Deep Convolutional Features for Image Classification

Dharmesh Tailor
School of Informatics, University of Edinburgh

1 Introducton

Convolutional neural networks (ConvNets), in the form proposed by LeCun in 1989 [1], consist of a stack of convolutional, pooling and fully-connected layers arranged such that computation is propagated in a feedforward fashion. In the 1990s, they were successfully applied to the recognition of handwritten zip code digits used by the U.S. Postal Service. Today, a ConvNet remains the benchmark for handwritten digit recognition [2]. Despite early success, ConvNets had little application to other tasks. Multi-layer neural networks were slow to train and gradient-based optimisation would often get stuck in poor local minima.

Interest in ConvNets revived when 'AlexNet' [3], a deep convolutional neural network, achieved a top-5[1] error rate of 15.4% in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 [4]. The next best model achieved an error rate of 26.2%. This corresponds to a reduction of approximately 40%; a significant achievement that led to a revival in research in neural networks. To demonstrate the prominence of ConvNets today, all subsequent winners of the ILSVRC have been variations of AlexNet's original architecture and have attained lower error rates each time.

One of ImageNet's tasks is the classification of images containing objects in realistic settings. In comparison to handwritten text, natural images contain far greater variability e.g. scale, pose, lighting of objects. This complex structure is better exploited in ConvNets whose architecture allows for the learning of feature hierarchies. However, to ensure good generalisation, large training sets with millions of labelled instances are required. The ImageNet dataset, first released in 2010, contains over a million labelled, high-resolution images. This is one of the factors that have allowed for the training of deep architectures such as ConvNets. Other factors include the emergence of ConvNet implementations for graphics processing units (GPUs) which have significantly reduced training time through

[1]Fraction of test samples where the target label was not one of the top 5 most confident classes predicted by the model

the parallelisation of matrix-vector operations. In addition, regularisation strategies such as DropOut [5] have helped to reduce the effects of overfitting.

The high-level architecture of traditional pattern recognition systems consists of two components: a feature extractor followed by a classifier. The classifier would be trained, using some learning algorithm, on vectors generated by the feature extractor on the training set images. For example, on the problem of handwritten digit recognition, the feature extractor in [6] uses four 3x3 Kirsch masks for the purpose of edge detection. The masks are convolved over the input pixels generating four feature maps. These are consequently passed to a standard multi-layer perceptron (MLP) for classification. This demonstrates that the purpose of a feature extractor is to exploit *a priori* knowledge about the recognition task. In this case, the Kirsch mask feature extractor would be insufficient for object recognition in natural images.

The resurgence of ConvNets is part of a wider trend where *a priori* knowledge is used to constrain the general architecture of the classifier. ConvNets are directly trained on raw pixels of the input image; feature extraction has become a learning process. The architecture of ConvNets would suggest a hierarchy of features are learned but, until recently, this hypothesis has only been assumed. We will be addressing the following questions: what features are learned and how are they represented, and more fundamentally, what is the internal operation of a ConvNet. In doing so, this paper will evaluate experiments into the functioning of intermediate layers and their generalisation ability to related tasks. Furthermore, we will review techniques in feature visualisation as well as how they can be used to design better ConvNet architectures.

2 Background

2.1 ConvNet Architecture

A convolutional neural network, first proposed in [1], was inspired by the 'Neocognitron' [7] - a neural network resembling the hierarchical arrangement of simple to complex cells in Hubel and Wiesel's model of the mammalian visual cortex [8]. ConvNets, like other multi-layer neural networks, are trained using back-propagation via gradient descent [9]. A ConvNet differs from a standard MLP in that it contains a mixture of convolutional and pooling layers. These components make ConvNets well-suited to data with strong local correlations i.e. nearby pixels in an image (spatial) or sequential values in an audio signal (temporal). In an MLP, the input, output and intermediate representations between layers are processed as a one-dimensional array of unit activities. This means the structure of input data is discarded, in contrast to ConvNets. For the task of image classification, the

input is typically an RGB image with a value specified for each of the 3 channels (red, green, blue) across the 2D plane of the image (the spatial dimension). Thus, the input can be viewed as a volume with three dimensions (higher dimensions are also possible). This structure is in place until after the last convolutional or pooling layer. Furthermore, the volume of units are often interpreted as a stack of *feature maps* where each feature map is a two-dimensional grid of units.

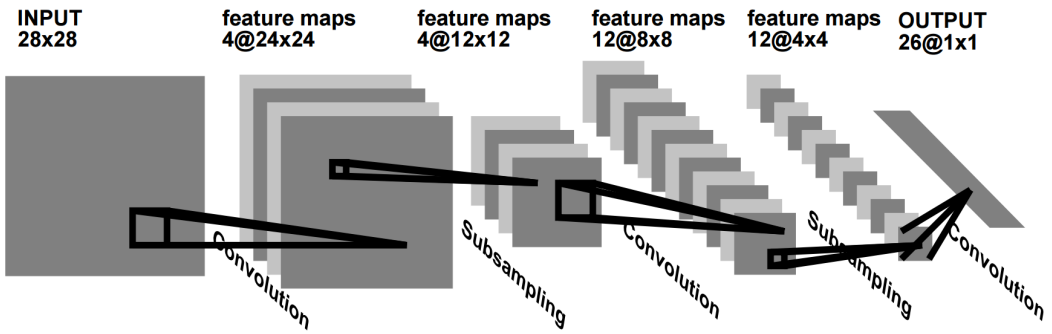

Figure 1: A ConvNet for handwritten digit recognition [10]

The convolutional layer defines several *kernels* (i.e. weight matrices) where the number of kernels correspond to the number of output feature maps. Each unit of an output feature map is computed by a dot product of a subset of the input volume with the corresponding kernel. The subset or *receptive field* is a small region of the input volume with the same dimensionality as the kernel. The layer (as well as the overall network) is called convolutional because each kernel is convolved with the input volume. It is worth noting that kernels are typically described by their spatial dimension e.g. 3x3 despite having the same depth as the input volume. This architecture is a form of *weight sharing* since activations for units in the same output feature map are computed using the same kernel. By restricting the weights, it is intended for kernels to learn to detect a particular feature regardless of where it appears in the input. The convolution operation then combines these local features.

In addition to the kernel size and number of feature maps, the *stride* length and size of input *padding* can be specified. The stride length is the number of units to shift the kernel when performing the convolution. Increasing the stride length further reduces the spatial dimensionality of the output feature maps. The input volume can be padded. This surrounds the input volume with a border of zeros (other values can also be used) thus increasing its spatial dimensionality. The precise choice of values for the two hyperparameters is constrained by valid combinations which give rise to integer spatial dimensions of the output feature maps. In practice, the stride is limited to 1 and padding is used to maintain the spatial dimension of the feature maps before and after the convolutional

Layer	Details
I	227 x 227
C1	96 kernels; 11 x 11; stride = 4
S1	Max-pooling; 3 x 3; stride = 2
C2	256 kernels; 5 x 5; stride = 1; padding = 2
S2	Max-pooling; 3 x 3; stride = 2
C3	384 kernels; 3 x 3; stride = 1; padding = 1
C4	384 kernels; 3 x 3; stride = 1; padding = 1
C5	256 kernels; 3 x 3; stride = 1; padding = 1
S3	Max-pooling; 3 x 3; stride = 2
F1	4096 units
F2	4096 units
O	1000 units

Table 1: Architecture of AlexNet

layer e.g. padding of 1 for 3x3 kernels with stride 1.

Similar to convolutional layers, each unit in the output feature maps after a pooling layer have a receptive field. Units inside the receptive field can be summarised or *pooled* in a number of ways including taking the maximum ('*max-pooling*') or the average ('*average-pooling*'). The purpose of a pooling layer is to reduce the precision to which distinctive features are encoded. By reducing the spatial dimension of feature maps, ConvNets can be made translation invariant. The output layer is a fully-connected layer and may be preceded by multiple fully-connected layers. This is no different to layers found in an MLP. Feature maps are flattened to a one-dimensional array of activations with every unit connected to every unit in the output layer. Just like in an MLP, activations following a convolutional or pooling layer are passed through an activation function.

2.2 ImageNet Classification

The success of AlexNet [3] propagated the use of ConvNets for image classification. All subsequent, top-performing submissions to the ImageNet challenge [4] have been variations of its architecture. As a result, many investigations on convolutional features have been performed on AlexNet.

Table 1 shows AlexNet's architecture where the different types of layers are: Input, Convolutional, Subsampling (i.e. pooling), Fully-connected and Output. We can analyse its *capacity* by calculating the number of weights in the network. This evaluates to 58,649,184 however weights in the convolutional layers only contribute 0.05% towards this total. In

comparison, LeNet-5 [11], a ConvNet for MNIST[1] classification, has a total of 59,470 weights where 0.92% of this total belongs to the convolutional layers. This corresponds to an increase of almost 10^5% reflecting the increased complexity of the ImageNet classification task. In addition, ImageNet contains 1.3 million training examples compared to MNIST's 60,000; this allows for the training of larger models.

AlexNet was one of the first deep, multi-layer neural networks to use the Rectified Linear Unit (ReLU) [12] in place of previously popular saturating activation functions e.g. hyperbolic tangent. The size of weight updates in the backpropagation training rule are restricted when using saturating activation functions. This results in significantly slower training, preventing reasonable training of deep networks.

A technique called *local response normalisation* was employed in AlexNet directly after the 1st and 2nd convolutional layers. This computes a response-normalised activity by dividing a given feature map activation by the sum over feature maps adjacent but for activations at the same spatial position. The principle is inspired by biological models of lateral inhibition; activations in different feature maps but at the same spatial position can be viewed as neurons with the same receptive field but responsible for detecting different features. Only the strongest activities are allowed to propagate forming part of more complex features in the latter regions of the visual pathway. The performance gain of using this technique has been questioned in recent work and hence has been omitted from some state-of-the-art models [13].

To achieve the top-5 error rate of 15.4%, a number of methods were used. Firstly, there was heavy use of data augmentation. This includes extracting random patches from the input image along with horizontal reflections in order to increase the training set size. The RGB components of input images were normalised using PCA[2] for the purpose of incorporating invariance to contrast and brightness. Secondly, the model was pre-trained on 15 million images of the ImageNet Fall 2011 release. Thirdly, ensemble learning was exploited, averaging predictions from 7 similar CNNs - 2 of which contained an additional convolutional layer.

3 Visualisation of features

In [3], the kernels of the first convolutional layer, after training on ILSVRC-2012 [4], are visualised. Since these kernels are convolved with the input image, they have a depth of 3 corresponding to the RGB channels. This means the kernels can be directly visualised al-

[1] Dataset of handwritten digits
[2] Principal component analysis

though the pixel values are typically processed in a consistent way to enhance viewing e.g. contrast enhancement. The visualisations show low-level features such as edges, corners and blobs of varying orientations and frequencies. This is one step towards confirming the view that convolutional kernels behave as feature detectors. However, confirming whether kernels in deeper convolutional layers detect more complex features in a hierarchical fashion would require visualisation of the higher layers. These kernels typically have a depth much greater than 3 e.g. 96 in the second convolutional layer of AlexNet [3]. One approach is to slice the kernels into 2D matrices, as done in [2], but in that particular case the input was a 2D, grayscale image. A similar problem arises when attempting to visualise the individual feature maps. Furthermore, deeper layers tend to use kernels with smaller spatial dimensions i.e. 3x3. These are not very informative when visualised directly as they do not reveal the specific complex structures the kernel is responsive to. This also ignores the fact that the effective receptive field of units in deeper feature maps is a combination of the receptive fields of previous convolutional and pooling layers.

3.1 Deconvolutional Network

A popular approach is to use a Deconvolutional Network (deconvnet) [14] to project specific activations within feature maps to the pixel space [15]. In doing so, the input stimuli of units in convolutional layers can be visualised. The input stimuli correspond to patterns within images that trigger large activations in a given feature map, often referred to as feature activations. This addresses the issue above since, in this approach, feature activations are back-propagated through previous layers. A deconvnet performs the same operations as a ConvNet but in reverse. A rectification layer is still required to ensure activations are always positive. The convolutional layer reconstructs input from the layer above by convolving with a transposed (i.e. vertically and horizontally flipped) kernel. Reversing the operation of a pooling layer is more complicated since the process is not invertible however it can be approximated. The network in [15] uses a max-pooling layer and this can only be reversed if the spatial positions of the propagated activations are recorded. This allows for the maximum activations to be placed in their correct positions in the reconstructed input. Clearly, due to the presence of pooling layers, the projection is input specific hence the deconvnet is tightly coupled to ConvNet at test time. In order to project a particular feature activation, all other activations across feature maps in the same layer are set to zero. The input stimulus is reconstructed by successively passing the activation through unpooling, rectification and convolutional layers until the input space is reached.

3.2 Hierarchy of features

In [15], the 9 strongest activations per feature map are visualised using the deconvnet method for a random subset of feature maps in each convolutional layer. These activations were recorded at intermediate layers when evaluating performance on the validation set of ILSVRC-2012 [4]. These visualisations confirm several properties of ConvNets that were previously assumed. Firstly, the reconstructions of the 9 activations for each feature map show strong coherence. For example, one of the feature maps in the third convolutional layer showed preference for mesh patterns, another in the fifth layer showed preference for faces human or otherwise. This suggests that even kernels in deeper convolutional layers behave as feature detectors, highly responsive to complex patterns and objects regardless of position in the image. Secondly, there is clearly a hierarchical arrangement of feature formation ascending the layers. From simple low-level features (layers 1, 2) to textures (layer 3) to segments of objects (layer 4) and finally to poses of objects (layer 5).

3.3 Evolution of features

The visualisation of features across training in [15] demonstrates the need for ConvNets to be trained for a substantial number of epochs. By observing the clarity of features visualised at different stages of training, it strongly suggests that kernels in deeper convolutional layers converge more slowly than those in the lower layers. Without visualisations, it would have been difficult to otherwise infer this fact and this, along with many other experiments in [15], confirm their usefulness at further understanding ConvNets.

4 Architecture Improvements

The 'ZFNet' [15], winner of the ILSVRC-2013 classification task [4], has an identical architecture to AlexNet with the same number of layers. The only difference is in the 1st convolutional layer where the kernel size is reduced from 11x11 to 7x7 and the stride length is reduced from 4 to 2. These changes were only made because visualisations of the first two convolutional layers highlighted a few concerns. Firstly, there appeared to be multiple filters with similar features as well as an absence of mid-frequency filters. This suggests that the receptive field is too large as distinctive low-level features are not being detected. Secondly, visualisations of the 2nd convolutional layer showed significant distortion i.e. aliasing artifacts. This indicates that the stride is too large; reducing the stride results in cleaner, more precise features.

Various models with architectural changes to AlexNet are evaluated in [15]. A model with

layer F2 (see table 1) omitted results in an increase of just 0.3% in the top-5 validation error. This is initially surprising as the layer contains 28.6% of the total number of weights. Next, a model with the fully-connected hidden layers (F1 and F2) omitted is evaluated. The resulting architecture only contains a single fully-connected layer (the output layer); 84.2% of weights have been removed. Despite this, the error only increases by 4.3%. Similarly, a model with two of the middle convolutional layers (C3 and C4) removed results in an error increase of 4.0%. From earlier, we know the convolutional layers in AlexNet only contribute 0.05% to the total number of weights. This indicates that performance of ConvNets is not strictly related to the number of parameters i.e. capacity of the model. This is further supported by a model in which the number of units in layers F1 and F2 is increased to 8192 (from 4096). The top-5 validation error of the resulting model is unchanged however the top-1 training error decreases by 8.3%, likely the result of overfitting. Furthermore, removing layers C3, C4, F1 and F2 causes the validation error to jump to 50.1%, an increase of 32%. This suggests that network depth is important.

Motivated by the findings above, 'VGGNet' [13] was constructed. This model further improved the performance on ImageNet [4], winning the localisation task of ILSVRC-2014 as well as achieving 2nd place in the classification task. This model reduced the kernel size to 3x3 and increased the number of convolutional layers. It turns out that a stack of three convolutional layers with 3x3 kernels has the same receptive field as a single 7x7 convolutional layer. However, using the stacked approach introduces greater non-linearity in the model since the convolutional layers are interweaved by rectification (ReLU) layers. Not only is the learned function more discriminative as a result but there are fewer parameters. Thus, this could also be considered as a form of regularisation. The work in [13] evaluated the impact of depth on performance by constructing models with 11, 13, 16 and 19 weight layers. 3 of these layers were constrained to be fully-connected layers with the same number of units as AlexNet. Evaluated on the ILSVRC-2012 dataset, performance improved with increased depth. Finally, a model ensemble incorporating the 19 weight layer network achieved a test error of 7.3% compared to ZFNet's 13.5%.

5 Semantic clustering of features

Visualisation of features looked at individual activations in deep convolutional layers. We now look at vectors of activations from the last hidden layer of ConvNets. The work in [16] explores the effectiveness of AlexNet [3] as a feature extractor by visualising the semantic clustering of feature vectors output from the second 4096-dimensional fully-connected layer. The t-SNE algorithm [17] is used which transforms the high-dimensional vectors

so they can be viewed in two dimensions. AlexNet [3] was trained on the training set of ILSVRC-2012 [4] hence the algorithm is run on feature vectors resulting from propagating the corresponding validation set through the ConvNet. For feature vectors extracted from layers close to the output layer, clustering of images belonging to the same class is expected due to the supervised nature of the task. Therefore, to properly examine the semantic clustering of deep features, labels are combined to form more general classes by using WordNet's word groupings [18]. The labels of the 50,000 validation set images are reclassified to one of 7 classes which are semantically distinct.

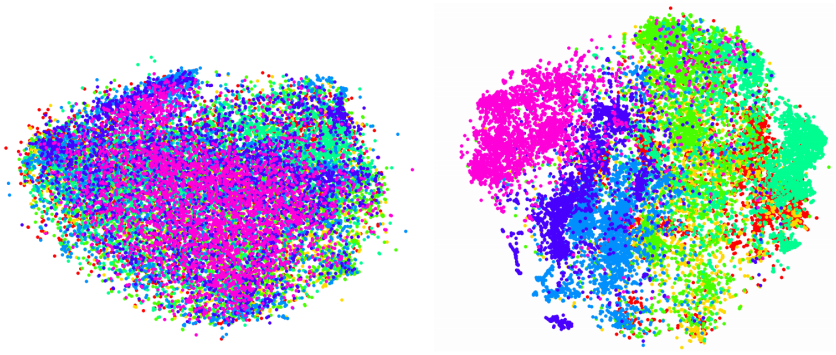

Figure 2: Feature vector visualisations for 1st convolutional layer (left) and last hidden layer (right) [16]

The visualisation in [16] shows clear semantic clustering of feature vectors from the last hidden layer (figure 2). A visualisation of feature vectors extracted from the first convolutional layer is also presented indicating wide intra-class variability. This indicates that information irrelevant to object categorisation such as pose and illumination, which cause significant differences in the pixel representation of images, are discarded in the deeper layers.

A similar experiment is performed in [3] where feature vectors from the last hidden layer are generated for images in the training and test set of ILSVRC-2012 [4]. Images from the test set are selected and for each one, images from the training set with the smallest Euclidean distance between corresponding feature vectors are obtained. It is observed that each set of images appear to form a coherent group e.g. images of elephants and that the images are not very similar on a pixel level. This provides further evidence that features extracted from higher layers have semantic clustering properties.

6 Generalisation to related tasks

From the visualisation of feature activations, we have seen that ConvNets are able to learn to discriminate between objects even if they are not explicitly identified within the training database. Further to this, vectors of feature activations extracted from deep layers show strong semantic clustering properties. This is, in part, a consequence of the large number of examples contained within the training set - ImageNet [4]. This motivates whether the trained network with its learnt features can be generalised to related tasks within computer vision. That is to say whether ConvNets trained on large, labelled datasets have "sufficient representational power and generalisation ability" [16] to successfully perform related tasks without re-training.

6.1 Transfer learning

The use of a trained, deep ConvNet as a feature extractor is often referred to as *transfer learning*, a form of multi-task learning [19]. Recent experiments have demonstrated the effectiveness of this approach, particularly on discriminative tasks with relatively few training examples. One way of doing this is to remove the output layer and use the feature vectors generated by the last hidden layer as input to a linear classifier (e.g. SVM). Images from the new dataset propagated through the network are preprocessed in the same way as when the ConvNet was originally trained. The linear classifier can be trained relatively quickly due to the small training set and small number of parameters. In fact, we are not restricted to the last hidden layer. Experiments in [16] have shown superior performance when using the second last hidden layer and it is even possible to use shallower intermediate layers. An alternative approach employed in [15] is to adjust the output, softmax layer to contain the appropriate number of dimensions for the new task. Only weights connected to the new output layer are randomly initialised; weights for unchanged connections are left to their trained state. These weights may be allowed to update via the backpropagation process during training or can be fixed. This can be viewed as a form of supervised pre-training in contrast to unsupervised techniques for deep architectures [20].

6.2 Experiments on Caltech

A quantitative evaluation on the Caltech-101 dataset [21], a dataset consisting of objects in natural settings, is performed in [15, 16]. The dataset is similar in nature to ImageNet [4] however there are only 101 classes and 9,146 images compared to ImageNet's 1000 classes and 1.3 million training images. The previous state-of-the-art when trained on 30 images

per class is 84.3% [22] - the mean accuracy per class. An accuracy of 86.5% is obtained using a pre-trained ConvNet in [15]. When all the weights in the network are randomly initialised, an accuracy of 46.5% is obtained. This confirms that it is the pre-training of weights which is responsible for the significant improvement in performance.

The work in [16] evaluates the performance of a logistic regression (LogReg) model and a support vector machine (SVM) trained on features extracted from the 5th, 6th and 7th layers (i.e. last 3 hidden layers). The best performance for the LogReg model is 84.9% attained on features from the 7th layer. However, the best performance on the SVM is obtained using features from the 6th layer - an accuracy of 84.8%. It is hypothesised that the superior performance of the pre-trained ConvNet over the linear classifiers is due to the fact that AlexNet [3] employs DropOut [5] in the fully-connected layers. DropOut is a method to prevent overfitting by setting activations output from the previous layer to zero during training. The activations are randomly selected with a probability p (usually $p = 0.5$) and vary throughout training. Inspired by DropOut, a linear classifier can be trained on features where during training half of the components in each feature vector are dropped. By employing DropOut, an SVM trained on features from the 6th layer achieves the best performance with an accuracy of 86.9% [16] exceeding the performance of the pre-trained ConvNet.

Different layers from the ConvNet achieve the best performance for the two feature extraction techniques discussed above. [15] evaluates the generalisation ability of every layer in their ConvNet. An SVM was trained on features ascending every layer for the Caltech-101 [21] and Caltech-256 [23] datasets. Generally, the performance improves, the deeper the layer used. This is expected since the deeper layers contain rich semantic representation where the composite feature detectors respond to more abstract patterns. Although the specific deep layer which results in the greatest generalisation performance varies depending on the dataset and task at hand.

7 Conclusion

Convolutional neural networks learn a hierarchy of features. In the 1st layer, features are directly represented by the specific combination of weights in the kernel. These depict low-level artifacts such as edges, corners and colour blobs with varying orientations and frequencies. However, for deeper layers, the weights in individual kernels are uninformative. These abstract features are represented by the combination of previous kernels and subsampling operations. As a result, discovering the different kinds of feature detectors learned is a difficult problem.

A deconvnet is one solution which back-propagates selected feature activations to the pixel space. In doing so, we can reconstruct patterns which give rise to large activations. This has shown that units within the same feature map are responsive to the same stimulus (translation invariance). In addition, patterns appear to get increasingly complex the deeper the layer; example reconstructions show textures in early layers, moving on to segments of objects in higher layers.

This hierarchical structure is further confirmed by considering each layer of activations as a feature vector. Feature vectors from deeper layers possess rich semantic properties. Visual representations show clustering in the feature space into categories that are not classes in the supervised task. Without fine-tuning, benchmarks on related tasks have been eclipsed demonstrating their generalisation ability.

Visualisations have shown that smaller receptive fields and unit strides give rise to more distinctive and precise features. Experimentation of ConvNets with a variety of configurations have demonstrated that the number of weight layers (depth) as opposed to the number of parameters (capacity) result in superior performance.

7.1 Future work

Recent ConvNet architectures have diverged away from the conventional design of a linear stack of convolutional, pooling and fully-connected layers [1, 3]. These new models include the 'GoogLeNet' [24] and the 'ResNet' [25] which offer radically different connectivity patterns. This means their internal operation may differ which includes the format of feature representation. Visualisation of unit activations is a starting point for future work followed by an investigation into the generalisation ability at various depths in the network. Despite this, both models continue a trend observed previously; that is increased depth and fewer parameters.

References

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[2] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3642–3649, IEEE, 2012.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[5] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[6] S. Knerr, L. Personnaz, and G. Dreyfus, "Handwritten digit recognition by neural networks with single-layer training," *IEEE Transactions on neural networks*, vol. 3, no. 6, pp. 962–968, 1992.

[7] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[8] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., DTIC Document, 1985.

[10] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[11] 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, 1998.

[12] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *2011 International Conference on Computer Vision*, pp. 2018–2025, IEEE, 2011.

[15] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*, pp. 818–833, Springer, 2014.

[16] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition.," in *ICML*, pp. 647–655, 2014.

[17] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[18] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[19] R. Caruana, "Multitask learning," in *Learning to learn*, pp. 95–133, Springer, 1998.

[20] I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. W. Aha, "Unsupervised and transfer learning challenge," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 793–800, IEEE, 2011.

[21] F.-F. Li, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[22] J. Yang, Y. Li, Y. Tian, L. Duan, and W. Gao, "Group-sensitive multiple kernel learning for object categorization," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 436–443, IEEE, 2009.

[23] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.