Encoded Invariance in Convolutional Neural Networks

Nathaniel Sauder
University of Chicago
nsauder@uchicago.edu

Abstract

Deep neural networks have proven remarkably effective on tasks where euclidean distance in the raw input space lacks semantic meaning and thus where an intermediate representation must be constructed before the usual suite of classifiers can be used effectively. Traditionally, these representations were painstakingly handcrafted in order to capture a sufficiently strong signal while reducing dimension. However, deep learning methods attempt to learn this representation over many examples. In this paper, we review and propose methods of encoding invariance in the learnt representations whereby the requisite dimension reduction can be achieved while nonetheless preserving the signal.

1 Neural Networks

1.1 Biological Inspiration

Neural networks were inspired by central nervous systems. However their machine learning implementations are simple to describe: repeated compositions of linear maps with non-linear activation functions. The biological analogs are dendrites sending neurotransmitters into a neuron which then passes the signal along to its neighbors if a threshold is met. Convolutional neural networks fix some weights to be equal. In particular, they encode the $2d$ translational covariance, i.e filters applied in the top right patch will also be applied in the bottom left. Invariance and covariance are essential to the success of convolutional neural networks. Indeed, the name requires a group structure over which to convolve. In computer vision, this group has traditionally been the set of $2d$ translations while in time series or speech modelling it has been the 1d time axis. In this paper, we propose methods for how convolutional neural networks might encode further, more complex invariance.

1.2 Applications

Convolutional Neural Networks are widely applied. Indeed, they are currently dominant on a variety of benchmarks in computer vision. Krizhevsky 2012 Imagenet ConvNet trained on 2 GPUs stands out as demonstrating the power of these methods. [1] However, ConvEnts have also performed well on seemingly unrelated tasks like classifying the quantum physical quantities of proposed molecules. [2]

1.3 Layers in Convolutional Neural Nets

ConvEnts are composed of several different types of layers. Convolutional layers compute response maps derived from convolving a 3d filter across the feature maps in the previous layer. Activation functions are then computed for each point and are analogous to thresholding in actual neurons. Sigmoids and hyperbolic tangents are popular choices however ReLUs, $max(0, x)$, are increasingly used due to their sparse output and lack of saturation. These altered response maps then undergo spatial subsampling or pooling, biologically inspired by Hubel and Wiesels work in the 1960s on models of human vision, where local responses – for instance, 7x7 windows – are pooled into a single value for input into the ensuing layer. After several layers of stacked convolution, activation, pooling blocks, the outputs are fed into a series of fully connected layers before softmax generates the class probabilities. Much experimentation has taken place regarding the optimal procedure for each of these components however we will briefly mention two more methods that were critical to Krizhevsky et al.s success in 2012: Local Response Normalization and Dropout. LRN layer involve normalizing weights at a specific point (x, y) over n feature maps. LRNs create competition between neurons for large activations. These make particular sense in the last convolutional layer as these features tend to be as high-level as houses or sheep – mutually exclusive categories. Finally, Dropout prevents cohabitation of features by randomly dropping half of the neurons in the fully connected layers. Given sufficient sparsity, Dropout approaches model averaging over exponentially many models.[3]

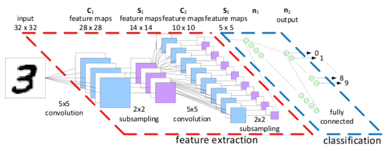


Figure 1: Basic architecture of ConvNets – missing many of the layers just discussed

2 Pooling

2.1 Introduction

Pooling is the primary source of dimension reduction and of local translation invariance in a ConvNet. Accordingly, pooling mechanisms are judged by how much they can reduce the dimension while preserving the signal. Invariance of the image class under local group actions are thus essential to understanding how these dueling objectives can be balanced. Theoretical results indicate that all forms of ℓ_p pooling preserve the signal provided the feature maps are sufficiently redundant. [4] However, in practise, different pooling mechanisms offer stark changes in performance. Accordingly, we consider and propose several pooling regimes, catalog their results on benchmark data-sets while providing some mathematical and computer vision to support their effects.

2.2 Ave vs Max

There are two main approaches to pooling: averaging the responses and taking their maximum. These are complementary in that they belong on opposite ends of the ℓ_p spectrum:

$$\begin{aligned} \ell_1 &\iff \text{avg} \\ \ell_\infty &\iff \text{max} \end{aligned}$$

Examining their differences will motivate the recently implemented pooling regimes as well those we are introducing ourselves. Boureau et al. have written the main paper on the subject which advocates using max pooling when representations are sparse. Intuitively, this seems reasonable as the presence of a house in a part of an image should not be diluted by its absence in other regions. [5] Similarly, the firing of a vertical edge detector in a part of the image should not be diluted by its absence in a local region. The outputs of average and max pooling networks in [6] confirm the intuition that average pooling dilutes the signal excessively. picture

2.3 Heterogeneous Pooling

An alternative to debating whether max or average is universally preferable is to include both in the network. In particular, under this scheme some feature maps are pooled by max, others by average. Indeed, for spatially distinct complex features (keypoints, maybe edge-like structure), max is likely preferable; whereas for feature maps that are less spatially localized, e.g., color or broad texture, average may be better. In our experiments, performance was essentially identical to max pooling while

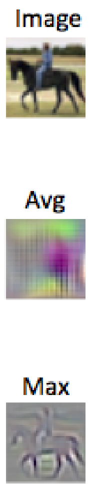


Figure 2: Effects of max and average on signal quality. [6]

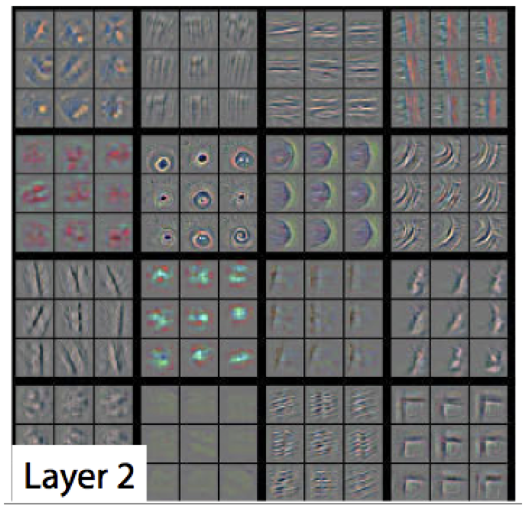


Figure 3: Visualizations of 2nd layer of Imagenet network. [7]

convergence was marginally quicker. Heterogeneous pooling can be easily extended to an arbitrary number of different pooling regimes. Finally, another alternative is feeding each set of responses to a suite of pooling regimes and thereby increasing the signal preservation while hampering the dimensionality reduction. In sum, while performance was not improved by adopting heterogenous pooling mechanisms within the same network, we believe that heterogenous applications of pooling may prove fruitful as there is substantial variety both within filter banks – color vs edge – as well as between layers – color vs house detector.

2.4 ℓ_p Pooling

Another natural extension of the max/avg debate is to observe that avg and max lie on opposite ends of the ℓ_p spectrum and question whether intermediate values may be optimal. In 2012, Sermanet et al. tested this hypothesis on the Street View House Numbers dataset where the task is identifying the street number from house signs. [8] They also chose pooling regions with significant overlap and performed ℓ_p pooling with a Gaussian envelope instead of pure ℓ_p pooling. Not only is the optimal norm not average or max, the advantage offered by choosing the optimal norm, ℓ_4 is significant: 3% improvement. While these results would suggest that grid search on the optimal norm for a given problem might yield a choice other than average or max, more recent experiments suggest that if $p < \infty$ then the model averaging approximation of dropout is weakened sufficiently to erode performance.

2.5 Stochastic and Entropy Pooling

Whether max or avg pooling is chosen, the resulting ConvNets overfit significantly. [6] Borrowing from the main idea of dropout: stochastically removing neurons in order to train each independently, Zeiler and Fergus propose a similar probabilistic regime for pooling. In particular, the probability of propagating response w_{ij} for the region P is $\frac{w_{ij}}{\sum_P w_{ij}}$ where the sum is taken over all units in the pooling region. As this generates too much noise when testing, they choose $\mathbb{E}(P)$ as output under those conditions. The effect is a large reduction in overfitting resulting in substantially better performance on the suite of benchmark small datasets: CIFAR-10, CIFAR-100, SVHN, and MNIST. To explain their results, Zeiler and Fergus argue that stochastic pooling enables model averaging in the convolutional layers. Furthermore, they claim that stochastic pooling ensures a degree of deformational invariance. Accordingly, it would be instructive to test Szegedys adversarial examples, [9], on a ConvNet with stochastic pooling.

As just discussed, a stochastic pooling ConvNet take the expectation of the response when testing. We observe that this amounts to $\frac{\ell_2^2}{\ell_1}$ pooling. They report that *training* with expectation pooling encounters the same troubles with overfitting. However, perhaps the superiority of max pooling suggests that pooling with probabilities assigned proportionally to an ℓ_p norm would yield stronger results. Instead of obeying the $\frac{\ell_{n+1}^2}{\ell_n}$ restriction, we tried entropy pooling: $\frac{\ell_\infty^2}{\ell_1}$ or $\frac{max^2}{avg}$. Intuitively, given a clear peak in response the output of entropy pooling will be proportional to the squared height of the peak; whereas for uniform responses, the output will be the average. [1] Essentially, this amounts to TF-IDF pooling – weighing down more common responses. Empirically however it does not offer an improvement.

2.6 Maxout

Building off the immense success Dropout training methods have had, [3], Goodfellow et al. sought to specifically design an activation function to improve the model averaging approximation. [10] Maxout units take the maximum of responses over a specific pooling region across multiple feature maps. However, maxout is an activation function as ReLUs are not applied to the responses before being fed into the maxout unit. (Indeed, doing so causes a 0.2% reduction in performance on MNIST) As a consequence, the response can be negative. An entire maxout unit can be seen as an universal approximator as any continuous function can be approximated by the maximum of a set of convex and concave functions. Thus, in a sense, the ideal activation function is being learnt. A potential example of a maxout unit involves maxing over two vertical edge detectors: one with rightward gradient and the other leftward. Taking the max of the two produces a positive response if there is an edge regardless of direction of color change. Goodfellow et al. claim that sparsity and non-saturation allows for closer approximations to model averaging. Furthermore, the increased sparsity of the gradient allows for the global minimum to be more easily approximated.

Given competing evidence for different pooling schemes, it is natural to ask whether the optimal norm can be learnt. Accordingly, Gulcehre et al. implement an ℓ_p unit that computes the ℓ_p norm over the responses in a spatial region across several feature maps. [11] However, the value of p in ℓ_p is learnt by backprop. The authors justify this use by arguing that the curved boundaries offered by the ellipses of ℓ_p units allow for more realistic approximation of the true activation function. Interestingly, for MNIST their optimal norm value is 3.4 – similar to the optimal choice, 4, of [8]. Indeed they also find that different units prefer similar norm values: the standard deviation is 0.38. [summary argument]

2.7 Global Pooling

Both maxout and learned norm pooling increase the complexity of the convolutional layers by introducing highly non-linear activation and pooling functions. Network in Network goes further by convolving with a 3 layer neural network rather than linear filters. [12] As this offers a great increase in complexity in the convolutional layers, the authors remove all fully connected layers by global average pooling across the last feature maps and feeding the resulting vector into the softmax layer. Furthermore, they require that the number of feature maps is equal to the number of classes thereby learning one feature map per class. Their explanation also relies on the language of universal approximation. After all, a 3 layer MLP is a universal approximator and thus can theoretically approach any filter. However, if we accept Mallats interpretation of the convolutional layers as rotations of the input space – to be followed by pooling contractions – then the non-linearity of the filter is immaterial. Indeed, under this interpretation the non-linearities in ReLUs and pooling operators are sufficient to achieve the non-linear twists in the decision boundary. Nonetheless, the results of NIN are excellent and furthermore they improve with the difficulty of the task. Successful performance on the significantly more challenging Imagenet data-set would certainly validate this approach further.

Perhaps of greater relevance to architecture choice in regular ConvNets is the global average pooling undertaken in the last convolutional layer. Indeed, *Network in Network* [12] show that global average pooling is also an effective regularization technique for traditional ConvNets. While it does not outperform dropout, it does improve upon the usual arrangement of fully connected layers. That

[1]While $\frac{max}{avg}$ might seem initially more appealing, it is discontinuous at zero.

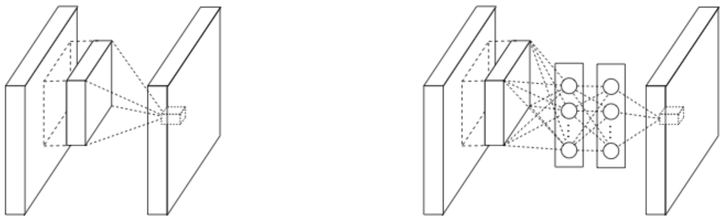


Figure 4: Left, convolution by linear filter. Right, convolution by 3 layer neural network [12]

a crude measure such as global averaging offers an improvement demonstrates the severity of the overfitting in the fully connected layers. Finally, it seems plausible that global max pooling would outperform global average pooling as max pooling would capture the simple existence of an object in the space whereas average pooling would dilute this signal – given the features learnt in the last convolutional layer, max pooling seems the obvious choice. Another benefit to these scheme is its invariance to aspect ratios of the input. While in classification tasks the input images have a consistent, usually dyadic shape, in detection the bounding boxes can be irregular.The current State of the Art on detection [13] addresses this problem by resizing the image to fit the desired size of the ConvNets thereby incurring a significant loss of information. Global pooling thus offers a simple alternative.

2.8 Summary

In sum, a heterogeneous approach to pooling seems preferable as the difference in the filters merit different approaches. Whereas the final layer benefits substantially from local response normalization and more generally from competition among its neurons, the earlier filters seem more complementary than adversarial. Indeed, whereas the earlier layers perhaps should be viewed primarily from a mathematical perspective: as first order Taylor approximations of the local "surface – color filters as measurements of height and edge filters as a measure of local gradients, the later layers seem more approachable by computer vision heuristics. Accordingly, we believe that varying pooling mechanisms both between layers and within layers will result in improvements in classification as well as detection, given more experimentation.

3 Invariance

3.1 Rigid Motion Invariance

Having explored a mechanism by which the ConvNets forcibly encodes a symmetry invariance – local translation – a natural question arises: how much invariance does a ConvNets learn without enforcing symmetry through architecture decisions. In particular, how different are the representations of two images which differ only by an affine action? By a small deformation? Goodfellow et al. reveal that they are very consistent across actions by rigid motion while highly unstable under adversarial deformation. Measuring Invariances in Deep Networks employs convolutional sparse encoders in order to measure invariance with respect to SO_2, SO_3 and T by observing whether neurons continue to fire while image is acted upon by group elements. Goodfellow et al. find that invariance is highly dependent on the depth of the architecture especially 3D rotations.

3.2 Invariance under Deformations

Invariance under deformation has been less studied however a recent paper presented some surprising results. [9] Given an image and a small choice of epsilon, they were able to find an adversarial example with the ball $\beta(image, \epsilon)$. In fact, for most images Szegedy et al. are able to find one that is visually indistinguishable and yet classified as an ostrich. Furthermore, these adversarial examples persist across different architectures, i.e. the found examples are robust. Finally, a spectral analysis of each of the layers in a SOA Imagenet network [1] revealed that the convolutional layers had large lipschitz coefficients while also finding that the lipschitz bound was sufficiently high to allow for highly expansive mappings. Accordingly, this result indicates that ConvEnts are not robust to deformation. Nonetheless, the Imagenet network continues to classify at 51% level on images that are distorted by *random* gaussian noise. Therefore, it is clear that while networks are not invariant to distortion, not much performance is lost on distortion errors.

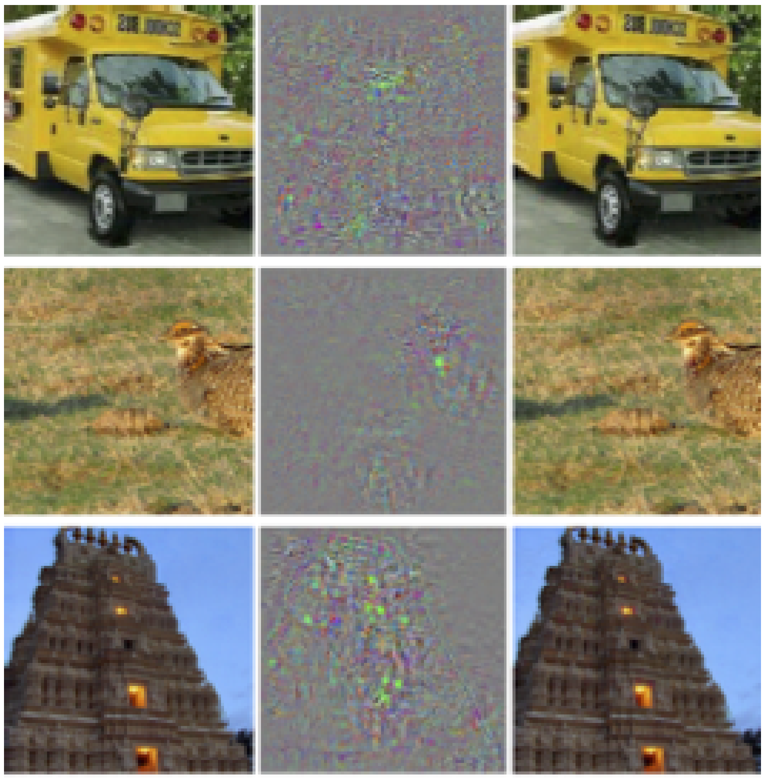


Figure 5: Left, original image classified correctly. Center, difference between left and right images. Right, classified as an ostrich. [9]

Thus, we find that our ConvNets are stable to action by rigid motions yet highly unstable under small deformations. Accordingly, we would like to find a representation that has a small lipschitz coefficient: it maps nearby points in image space to nearby points in the representation space. A deep learning evangelist may argue that we should simply learn these invariances and not constrain the architecture. The following work addresses some of these concerns.

3.3 Wavelet Network Algorithm

A couple of years ago, Mallat proposed cascading wavelets filters as a means of representing natural images or textures. The objective is to construct a representation that is not only invariant to rigid motions as ConvNets appear to be, but also to deformations. Accordingly, Mallat proposes wavelet networks which essentially are ConvNets where the learnt linear filters are replaced by fixed wavelets. Critically, given a compact lie group, G, Mallat proves in [14] that a wavelet network can be constructed that is invariant to G as well as *stable* under deformations. In particular, the algorithm calls for choosing k multiscale wavelet filters and convolving them with the image space over the desired group of actions – for example, $2d$ translations and rotations. Applying the modulus operation to each response yields k non-negative response fields. Now, convolve each feature map with the same suite of wavelets and reapply the modulus operator. Doing so to the desire depth, d, yields k^d feature maps. At the end, subsample using Gaussian ℓ_1 pooling and feed the resulting (scattering) coefficients into a linear classifier. In sum, wavelet networks provide theoretical guarantees of invariance under deformation at the cost of not learning task specific filters.

While the previous section describes the general flavor of the procedure, some facts are worth quickly mentioning. First, clearly for computational reasons a full wavelet basis cannot be used. Secondly, as repeated modulus along a given path dilute the signal, responses that result from paths with three or more modulus operations are discarded as well as those where the second wavelet has a higher frequency than the first. Pruning the possible paths in this way yields a complexity of $n * log(n)$ instead of n^2 for a three layer wavelet network. Thirdly, after each layer the ℓ_1 averages of the responses are added to the representation vector. Some convolutional neural networks also output each layer into the representation, i.e. multistage architectures [8].

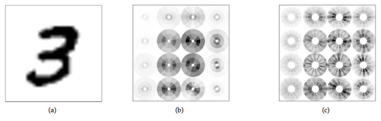


Figure 6: scattering coefficients for digit 3. Eight rotations of wavelet are chosen hence eight scales. Each slice is divided into eighth depending on orientation of the second wavelet. Scale information in radial direction. [15]

3.4 Pooling in Wavelet Networks

Unlike Boureau et al. suggest and other ConvNets implement, Mallat and Bruna find that average pooling outperforms max pooling. Indeed, it does so by 8% when using complex wavelets while average and max perform equally well when employing real wavelets. They justify this result by deriving the ℓ_1 norm as the only functional, M, that satisfies the following conditions:

$$\text{1) commutes with deformations: } M \circ \Lambda_\tau = \Lambda_\tau \circ M$$
$$\text{2) unitary: } \|Mx\| = \|x\|$$
$$\text{3) lipshitz condition: } \|Mx - My\| \leq \|x - y\|$$

Finally, as the wavelet representations can be inverted with the wavelet transform little information is lost – (some is lost as the full frequency is not employed). It would be interesting to see whether the distinction between average and max in regular ConvNets evaporates if multistage architectures are employed.

3.5 MNIST experiment

Mallat and Bruna first apply these networks to MNIST. The convolution is only over the translation group while the wavelets are scaled and rotated. Note this differs from ConvEnts where the invariance group are again $2d$ translations however the filters are not required to be scaled and rotated versions of a mother fitler. They derive near SOA results on the canonical dataset. The scattering coefficients are fed into PCA regression as well as an SVM. In an additional test that illustrates the strength of the invariance in these networks, the authors rotate randomly rotate the digits in the MNIST Dataset – removing 9s – and rerun their experiments and compare against a retrained ConvNet. The wavelet wetwork dramatically outperforms the ConvNet.[15]

3.6 Texture and Image Classification Experiments

Sifre and Mallat extended this work by attempting to build invariance to the entire affine group rather than simply the group of translations. One method of doing so might involve cascading a layer encoding horizontal invariance with one encoding vertical invariance with one encoding rotational invariance etc... In other words, suppose our desired invariance group is G and $G = G_1 \rtimes G_2$. Now let the first layer be invariant to G_1 and covariant to G_2 and the second be invariant to G_2. We then arrive at a representation that is invariant to actions by G. However, this approach can subtly run astray. For example, the figure below shows that unless horizontal and vertical invariances are established simultaneously two different images can be mapped to the same representation. Thus, they attempt to compute the rotation and relational invariants simultaneously for the first layer while separating them for the second. However, as scaling obeys a power law, it can be linearized by the logarithm and thus separated from the other invariances. Finally, as small deformations are linearized by cascading wavelets networks, [14] they are eliminated in the PCA regression on the scattering coefficients.

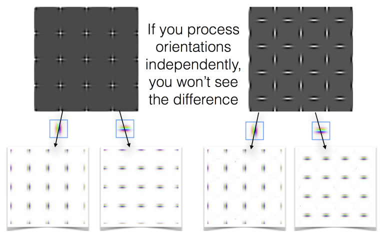

If you process orientations independently, you won't see the difference

Figure 7: Loss of information if convolutions are separately computed. [15][15]

Mallats group recently ran two other experiments using wavelet networks. One, scattering coefficients are computed on images of textures where actions by the affine group can substantially distort the texture image. [16] The figure below demonstrates that the second layer was instrumental in extracting higher moment information required to distinguish the two textures which have the same fourier power spectrum. The results presented in figure 9 indicate how critical encoding invariance to the entire affine group is as roto-translation subgroup does not offer state of the art performance by itself.

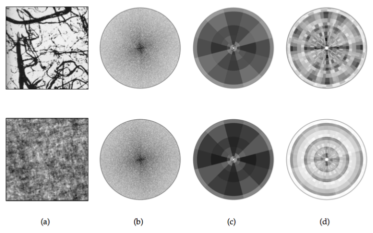
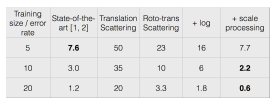


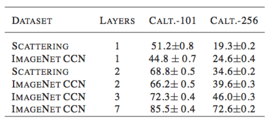
Figure 8: Demonstrates capacity of wavelet networks to differentiate between two images with identical fourier modulus. a) Original Image b) Fourier modulus c) first layer scattering coefficients d) second layer scattering coefficients [15]

Two, they are also computed for Caltech $101/256$ and compared against representations learnt by the first two layers of a ConvNet. In the second experiment, Oyallon et al. [17] compute the scattering coefficients generated by the first and second layers of wavelet networks as well as the representation derived from the first and second layer of a regular ConvNet by feeding them into separate SVMs. The results are very similar: see figure 10. These results suggest that the quality of the filters produced by Mallats wavelet networks possess similar representative power as those learnt in the in the first two layers of a regular ConvNet.



Training size / error rate	State-of-the-art [1, 2]	Translation Scattering	Roto-trans Scattering	+ log	+ scale processing
5	**7.6**	50	23	16	7.7
10	3.0	35	10	6	**2.2**
20	1.2	20	3.3	1.8	**0.6**

Figure 9: Results on UIUC benchmark texture dataset. [15]



Dataset	Layers	Calt.-101	Calt.-256
Scattering	1	51.2±0.8	19.3±0.2
ImageNet CCN	1	44.8 ± 0.7	24.6±0.4
Scattering	2	68.8± 0.5	34.6±0.2
ImageNet CCN	2	66.2± 0.5	39.6±0.3
ImageNet CCN	3	72.3± 0.4	46.0±0.3
ImageNet CCN	7	85.5± 0.4	72.6±0.2

Figure 10: Precise differences – slight – between results of avg pooling with wavelet networks. [17]

4 Future Work

4.1 Small Illuminating Tweaks

In sum, there are three main methods of introducing invariance into a Conv Net: convolving over a given group and accumulating the results, altering the pooling function to include more general local invariance, and finally data augmentation – presenting the network with rotated images and averaging the final probabilities. We propose some quick empirical tests to develop better intuition for ConvNet representations:

- repeat Girshick experiments with global max pooling; determine whether there is a relationship between degree of rescaling and error rate
- varied regularization constants for fully connected and convolutional layers. Most agree, that fully connected layers are responsible for overfitting. Does independently regularizing these layers reduce overfitting?
- visualize maxout filters in lower levels to determine how color and edge filters are being combined.
- regularize the lipschitz constant in the convolutional layers in order to prevent expansive behavior in convolutional layers.
- run same adversarial paradigm to determine whether it is always possible to find rotations or translations that result in ostrich classification

4.2 Larger Scale source of improvements

While the previous experiments offer an opportunity to develop a deeper empirical understanding of ConvNets, we believe that more fundamental improvements can be found in the following two ways:

1. Transferring the bulk of the parameter set from the fully connected layers into convolutional layers. There are many methods are doing so: global max pooling (previously discussed), maxout was able to learn productively to significantly greater depth – 7 layers, network in networks convolution by neural networks, and multistage architectures are some examples. It seems that Sermanets 2014 Imagenet network will illustrate the power of this approach.
2. Encoding symmetry in convolutional neural networks more explicitly. Enforcing rotationally symmetric and scaled filters, enforcing invariances at different layers according to the desired locality, or factoring higher dimensional convolutions into multiplications of lower dimensional ones. For example, while at Google, Joakim Anden factored the $3d$ convolutions in an imagenet network, i.e. apply $2d$ filters and then pass a $1d$ convolution over their outputs to arrive a factore $3d$ convolution, resulting in substantially faster convergence.

4.3 Conclusion

The successful move from fully connected feed-forward neural nets to convolutional neural nets was an insight of symmetry: local translation invariance, global translation covariance. Given that natural images and especially domain-specific images (houses, eyeballs, etc...) obey far more symmetries, developing a globalized framework to encode domain and hierarchical invariance would be useful both theoretically and practically– currently, invariance is usually provided by data augmentation a la Sander Deliemans Kaggle Galaxy Zoo solution. A step in this direction was a recent paper [18] where ConvNets were applied to generalized graphs rather just grid graphs. A final step would be learning the invariances directly from the data. While this direction proffers fanciful hopes such as learning physics rules directly from data, in the short term we believe that increased attention to encoding invariance may offer faster convergence, fewer examples required, and perhaps better results as greater depth becomes possible with reduction in parameters.

4.4 Acknowledgments

I would first like to thank Prof. Shakhnarovich for his wonderful mentorship this summer. He was exceptionally kind in welcoming the REU students into his group. I am particularly grateful for the extended time he spent with me ensuring a clean start to this project. Secondly, I would like Prof Babai and Prof Kurtz for organizing this wonderful REU as well as for preparing some wonderful food for the potluck.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Lon Bottou, and Kilian Q. Weinberger, editors, *NIPS*, pages 1106–1114, 2012.

[2] G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Müller, and O. Anatole von Lilienfeld. Machine learning of molecular electronic properties in chemical compound space. *New Journal of Physics*, 15(9):095003, September 2013.

[3] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[4] J. Bruna, A. Szlam, and Y. LeCun. Signal Recovery from Pooling Representations. *ArXiv e-prints*, November 2013.

[5] Y-Lan Boureau, Jean Ponce, and Yann Lecun. A theoretical analysis of feature pooling in visual recognition. In *27TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, HAIFA, ISRAEL*, 2010.

[6] M. D. Zeiler and R. Fergus. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. *ArXiv e-prints*, January 2013.

[7] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[8] P. Sermanet, S. Chintala, and Y. LeCun. Convolutional Neural Networks Applied to House Numbers Digit Classification. *ArXiv e-prints*, April 2012.

[9] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[10] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout Networks. *ArXiv e-prints*, February 2013.

[11] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio. Learned-Norm Pooling for Deep Feedforward and Recurrent Neural Networks. *ArXiv e-prints*, November 2013.

[12] M. Lin, Q. Chen, and S. Yan. Network In Network. *ArXiv e-prints*, December 2013.

[13] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[14] S. Mallat. Group Invariant Scattering. *ArXiv e-prints*, January 2011.

[15] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *CoRR*, abs/1203.1513, 2012.

[16] Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 1233–1240, 2013.

[17] Edouard Oyallon, Stéphane Mallat, and Laurent Sifre. Generic deep networks with wavelet scattering. *CoRR*, abs/1312.5940, 2013.

[18] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral Networks and Locally Connected Networks on Graphs. *ArXiv e-prints*, December 2013.