Transforming Autoencoders

G. E. Hinton, A. Krizhevsky & S. D. Wang

Department of Computer Science, University of Toronto
{geoffrey.hinton, akrizhevsky, sidawang88}@gmail.com

Abstract. One way to design an object recognition system is to define objects recursively in terms of their parts and the required spatial relationships between the parts and the whole. A natural way for a neural network to implement this knowledge is by using a matrix of weights to represent each part-whole relationship and a vector of neural activities to represent the pose of each part or whole relative to the viewer [10]. This leads to neural networks that can recognize objects over a wide range of viewpoints using neural activities that are "equivariant" rather than invariant: as the viewpoint varies the neural activities all vary even though the knowledge in the weights is viewpoint-invariant. The "capsules" that implement the lowest-level parts in the shape hierarchy need to extract explicit pose parameters from pixel intensities. This paper shows that these capsules are quite easy to learn from pairs of transformed images if the neural net has direct, non-visual access to the transformations.

Keywords: Invariance, auto-encoder, shape representation

1 Introduction

Current methods for recognizing objects in images perform poorly and use methods that are intellectually unsatisfying. Some of the best computer vision systems use histograms of oriented gradients as "visual words" and model the spatial distribution of these elements using a crude spatial pyramid. Such methods can recognize objects correctly without knowing exactly where they are – an ability that is used to diagnose brain damage in humans. The best artifical neural networks [3, 8, 4] use hand-coded weight-sharing schemes to reduce the number of free parameters and they achieve local translational invariance by subsampling the activities of local pools of translated replicas of the same kernel. This method of dealing with the changes in images caused by changes in viewpoint is much better than no method at all, but it is clearly incapable of dealing with recognition tasks, such as facial identity recognition, that require knowledge of the precise spatial relationships between high-level parts like a nose and a mouth. After several stages of subsampling in a convolutional net, high-level features have a lot of uncertainty in their poses. This is generally regarded as a desireable property because it amounts to invariance to pose over some limited range, but it makes it impossible to compute precise spatial relationships.

This paper argues that convolutional neural networks are misguided in what they are trying to achieve. Instead of aiming for viewpoint invariance in the

activities of "neurons" that use a single scalar output to summarize the activities of a local pool of replicated feature detectors, artifical neural networks should use local "capsules" that perform some quite complicated internal computations on their inputs and then encapsulate the results of these computations into a small vector of highly informative outputs. Each capsule learns to recognize an implicitly defined visual entity over a limited domain of viewing conditions and deformations and it outputs both the probability that the entity is present within its limited domain and a set of "instantiation parameters" that may include the precise pose, lighting and deformation of the visual entity relative to an implicitly defined canonical version of that entity. When the capsule is working properly, the probability of the visual entity being present is locally invariant – it does not change as the entity moves over the manifold of possible appearances within the limited domain covered by the capsule. The instantiation parameters, however, are equivariant – as the entity moves over the appearance manifold, the instantiation parameters change by a corresponding amount because they are representing the intrinsic coordinates of the entity on the appearance manifold.

One of the major advantages of capsules that output explicit instantiation parameters is that they provide a simple way to recognize wholes by recognizing their parts. If a capsule can learn to output the pose of its visual entity in a vector that is linearly related to the "natural" representations of pose used in computer graphics, there is a simple and highly selective test for whether the visual entities represented by two active capsules, A and B, have the right spatial relationship to activate a higher-level capsule, C. Suppose that the pose outputs of capsule A are represented by a matrix, T_A, that specifies the coordinate transform between the canonical visual entity of A and the actual instantiation of that entity found by capsule A. If we multiply T_A by the part-whole coordinate transform T_{AC} that relates the canonical visual entity of A to the canonical visual entity of C, we get a prediction for T_C. Similarly, we can use T_B and T_{BC} to get another prediction. If these predictions are a good match, the instantiations found by capsules A and B are in the right spatial relationship to activate capsule C and the average of the predictions tells us how the larger visual entity represented by C is transformed relative to the canonical visual entity of C. If, for example, A represents a mouth and B represents a nose, they can each make a prediction for the pose of the face. If these predictions agree, the mouth and nose must be in the right spatial relationship to form a face.

2 Capsules and the multiple instances problem

A capsule has real-valued outputs that represent the pose of its visual entity. Using multiple real-values is the natural way to represent pose information and it is much more efficient than using coarse coding[2], but it comes at a price: The only thing that binds the values together is the fact that they are the pose outputs of the same capsule so it is not possible for a capsule to represent more than one instance of its visual entity at the same time.

It might seem that the inability to allow several, simultaneous instances of the same visual entity in the same limited domain is a serious weakness and indeed it is. It can be ameliorated by making each of the lowest-level capsules operate over a very limited region of the pose space and only allowing larger regions for more complex visual entities that are much less densely distributed. But however small the region, it will always be possible to confound the system by putting two instances of the same visual entity with slightly different poses in the same region. It is therefore comforting, at least to scientists, that there is evidence that the human visual system may suffer from this weakness – a phenomenon known as crowding [7].

3 Learning the first level of capsules

Once pixel intensities have been converted into the outputs of a set of active, first-level capsules each of which produces an explicit representation of the pose of its visual entity, it is relatively easy to see how larger and more complex visual entities can be recognized by using agreements of the poses predicted by active, lower-level capsules. But where do the first-level capsules come from? How can an artificial neural network learn to convert the language of pixel intensities to the language of pose parameters? That is the question addressed by this paper and it turns out that there is a surprisingly simple answer which we call a "transforming autoencoder". We explain the idea using simple 2-D images and capsules whose only pose outputs are an x and a y position. We generalize to more complicated poses later.

Consider the feedforward neural network shown in figure 1. The network is deterministic and, once it has been learned, it takes as inputs an image and

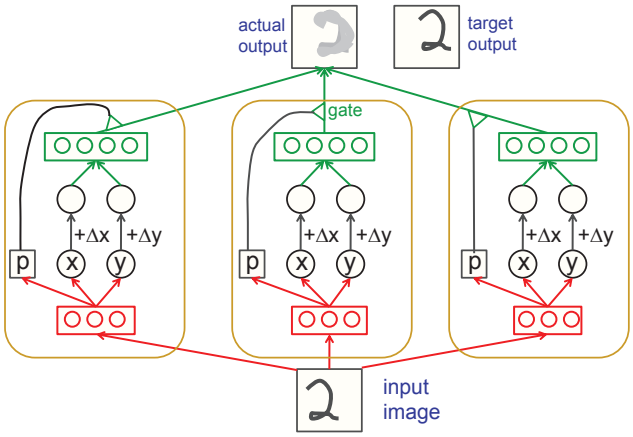

Fig. 1. Three capsules of a transforming autoencoder that models translations. Each capsule in the figure has 3 recognition units and 4 generation units.

desired shifts, Δx and Δy, and it outputs the shifted image. The network is composed of a number of separate capsules that only interact at the final layer when they cooperate to produce the desired shifted image. Each capsule has its own logistic "recognition units" that act as a hidden layer for computing three numbers, x, y, and p, that are the outputs that the capsule will send to higher levels of the vision system. p is the probability that the capsule's visual entity is present in the input image. The capsule also has its own "generation units" that are used for computing the capsule's contribution to the transformed image. The inputs to the generation units are $x + \Delta x$ and $y + \Delta y$, and the contributions that the capsule's generation units make to the output image are multiplied by p, so inactive capsules have no effect.

For the transforming autoencoder to produce the correct output image, it is essential that the x and y values computed by each active capsule should correspond to the actual x and y position of its visual entity. We do not need to know in advance what this visual entity will be, nor do we need to know the origin of the coordinate system that it uses: Whatever the visual entity and whatever the origin of its coordinate system, the location of the contribution it

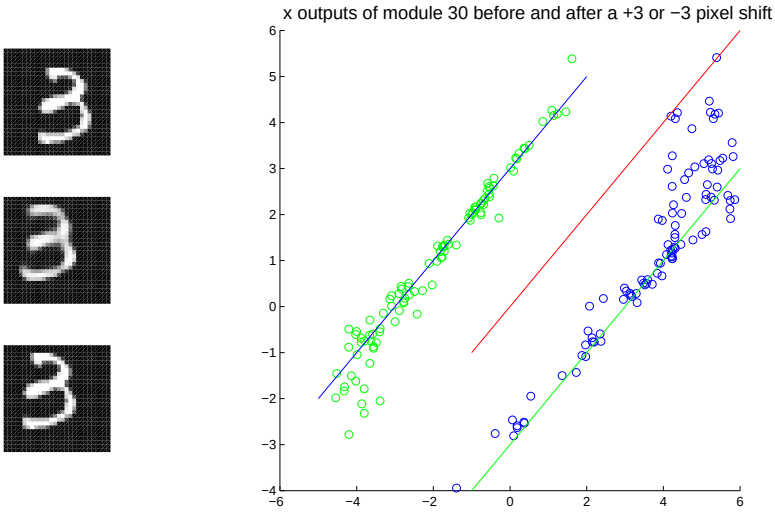

Fig. 2. Left: After learning, a transforming autoencoder can translate an input image (top) to produce an output image (middle) that has the specified translation. The correct output is shown at the bottom. **Right:** A scatterplot in which the vertical axis represents the x-coordinate that one of the capsules outputs for each digit image and the horizontal axis represents the x-coordinate that the capsule outputs if that image is shifted by +3 or −3 pixels in the x direction. If the original image is already near the limit of the x positions that the capsule can represent, shifting further in that direction causes the capsule to produce the wrong answer, but this does not matter if the capsule sets its probability to 0 for data outside its domain of competence.

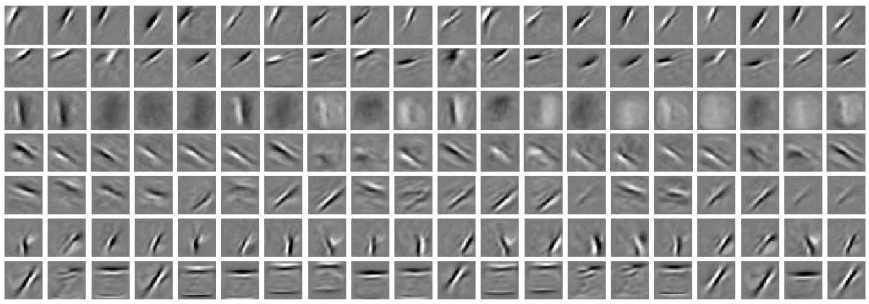

Fig. 3. The outgoing weights of all 20 generative units for 7 of the capsules. The bottom row is a capsule that specializes in representing the location of the bounding box that was used to normalize the digit image.

makes to the output image, assuming $p = 1$, is determined solely by $x + \Delta x$ and $y + \Delta y$.

As a simple demonstration of the efficacy of the transforming autoencoder, we trained a network with 30 capsules each of which had 10 recognition units and 20 generation units. Each capsule sees the whole of an MNIST digit image. Both the input and the output images are shifted randomly by -2, -1, 0, +1, or +2 pixels in the x and y directions and the transforming autoencoder is given the resulting Δx and Δy as an additional input. Figure 2 shows that the network learns to produce the appropriately shifted image and that the capsules do indeed output x and y values that shift in just the right way when the input image is shifted. Figure 3 shows that the capsules learn generative units with projective fields that are highly localized. The receptive fields of the recognition units are noisier and somewhat less localized.

3.1 More complex 2-d transformations

If each capsule is given 9 real-valued outputs that are treated as a 3×3 matrix A, a transforming autoencoder can be trained to predict a full 2-D affine transformation (translation, rotation, scaling and shearing). A known transformation matrix T is applied to the output of the capsule A to get the matrix TA. The elements of TA are then used as the inputs to the generation units when predicting the target output image.

3.2 Modeling changes in 3-D viewpoint

A major potential advantage of using matrix multiplies to model the effects of viewpoint is that it should make it far easier to cope with 3-D. Our preliminary experiments (see figure 6) used computer graphics to generate stereo images of various types of car from many different viewpoints. The transforming autoencoder consisted of 900 capsules, each with two layers (32 then 64) of rectified

Fig. 4. Full affine transformations using a transforming autoencoder with 25 capsules each of which has 40 recognition units and 40 generation units. **Top row:** Input images. **Middle row:** Output images **Bottom row:** Correct output images.

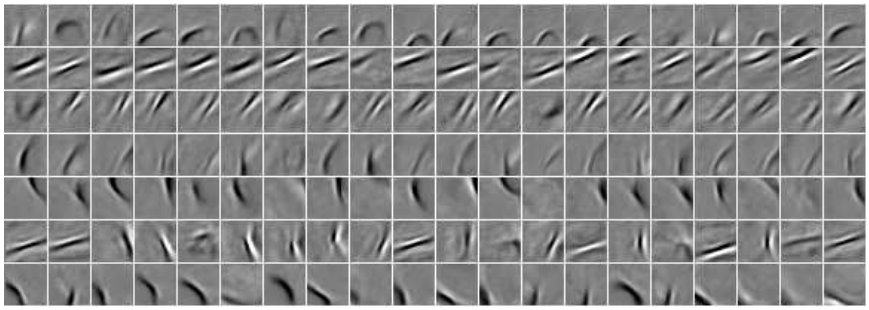

Fig. 5. The output weights of the first 20 generation units of the first 7 capsules for the model that produced the reconstructions in figure 4.

linear recognition units [6]. The capsules had 11x11 pixel receptive fields which were arranged on a 30x30 grid over the 96x96 image, with a stride of 3 pixels between neighbouring capsules. Each capsule produced from its layer of 64 recognition units a 3x3 matrix representation of the 3-D orientation of the feature that it was tuned to detect, as well as a probability that its implicitly defined feature was present. This 3x3 matrix was then multiplied by the real transformation matrix between the source and target images, and the result was fed into the capsule's single layer of 128 generative rectified linear units. The generation unit activities were multiplied by the capsule's "feature presence" probability and the result was used to increment the intensities in a 22x22 patch of the reconstructed image centered at the center of the capsule's 11x11 receptive field. Since the data consisted of stereo pairs of images, each capsule had to look at an 11x11 patch in both members of the stereo pair, as well as reconstructing a 22x22 patch in both members.

4 Discussion

From a pure machine learning perspective, providing the network with additional external inputs that specify the way in which the image has been transformed may appear unnecessary because this information could, in principle,

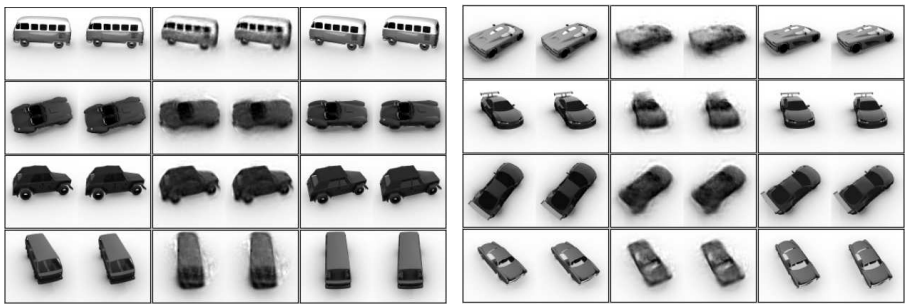

Fig. 6. Left: Input, output and target stereo-pairs for training data. **Right:** Input, output and target stereo-pairs for car models not seen during training.

be computed from the two images [5]. However, this information is often readily available and it makes the learning much easier, so it is silly not to use it. Eye-movements, for example, cause translations of the image on the retina, and visual cortex receives precise information about eye-movements.

A capsule bears some resemblance to a local pool of units in a convolutional neural network, because many of the recognition units have quite similar receptive fields in slightly different positions. There is a very important difference, however, in the way in which the outputs of all the recognition units are encapsulated for use by higher levels. In a convolutional pool, the combined output after subsampling is typically the scalar activity of the most active unit in the pool [9]. Even if the location of this unit is used when creating the reconstruction required for unsupervised learning, it is not used by higher levels [4] because the aim of a convolutional net is to make the activities translation invariant. Also, even if the location that is used for reconstruction were to be passed to higher levels, it would only have integer-valued coordinates. A capsule makes much better use of the outputs of the recognition units by using them to compute precise position coordinates which are accurate to a small fraction of a pixel. In this respect it resembles a steerable filter [1], but unlike most steerable filters it learns the receptive fields of the recognition units to optimize the accuracy of the computed coordinates and it also learns what visual entity to represent. Replicated copies of exactly the same weight kernel are far from optimal for extracting the pose of a visual entity over a limited domain, especially if the replication must cover scale and orientation as well as position.

Transforming autoencoders also have an intersting relationship to Kalman filters. The usual way to apply Kalman filters to data in which the dynamics is a non-linear function of the observations is to use an "extended" Kalman filter that linearizes the dynamics about the current operating point. This often works quite well but it is clearly a fudge. If we view the input and desired output images as temporally adjacent, the transforming autoencoder is a more principled way to apply a Kalman filter. The recognition units learn to map the input to a representation in which the dynamics really are linear. After the poses of the

capsules have been linearly transformed, the generation units map back to the observation domain. By measuring the error in the observation domain, we avoid the need to compute determinants that keep track of the extent to which errors have been compressed or expanded in moving between domains.

If we eliminate the extra input that gives the transforming autoencoder direct knowledge of the transformation, we can model the small transformations between adjacent time-frames as zero-mean Gaussian noise. This reduces the transforming autoencoder to a much less powerful learning method that is only able to find "slow features" that do not change much between successive images.

A transforming autoencoder can force the outputs of a capsule to represent *any* property of an image that we can manipulate in a known way. It is easy, for example, to scale up all of the pixel intensities. If a first-level capsule outputs a number that is first multiplied by the brightness scaling factor and then used to scale the outputs of its generation units when predicting the brightness-transformed output, this number will learn to represent brightness and will allow the capsule to disentangle the probability that an instance of its visual entity is present from the brightness of the instance. If the direction of lighting of a scene can be varied in a controlled way, a capsule can be forced to output two numbers representing this direction but only if the visual entity is complex enough to allow the lighting direction to be extracted from the activities of the recognition units.

References

1. W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.
2. G. E. Hinton. Shape representation in parallel systems. In *Proc. 7th International Joint Conference on Artificial Intelligence Vol 2*, 1981.
3. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
4. H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *ICML*, 2009.
5. R. Memisevic and G.E. Hinton. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Comp.*, 22:1473–1492, 2010.
6. V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. 27th International Conference on Machine Learning*, 2010.
7. D. G. Pelli and K. A. Tillman. The uncrowded window of object recognition. *Nature Neuroscience*, 11:1129 – 1135, 2008.
8. M. Ranzato, F.J. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'07)*. IEEE Press, 2007.
9. M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999.
10. R. S. Zemel and G. E. Hinton. Discovering and using the single viewpoint constraint. In D. S. Touretzky, editor, *Neural Information Processing Systems 3*, San Mateo, CA, 1991. Morgan Kauffman. Preprint.