Inception & ResNet: Same Training, Same Features

David G. McNeely-White, J. Ross Beveridge, and Bruce A. Draper

Colorado State University, Fort Collins, CO 80523, USA

Abstract. Deep convolutional neural networks (CNNs) are the dominant technology in computer vision today. Unfortunately, it's not clear how different from each other the best CNNs really are. This paper measures the similarity between two well-known CNNs, Inception and ResNet, in terms of the features they extract from images. We find that Inception's features can be well approximated as an affine transformation of ResNet's features and vice-versa.
The similarity between Inception and ResNet features is surprising. Convolutional neural networks learn complex non-linear features of images, and the architectural differences between the systems suggest that these functions should take different forms. Instead, they seem to have converged on similar solutions. This suggests that the selection of the training set may be more important than the selection of the convolutional architecture.

Keywords: ResNet, Inception, CNN, Feature Mapping

1 Introduction

Deep convolutional neural networks (CNNs) are the dominant technology in computer vision today, and much of the recent computer vision literature can be thought of as a competition to find the best vision architecture within the deep convolutional framework. Despite all this effort, however, it's not clear how different from each other the best architectures really are. The best systems correctly classify images at approximately the same rate. Are they extracting the same information from images, or just equally discriminating information?

This paper compares two sophisticated CNNs, Inception and ResNet, in terms of the features they extract from images. Both systems perform similarly on the ILSVRC2012 image recognition challenge [8]. ResNet-v2 152 [4] labels 78.9% of ILSVRC2012 test images correctly, while Inception-v4 [10] labels 80.2% correctly. On the surface, however, Inception and Resnet appear quite different. Inception [11] divides processing by scale, merges the results, and repeats. ResNet [3] has a simpler, single-scale processing unit with data pass-through connections. Inception produces 1,536 features per image, while ResNet produces 2,048. They also have disjoint pedigrees: Inception was developed by Google [5, 10–12]; ResNet was developed at Microsoft in 2016 [3, 4]. Nonetheless, we find that the features extracted by Inception are very similar

to the properties extracted by ResNet, in the sense that affine mappings of one predict the other. These mappings are accurate enough that mapped features can be used to label images without retraining the underlying classifier. This suggests that Inception and ResNet, despite their structural differences, extract essentially the same properties from images.

The finding that Inception and ResNet features are linked by affine transformations is surprising. Convolutional neural networks have revolutionized the field of computer vision precisely because they learn complex non-linear features of images, and the architectural differences between the systems suggest that these non-linear functions take different forms. Yet, Inception and ResNet seem to extract similar properties from images, since their features are (almost) linear transformations of each other. In essence, their training algorithms seem to hill-climb in totally different spaces and yet find similar local optima. This may explain why the two systems perform similarly, but we believe it also has broader implications. It suggests that the features extracted by Inception and ResNet are driven less by the details of convolutional architectures, and more by the content of the training images. If this is true, one would expect many sophisticated CNNs to perform at similar levels.

2 Related Work

Comparison of task performance (e.g. classification accuracy) on a common dataset is the most prevalent method for comparing deep learning systems [8]. Visualization techniques are used to peer into internal model representations, for example network inversion [2] and dissection [1]. Transfer learning can also be used to compare networks [6].

The closest work to this paper is Lenc et al. [7]. They also find affine mappings between CNNs (AlexNet, VGG-16 and ResNet-50). However, they learn mappings between convolutional layers and train their mappings using supervision in the form of image labels. We learn unsupervised mappings between final convolutional layers and fully-connected classifiers. As a result, the notion of feature equivalence established here is independent of manually-assigned image labels.

3 Methodology

Our goal is to compare convolutional neural networks, not in terms of their recognition rates, but in terms of how similar their features are. To do this, we describe CNNs as the composition of two functions. The first function, F(), extracts features from input using convolutional layers. In this paper the inputs are images, but in other domains they could be videos or audio signals or any other complex input. The second function, C(), is the classifier, usually implemented through fully-connected layers. Classifiers map feature vectors to labels. Thus a typical CNN is written as C(F()).

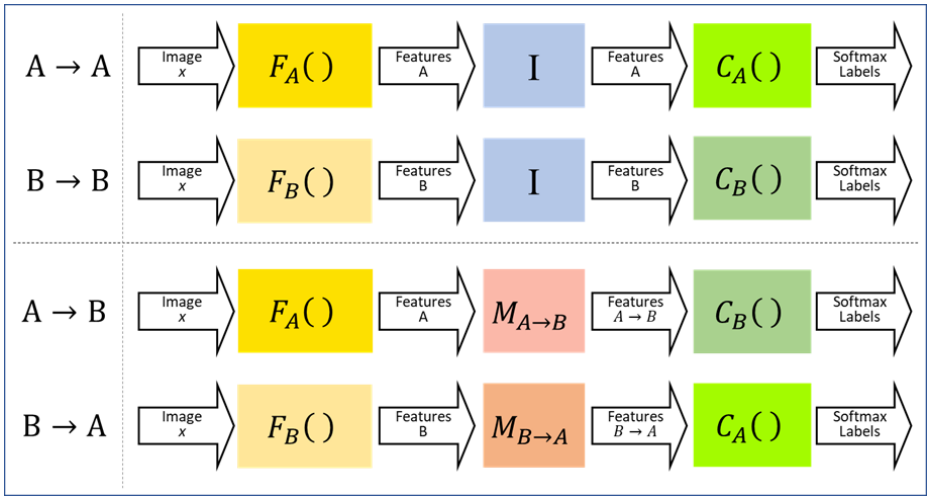

Fig. 1. Illustration of how a pair of standard CNNs can be used to create two alternative CNNs where the features from one are affine mapped to the classifier of another. The first two rows show CNNs A and B without alteration; the mapping from $F()$ to $C()$ is the identity mapping I. The next two illustrate the swapping of classifiers and the introduction of affine mappings $M_{A\to B}$ and $M_{B\to A}$.

Throughout the evaluation process, we never retrain or otherwise modify any of the classification functions C() or feature extraction functions F(). We do, however, fit affine matrices $M_{A\to B}$ and $M_{B\to A}$ which create linear mappings between F_A and F_B. In other words

$$\tilde{F}_A() = M_{B\to A}F_B() \qquad \text{and} \qquad \tilde{F}_B() = M_{A\to B}F_A()$$

where $\tilde{F}_A$ is the best approximation to F_A and $\tilde{F}_B$ is the best approximation to F_B across the training data set.

Our goal is to determine if $F_A()$ and $F_B()$ extract the same properties from the input. The key insight is that $C_A()$ was trained to classify samples based on the information in $F_A()$. If the same information is in $F_B()$, then the mapping $M_{B\to A}$ should preserve it, and $C_A()$ should be able to label images as well from $M_{B\to A}F_B()$ as from $F_A()$. If, on the other hand, $F_A()$ contains information not in $F_B()$, then that information will be missing in $M_{B\to A}F_B()$.

We therefore define a directional loss function between feature extractors. Let $Acc(C())$ be the accuracy of a classifier $C()$ across a test data set. Then

$$Loss_{A,B} = Acc(C_A(F_A())) - Acc(C_A(M_{B\to A}F_B()))$$

is the loss in discriminatory power that results from replacing A with B. For example, if system A classifies 90% of all inputs correctly using its own features $F_A()$, but only classifies 80% of samples using $\tilde{F}_A() = M_{A\to B}F_B()$, then $Loss_{A,B} = 10\%$, because 10% of the discriminatory power in $F_A()$ was missing

from $F_B()$. Note that if A and B are both good systems but extract and rely on different information, both $Loss_{A,B}$ and $Loss_{B,A}$ may be high. It is similar to a divergence in this sense.

It is possible for the loss function defined above to be negative (in which case we refer to it as a gain). It implies that system A's classifier, which was trained on system A's features, nonetheless performs better on the mapped version of system B's features. Since $C_A()$ is not retrained, the information gain cannot be in the form of a new property extracted by B but ignored by A, since the information would have been dropped in the mapping process (and A's classifier wouldn't know what to do with it anyway). Instead, it implies that some property detected by A is detected more robustly by B, so that $M_{B\to A}F_B()$ is a more reliable predictor of the property than $F_A()$ is.

The experiments in this paper use the Inception-v4 network found in the Tensorflow-Slim GitHub repository [9] and described in [10], and the ResNet-v2 network found in the same repository and described in [4]. These networks were trained "internally at Google" using identical preprocessing on all 1.1 million ILSVRC2012 training samples, as described in Szegedy et al. [10]. After loading these pretrained parameters, we internally validated each network's top-1 single-crop classification accuracy on all 50k ILSVRC2012 validation set samples. This internal validation yielded 78.0% accuracy for ResNet, as opposed to the 78.9% reported in He et al. [4]. Our Inception-v4 model achieved 80.1%, as opposed to the 80.2% reported by Szegedy et al. [10].

Given source features from Inception $F_I() \in \mathbb{R}^{1536}$ and target features from ResNet $F_R() \in \mathbb{R}^{2048}$, we solve for an affine mapping. Specifically, we define the affine mapping as:

$$\tilde{F}_R(T) = M_{I\to R}\begin{bmatrix} F_I(T) \\ \mathbf{1} \end{bmatrix} \tag{1}$$

The weights in the affine transformation are trained using the ILSVRC2012 1.1 million training images for T in Equation 1. The terms in the affine transformation are trained using gradient descent. The error function E is the euclidean distance between the unit-normalized true and predicted vectors, i.e.

$$E(x,y) = \left(\frac{x}{\|x\|} - \frac{y}{\|y\|}\right)^2$$

where x and y are feature vectors. We compute this function on random batches of predicted vectors $y \in \tilde{F}_R(T)$ and their corresponding true vectors $x \in F_R(T)$. This error function was chosen as it is similar to angular distance, but is not sensitive to computational issues surrounding zero vectors.

After training, features are once again generated as in the previous section. Using the ILSVRC2012 50,000 validation images V, we produce class predictions $C_R(M_{I\to R}F_I(V))$ for mapping $I \to R$ and similarly $C_I(M_{R\to I}F_R(V))$ for mapping $R \to I$. This is illustrated in Fig. 1. Evaluation is performed by simply measuring the amount of correctly classified samples for each mapping configuration.

4 Results

Section 3 presented the technical details of how we create mappings from one feature space to another. This section returns to the question of whether the features learned by Inception-v4 and ResNet-v2 are similar. We measure similarity over the validation images from the ILSVRC2012 image recognition challenge. We apply $F_I()$, Inception's convolutional network, to these images, producing $1,536$ features per image. We also apply $F_R()$, ResNet's convolution network, to the validation images, producing $2,048$ features per image. Finally, we apply the I→R and R→I mapping functions described above, thereby creating $\tilde{F}_I = M_{R\to I}F_R()$ and $\tilde{F}_R = M_{I\to R}F_I()$. By applying $C_I()$ to $F_I()$ and $\tilde{F}_I()$ and applying $C_R()$ to $F_R()$ and $\tilde{F}_R()$ we can measure the similarity between Inception's and ResNet's features.

Description	Mapping	Correct	Percent	Loss
Inception-v4	(I → I)	40,037	80.2	
ResNet-v2 152	(R → R)	39,022	78.0	
Inception-v4 to ResNet-v2 152	(I → R)	39,686	79.4	-1.4
ResNet-v2 152 to Inception-v4	(R → I)	37,866	75.7	4.5

Table 1. Comparison of mapped and unmapped network performance. The values indicate the number of correctly labelled samples out of 50,000 on the ILSVRC2012 validation dataset is shown along with the percent correct. For the bottom two with the affine mappings the loss defined in Sect. 3 is shown.

The central question of the paper is whether the features learned by Inception-v4 and ResNet-v2 are equivalent. Table 1 shows the experimental results for mapping Inception features onto ResNet features (I→R) and vice-versa (R→I). The Inception classifier correctly labels 40,037 validation images using its own features $F_I()$; using ResNet features mapped through R→I (i.e. $\tilde{F}_I()$) it correctly labels 37,866 images. This is a loss of 4.5%. In the other direction, the ResNet classifier correctly labels 39,022 validation images using its own features $F_R()$ and 39,686 images using Inception's features mapped through I→R (i.e. $\tilde{F}_R()$). This is a gain rather than loss of 1.4%.

This result strongly suggests that all the information in ResNet's 2,048 features is also contained in Inception's 1,536 features. Otherwise, I→R could not have learned a mapping with no loss. The result in the other direction is weaker. The performance of Inception's classifier drops by 4.5% when it is given the mapped version of ResNet's features. This suggests that Inception-v4, which is the higher performing of the two systems, may capture some information in its features that ResNet's features miss.

5 Conclusion

We looked for equivalence between ResNet and Inception features to see whether they capture different but roughly equally discriminative information, or whether they actually extract the same information, just using a different architecture and encoding to do it. The findings suggests that they are, in fact, extracting qualitatively the same features, while Inception does it slightly more robustly. We speculate that this may have implications for other convolutional nets as well. As nets grow more complex, there may be a law of diminishing returns if they all end up extracting similar features, but that may be what happens because that is what the training data supports.

References

1. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6541–6549 (2017)
2. Dosovitskiy, A., Brox, T.: Inverting visual representations with convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4829–4837 (2016)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
4. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European conference on computer vision, pp. 630–645. Springer (2016)
5. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
6. Kornblith, S., Shlens, J., Le, Q.V.: Do better imagenet models transfer better? arXiv preprint arXiv:1805.08974 (2018)
7. Lenc, K., Vedaldi, A.: Understanding image representations by measuring their equivariance and equivalence. International Journal of Computer Vision **127**(5), 456–476 (2019). DOI 10.1007/s11263-018-1098-y. URL https://doi.org/10.1007/s11263-018-1098-y
8. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
9. Silberman, N., Guadarrama, S.: Tensorflow-slim image classification model library (2016). URL https://github.com/tensorflow/models/tree/master/research/slim
10. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
11. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9 (2015)
12. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818–2826 (2016)