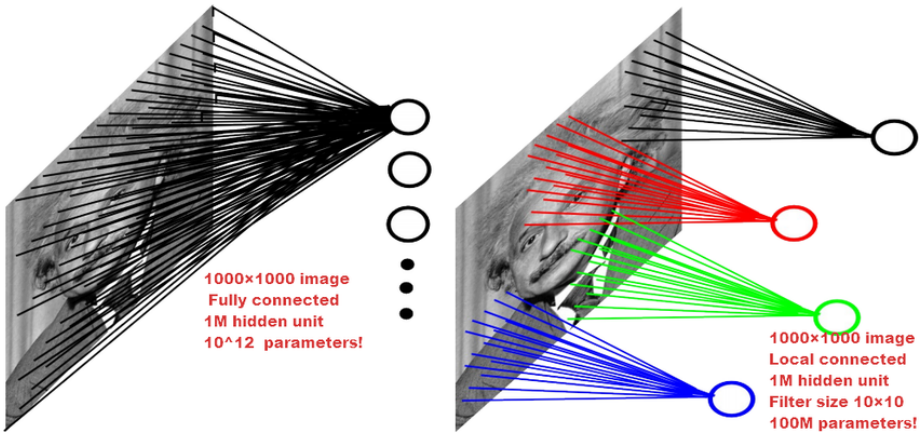


Figure 2: Comparison between a fully connected layer (left), every pixel in the image is connected to every neuron, and a convolutional layer (right), where the neurons are only locally connected. Source: (Qiu, 2016)

This not only makes training and usage of the network more computationally efficient but also ensures that the network attends to the same patterns across the entire image. These elementary patterns, identified by the kernel, are also referred to as features and are crucial building blocks for the semantic understanding of the image.

Subsequent layers in the network combine these elementary patterns to form more complex features. See figure 1 for an example. A collection of neurons within a layer is commonly referred to as a feature-/activation-map, where all neurons share the same set of weights. This design constraint enables the network to perform the same convolutional operation across different parts of the image.

Within a single convolutional layer, multiple filters/kernels can be employed, each focusing on different features within the image. For instance, we could have one filter which focuses on detecting edges, while another filter identifies corners.

2.3 Structure of CNN

There are two essential types of layers in a CNN: convolutional layer and pooling layers. Additional CNNs may also include fully connected layers, but they are not the core component of the CNN. In the following sections, I want to provide an explanation of the core elements of a CNN.

2.4 Convolutional Layer

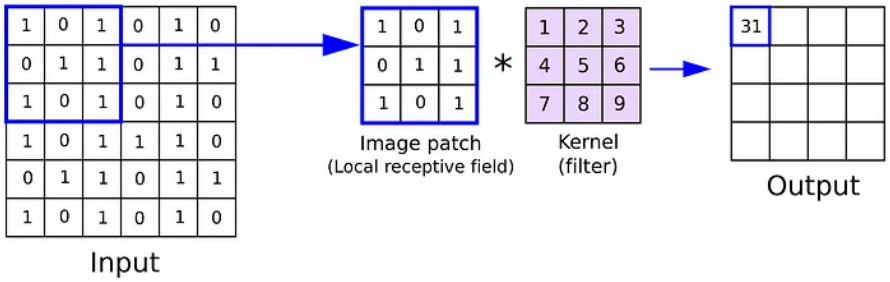


Figure 3: Example of a convolution operation. Source: `https://anhreynolds.com/blogs/cnn.html`

The convolutional layer is an important building block within the CNN Architecture. It uses weight sharing to drastically reduce the number of parameters that need to be learned by the network.

During the forward pass, the convolutional layer applies each filter/kernel by convolving it across the input, resulting in the generation of a two-dimensional activation map.

The concept of convolution arises from the *convolutional operation.* [1]

The convolution operation S, with input I and kernel K with size, $(2N + 1) \times (2N + 1)$[2] is defined as:

$$S(x, y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} I(x + i, y + j)K(x, y)$$

Essentially, the convolutional operation can be understood as sliding a kernel across the entire image matrix and computing the dot product between a patch from the input matrix and the kernel.

To better understand this operation, refer to figure 3 for a graphic depiction of the operation or to this for a video representation.

Two important hyperparameters are associated with the convolution operation:

- The first is stride, which determines the number of pixels the kernel slides by during each operation. For example, a stride of 1 means that the sliding window moves 1 pixel at a time.

[1] In practice *cross-correlation* is used instead, the terms *convolutional operation* and *cross-correlation* are often used interchangeably, although they are slightly different.

[2] The size of the kernel doesn't need to be odd, but it is convenient, so we can suppose that as it shifts, its center is right on top of an element of image I.

- The second hyperparameter is the kernel size, which determines the dimensions of the kernel sliding over the input image. For instance, a kernel size of 4×4 indicates that we have a kernel matrix of size 4×4 that slides over the input matrix.

The kernel size can also be understood as the size of the local receptive field of the neurons.

2.5 Subsampling

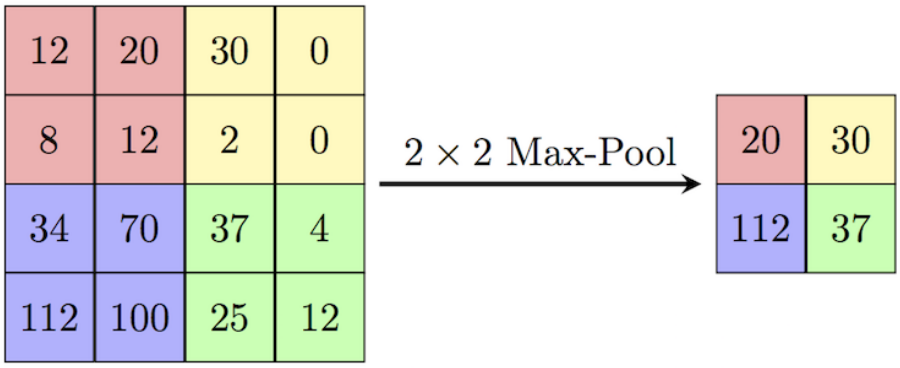


Figure 4: Demonstration of the max-pooling operation with stride 2 and filter size 2x2. We go over each image patch and take the highest value. Source: `https://computersciencewiki.org/index.php/File:MaxpoolSample2.png`

Training an infinitely large neural network is not feasible due to the increasing computational costs associated with both training and inference. Therefore, it is essential to ensure that a network is not unnecessarily larger than necessary. One way to achieve this is by reducing the dimensions of the network layers, where the dimension means the number of neurons in a layer.

To accomplish dimension reduction, subsampling techniques can be employed. Two popular subsampling methods are average pooling and max pooling.

In average pooling, similar to convolution layers, we slide over our image matrix and take the average of a neighborhood region in the image and map it to a single neuron in the next layer. On the other hand, in max pooling, the highest value within a neighborhood region is selected and mapped to a neuron in the subsequent layer. See figure 4 for an example of max pooling.

The max-pooling operation can be seen as propagating only the most significant signal that excites a neuron the most while discarding the rest. This approach bears some resemblance to the way the brain processes information. What type of pooling layer is better is highly dependent on the task (Zafar et al., 2022).

Both of these approaches aid in reducing the dimension of the network.

3 Self-Trained CNN

In order to demonstrate the effectiveness of a CNN, I conducted an experiment where I trained a CNN from scratch to recognize black-and-white digits from images.

The code I used to train and evaluate the models is available at `https://github.com/LuposX/SeminareCNN/tree/main`.

3.1 Dataset

I trained my CNN using the MNIST dataset (Lecun et al., 1998), the MNIST dataset is a collection of 70,000 hand-drawn digits. These digits are gray-scaled images and have been normalized to fit within a 28×28 pixel bounding box.

I choose this dataset due to its popularity within the machine learning community and its low computational requirements.

Moreover, the dataset was commonly used for evaluating the performance of CNNs. The MNIST dataset is divided into 60,000 training images and 10,000 testing images. To prepare the image for training, I normalize them to ensure that the input data has a mean of 0 and a standard deviation of 1, which can improve the stability of the learning process. (Sola & Sevilla, 1997).

3.2 Architecture

The CNN architecture, as illustrated in Figure 5, consists of 6 layers with learnable weights. The initial 4 layers are convolutional, while the subsequent 2 layers are fully connected.

The first convolutional layer takes a 28×28×1 [3] image as input and applies 16 filters to extract relevant features. Subsequent convolutional layers have the following number of filters: 32, 64, and 128.

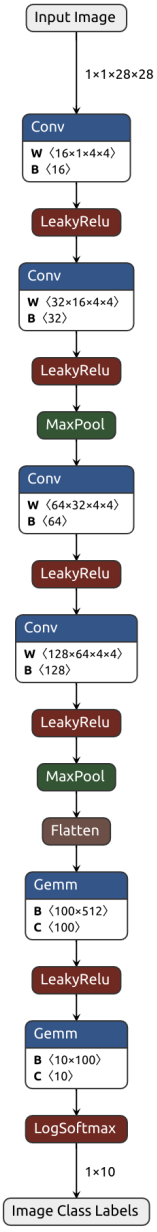


Figure 5: The architecture of the Baseline CNN-Model.

[3] This refers to (Image Width × Image Height × Number of Color Channels).

To reduce the spatial dimension of the feature maps, a max pooling layer is applied every two convolutional layers. This layer uses a kernel size of 2x2 with a stride of 2, effectively halving the spatial dimension of the image.

Transitioning to the fully connected layers, the network contains 5776, 200, and 10 neurons, respectively. As activation-function, Leaky ReLU is applied to all layers except the last one. Leaky Relu has the advantage over ReLu that it leads to a slightly more stable learning process [4] (Maas, 2013). Formally, the Leaky ReLu function is defined as:

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ \alpha \cdot x & \text{otherwise.} \end{cases}$$

where $\alpha = 0.2$ this value was determined to be effective based on findings from the paper (Xu, Wang, Chen, & Li, 2015).

The final layer's output is passed through a 10-way log-softmax function, generating a probability distribution across 10 class labels. Formally, the log-softmax function is defined as:

$$\text{LogSoftmax}(x_i) = log(\frac{\exp(x_i)}{\sum_{j \in \{0,1,\ldots,9\}} \exp(x_j)})$$

Applying log-softmax in combination with *negative log-likelihood loss* has several advantages, its primary benefit being its numerical properties that contribute to faster training time.

We can interpret the output of the neurons as the model's confidence in each class label. A high output value associated with a specific label indicated that the model thinks that label is more probable to be the correct output.

3.3 Trainings-Details

The model was trained using the backpropagation algorithm and the *negative log likelihood* [5] loss function, which is commonly used for multi-class classification tasks.

For efficient training, Google Colab was used, taking advantage of the free dedicated GPU to accelerate the training process.

The CNN was trained for 5 epochs using a batch size of 1024. The total training time was approximately 5 minutes.

As training optimizer, Adam was used with a learning rate of 0.001. Adam is a state-of-the-art optimizer known for its ability to adaptively adjust learning rates during training and the use of momentum (Kingma & Ba, 2014).

3.4 Results & Ablation Study

To assess the robustness of Convolutional Neural Network(CNN) compared to fully connected neural networks(FCNN), I experimented with both architectures. In addition to the CNN-Model, I trained an additional FCNN once on

[4] It solves the problem of dead-ReLus.

[5] Is defined as $L(X, Y) := -log(p(Y|X))$

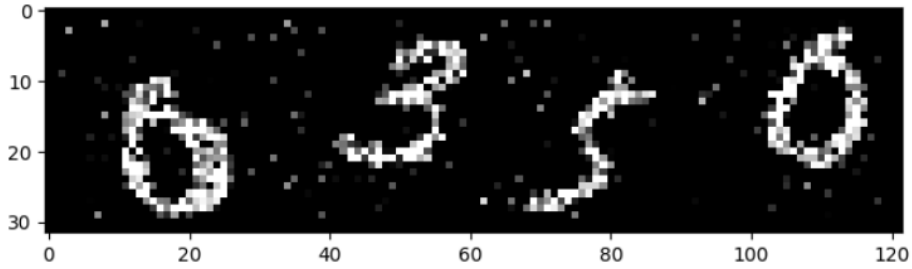


Figure 6: Sample Data from the modified MNIST dataset.

the original dataset and once on a modified dataset, on which I applied various transformations. The transformation applied to the dataset were as follows:

- Adding Random Noise: I generated random tensors of the same size as the input images. These tensors were sampled from a Gaussian Distribution N [6], I added these sampled tensors to the corresponding images.
- Random Image Cropping: All images in the dataset were randomly cropped, resulting in images that were no longer centered.

A sample from the modified dataset can be seen in figure 6.

After performing these transformations, I trained a new CNN model (referred to as *CNN_Noise_RandomCrop*) with the same architecture as the original model, using the transformed dataset. Additionally, I trained an FCNN of equivalent size [7] as the CNN model. The FCNN was trained twice, once on the original dataset (referred to as *FC_NN*) and once on the transformed dataset (referred to as *FC_NN_Noise_RandomCrop*).

As depicted in figure 7 the CNN trained on the original dataset achieved a performance of approximately 99%. However, when I trained on the transformed dataset, the CNN's accuracy decreased to 92%. On the other hand, the FCNN achieved an accuracy of 95% when trained on the original dataset. When trained on the transformed dataset, the FCNN's accuracy dropped significantly to 75%.

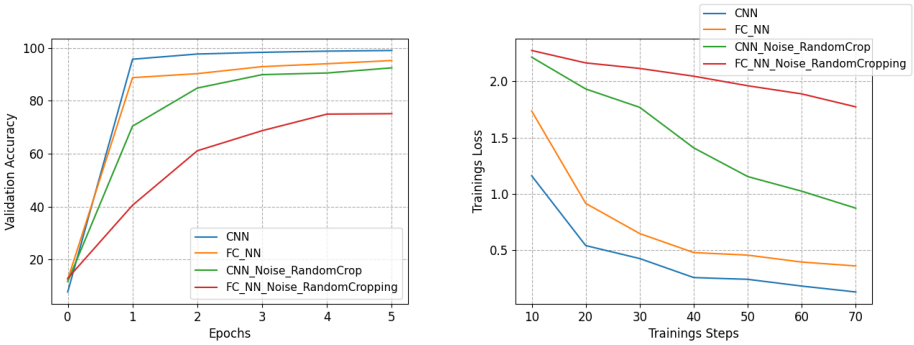
These outcomes demonstrate the robustness of CNNs, as they exhibit greater resilience to image transformations, as explained in Section 2.

4 Conclusion & Discussion

This proseminar provides an overview of CNNs, including their definition and functionality. Additionally, a practical demonstration was conducted aimed to show the capabilities of CNNs and illustrate how a CNN architecture might

[6] $N \sim \mathcal{N}(0.6, 0)$, the standard deviation in this case can be interpreted as the intensivity of the noise.

[7] Size refers here to the parameter count, for more information about the architecture of the FCNN model check the projects GitHub page.



(a) The Validation Accuracy

(b) The trainings Loss

Figure 7: Following abbreviations apply CNN = Convolutional Neural Network, FC_NN = Fully Connected Neural Network, Noise = Random Noise was added do the training's data, RandomCrop = the training data was randomly cropped.

look like. This serves as an example to showcase the effectiveness of CNNs and provide insight into their architectural structure.

It would have been interesting to investigate the possibility to make the FCNN more competitive with the CNN through the use of additional techniques.

5 Acknowledgment

I would like to thank Yuri Koide for giving me pointers while I was working on the slides to present the topic of this proseminar.

References

Boulent, J., Foucher, S., Théau, J., & St-Charles, P.-L. (2019). Convolutional neural networks for the automatic identification of plant diseases. *Frontiers in Plant Science*, *10*. Retrieved from https://www.frontiersin.org/articles/10.3389/fpls.2019.00941 doi: 10.3389/fpls.2019.00941

CISCO. (2022). *Cisco annual internet report.* Retrieved 2022-06-02, from https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html

Gouda, W., Sama, N. U., Al-Waakid, G., Humayun, M., & Jhanjhi, N. Z. (2022, June). Detection of skin cancer based on skin lesion images using deep learning. *Healthcare (Basel)*, *10*(7), 1183.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359-366. Retrieved from https://www.sciencedirect.com/science/article/pii/0893608089900208 doi: https://doi.org/10.1016/0893-6080(89)90020-8

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, *160*(1), 106.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541-551. doi: 10.1162/neco.1989.1.4.541

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. doi: 10.1109/5.726791

Maas, A. L. (2013). Rectifier nonlinearities improve neural network acoustic models..

Qiu, J. (2016). *Convolutional neural network based age estimation from facial image and depth prediction from single image* (Unpublished doctoral dissertation).

Rao, Q., & Frtunikj, J. (2018). Deep learning for self-driving cars: Chances and challenges. In *Proceedings of the 1st international workshop on software engineering for ai in autonomous systems* (p. 35–38). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3194085.3194087 doi: 10.1145/3194085.3194087

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation* (Tech. Rep.). California Univ San

Diego La Jolla Inst for Cognitive Science.

Sola, J., & Sevilla, J. (1997). Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, *44*(3), 1464-1468. doi: 10.1109/23.589532

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., ... Almotairi, S. (2022). A comparison of pooling methods for convolutional neural networks. *Applied Sciences*, *12*(17). Retrieved from `https://www.mdpi.com/2076-3417/12/17/8643` doi: 10.3390/app12178643

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer vision–eccv 2014: 13th european conference, zurich, switzerland, september 6-12, 2014, proceedings, part i 13* (pp. 818–833).