Inception and ResNet Features are (Almost) Equivalent

David McNeely-White, J. Ross Beveridge, Bruce A. Draper
Colorado State University, Fort Collins, CO 80523

Abstract

Deep convolutional neural networks (CNNs) are the dominant technology in computer vision today. Much of the recent computer vision literature can be thought of as a competition to find the best architecture for vision within the deep convolutional framework. Despite all the effort invested in developing sophisticated convolutional architectures, however, 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 properties they extract from images. We find that the properties extracted by Inception are very similar to the properties extracted by ResNet, in the sense that either feature set can be well approximated by an affine transformation of the other. In particular, we find evidence that the information extracted from images by ResNet is also extracted by Inception, and in some cases may be more robustly extracted by Inception. In the other direction, most but not all of the information extracted by Inception is also extracted by ResNet.

The similarity between Inception and ResNet features is surprising. Convolutional neural networks learn complex non-linear features of images, and the architectural differences between systems suggest that these non-linear functions should take different forms. Nonetheless, Inception and ResNet were trained on the same data set and seem to have learned to extract similar properties from images. In essence, their training algorithms hill-climb in totally different spaces, but find similar solutions. This suggests that for CNNs, the selection of the training set may be more important than the selection of the convolutional architecture.

Preprint submitted to Cognitive Systems Research *October 22, 2019*

Keywords: ResNet, Inception, CNN, Feature Evaluation, Feature Mapping

1. Introduction

Deep convolutional neural networks (CNNs) are the dominant technology in computer vision today. CNNs are the top performers in object recognition competitions [19], and have been applied to many other visual tasks including object detection and localization [4, 18], segmentation [16], pose estimation [5], gesture recognition [17], and visual saliency [14]. As a cognitive architecture, however, CNNs leave something to be desired, because they aren't so much a single architecture as they are a framework that can be used to instantiate many architectures. Inception [23], for example, is a CNN-based architecture that divides processing by scale, merges the results, and repeats. ResNet [6], on the other hand, has a simpler, single-scale processing unit with many more layers and a data pass-through between levels (an idea expanded on in DenseNet [8]). Indeed, much of the computer vision literature over the last five years can be thought of as a competition among labs to find the best architecture for vision within the deep convolutional framework.

Despite all the effort invested in developing convolutional architectures, it's not clear how different from each other the best ones really are. Inception was originally developed by Google in 2014, and refined over the next 2 years [23, 9, 24, 22]. ResNet was developed at Microsoft in 2016 and also subsequently refined [6, 7]. Yet despite the conceptual differences in their architectures and their disjoint pedigrees, they perform similarly on the ILSVRC2012 image recognition challenge [19]. ResNet-v2 152 [7] labels 78.9% of ILSVRC2012 test images correctly, while Inception-v4 [22] labels 80.2% correctly. More recent architectures such as PNASNet have improved on these numbers [15], but only slightly (82.9%). Indeed, much of the recent computer vision literature can be thought of as a competition among labs to find the best architecture for vision within the deep convolutional framework.

This paper addresses the question of how similar Inception and ResNet are to

each other, not in terms of the number of ILSVRC2012 test images they correctly label, but in terms of the properties they extract from images. Both Inception and ResNet use a convolutional architecture to extract features from images, followed by fully-connected classifiers to assign labels to images based on the extracted features. The architectures used by the two systems to extract features are different, as are the numbers of features extracted (Inception produces 1,536 features per image, while ResNet produces 2,048). We also know from network visualization techniques that the features learned by the early layers of Inception are qualitatively different from the features learned by the early layers of ResNet [1] (for a discussion, see the Related Work section below). Nonetheless, we find that the properties 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, exploit essentially the same properties of images. At the same time, a deeper analysis of our findings suggests that while they may extract the same properties, Inception appears to do it more robustly than ResNet, and may extract a few more properties.

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 solutions. This affine connection explains why the two systems perform similarly, and we believe it also has broader implications. It suggests that the features extracted by Inception and ResNet are driven less by the details of their 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

The most common method for comparing deep learning systems is black box evaluation. A task is found for which there is a common dataset, and networks are evaluated to see which produces the lower error rate. For the task of image labeling, ILSVRC2012 is the most widely used data set. Black box evaluations are appropriate when the goal is to pick a system to minimize error rates, but they provide little information about the relative strengths and weaknesses of systems. When two systems perform similarly, as with Inception and ResNet, black box evaluations fail to disclose whether the two systems are doing essentially the same thing, or instead are very different and just happen to label roughly the same number of images correctly.

Unlike black box techniques, CNN visualization techniques were developed to reveal what image properties a CNN responds to. In activation maximization, synthetic images are created which maximally activate a particular neuron, often revealing interpretable patterns [3, 21]. Using an inverted CNN structure, deconvolution can be used to find patterns in an image that a particular neuron responds to [25]. Network inversion allows reconstruction of input images from intermediate layer-wise activations, revealing which areas in an image are salient to a feature [2]. Not all neurons respond to easily recognizable visual patterns, however. Dissection addresses this by correlating neurons or layers with semantic concepts including textures, materials, and scenes. This last technique was used to compare Inception-v1 and ResNet-v2 152 features, and concluded that ResNet features discriminate more semantic concepts within a data set than Inception-v1 features do [1]. This is somewhat at odds with the findings in this paper, but the analysis was done on an older version of Inception (Inception-v1, not Inception-v4), and analyzed individual features rather than evaluating all features as a set.

Initialization-based comparisons are appropriate for comparing CNNs with the same architecture that were trained for different tasks. The idea is that if the tasks (and therefore the networks) are similar, one task should train more

quickly and accurately when initialized using the trained weights from the other task. In essence, similarity is measured by the benefits of a warm start, as in [11]. Unfortunately, this comparison technique is not applicable when comparing across architectures.

The most similar prior work to this paper is the paper by Lenc et al. [13]. As in this paper, Lenc et al. learn affine transformations between CNNs trained on ILSVRC2012; in their case, the networks are AlexNet [12], VGG-16 [21] and ResNet-50 v1 [6]. Their emphasis is different, as they train mappings between convolutional layers, sometimes requiring interpolation, whereas our transformations map between the output of the final convolutional layer of one network and the fully-connected classification layer of the other. More significantly, Lenc et al. used supervision in the form of image labels to train the mappings. This effectively creates newly-trained networks. In this paper, we train affine transformations to predict one feature set from the other, without supervision and independent of any semantic image labels.

3. Background

This section gives the background information about Inception-v4 and ResNet-v2 necessary to fully understand this paper. Nothing in this section is novel, but it is included for completeness. Readers who are already familiar with Inception and ResNet may choose to skip to Sect. 4.

3.1. ResNet

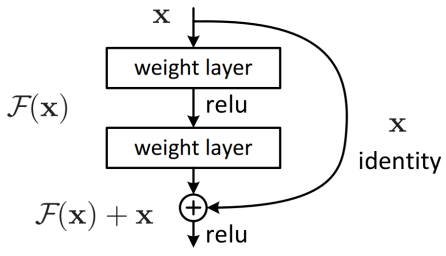

Figure 1: ResNet-v1 shortcut connections

In 2015, He, et al. introduced ResNet and the residual block (Fig. 3.1), consisting of two convolutional layers and a non-parameterized shortcut connection which passes the previous block's output to the next block, unmodified [6]. This provided a leap in state-of-the-art performance on the ILSVRC2012 challenge using a 152-layer ResNet, and established the property that more layers always result in higher recognition accuracy. Continued improvements can be demonstrated with even $1,000$ layers—a previously unattainable feat [6]. Following this result, they modify the residual block so that ReLU activation is not applied to the shortcut connection, providing further improvement with ResNet-v2 in 2016 [7]. This simple structural feature has made its way into many other deep CNNs, providing broad success (e.g. DenseNet [8], Inception-ResNet-v2 [22]). ResNet-v2 152 labels 78.9% of samples correctly using a single model and 320x320 single-crop on the ILSVRC2012 challenge's 50k validation samples, remaining a top-performer today.

3.2. Inception

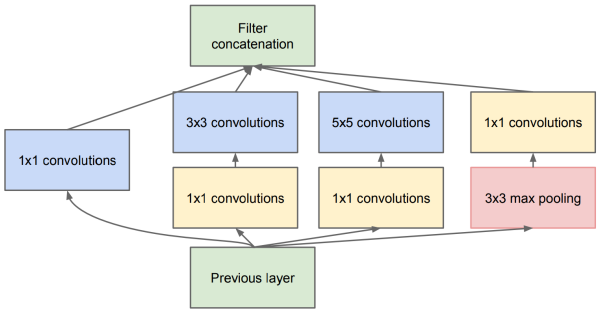

Figure 2: Inception-v1 module

Szegedy et al. took a different approach, beginning with a core architecture for addressing scale variation, known as the Inception module (Fig. 3.2) [23]. These modules use a combination of convolutional layers at different scales and max pooling computed on the same input and concatenated together. Many of these modules are stacked together to create Inception. While the modular architecture was never abandoned, numerous refinements were made to these

modules as the authors optimized for classification performance, training time, and computational footprint [9, 24, 22]. Though no individual architectural feature is responsible for Inception's performance, notable strategies include the use of 1x1 convolutions (a form of learned pooling for dimension reduction), factorizing nxn convolutional layers into stacked nx1 and 1xn layers (a computational footprint reduction), and batch normalization (a technique for reducing covariance shift). The most recent Inception variant (Inception-v4) is still a top-performer on the ILSVRC2012 challenge dataset, correctly labelling 80.2% of ILSVRC2012 validation samples using a single model and 299x299 single-crop [22].

4. Methodology

Our goal is to compare convolutional neural networks, not in terms of their recognition rates, but in terms of similarity of the features used by the final classification layers to label images. 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 $C(F())$.

Since our goal is to compare CNNs, we assume two CNNs $A = C_A\left(F_A()\right)$ and $B = C_B\left(F_B()\right)$. 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 $M_{A\rightarrow B}$ and $M_{B\rightarrow A}$ which create linear mappings between F_A and F_B. In other words

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

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.

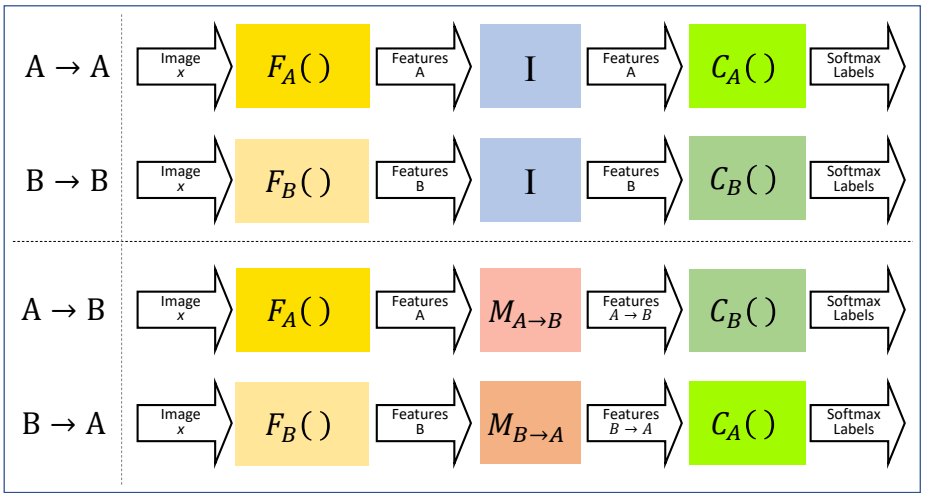

Figure 3: Illustration of how a pair of standard CNNs can be used to create two alternative CNNs where the features from one are affine mapped to the classifier of another. The notation used in this figure, in particular $F()$ and $C()$ to capture the mapping from input to the feature space and then feature space to final classification softmax is further described below. 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 $M_{B\to A}$.

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 $F_B()$. Similarly

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

is the information lost by replacing B with A. 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). This happens when $Acc(C_A(F_A()))$ is less than $Acc(C_A(M_{A\to B}F_B()))$, which is an interesting situation. 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.

4.1. Features

The experiments in this paper use the Inception-v4 network found in the TensorFlow-Slim GitHub repository [20] and described in [22], and the ResNet-v2 network found in the same repository and described in [7]. These networks were trained "internally at Google" using identical preprocessing on all 1.1 million ILSVRC2012 training samples, as described in Szegedy et al. [22]. 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. [7]. Our Inception-v4 model achieved 80.1%, as opposed to the 80.2% reported by Szegedy et al. [22].

4.2. Fitting Affine Maps

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:

$$\tilde{F}_R(T) = M'_{I \to R} F_I(T) + \mathbf{b}_{I \to R} \qquad \text{or} \qquad \tilde{F}_R(T) = M_{I \to R} \begin{bmatrix} F_I(T) \\ \mathbf{1} \end{bmatrix} \tag{1}$$

Note for clarity Equation 1 shows two alternative equivalent forms. On the left the offsets $\mathbf{b}_{I \to R}$ are shown explicitly. On the right is the compact augmented form using a single affine matrix $M_{I \to R}$. The matrix $M'_{I \to R}$ is embedded in the upper left portion of $M_{I \to R}$. An extra column is added containing $\mathbf{b}_{I \to R}$. An extra row is added consisting of all zeros followed by a single 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[1]. 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. This error function was chosen as it is similar to angular distance, but is not sensitive to computational issues surrounding zero vectors. Internally, we find this error function to produce higher-performing mappings than Euclidean distance of non-normalized true and predicted vectors[2]. We compute this function on random batches ($n = 2048$) of predicted vectors $y \in \tilde{F}_R(T)$ and their corresponding true vectors $x \in F_R(T)$ over 100k iterations. This training procedure is also computed in the other direction, $M_{R \to I}$. See Fig. 4 for error computed over each batch to illustrate training completeness.

[1] Specifically, the Adam [10] algorithm is used, initialized with a learning rate of 0.1 and decayed exponentially (at a rate of $1 - 1 \times 10^{-5}$ per iteration). Remaining Adam parameters are left at TensorFlow defaults ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-8}$).

[2] Our intuition is to incentivize the mapping to reproduce the *pattern* of the true vector, rather than it's absolute position.

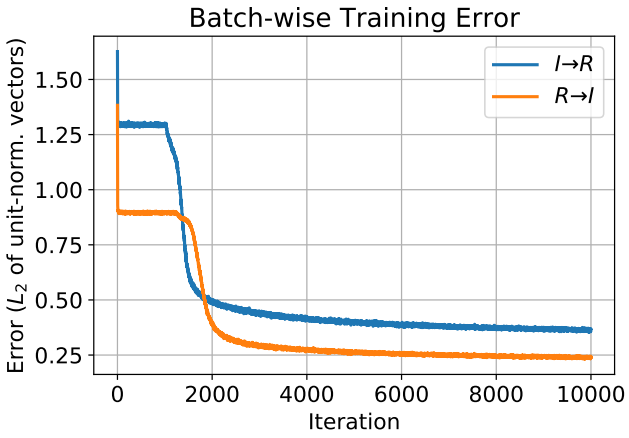

Figure 4: Training error calculated for the first 10k training iterations. Each batch consists of 2048 training samples, randomly sampled from the ILSVRC2012 training dataset.

4.3. Evaluating Mappings

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→R and similarly $C_I(M_{R\to I}F_R(V))$ for mapping R→I. This is illustrated in Fig. 3. We chiefly use these classifications for evaluating the performance of the mapping. A first order evaluation can be performed by simply measuring the number of correctly classified samples for each mapping configuration. Second, we can examine those correctly-classified mapped samples to those classified correctly by either ResNet or Inception for evaluation of mapping performance in multiple aspects. Evaluation is performed by simply measuring the number of correctly classified samples for each mapping configuration.

5. Results

Section 4 explains 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. 4 is shown.

5.1. Black Box Analysis

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 a 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 involves

some loss. In particular, the performance of Inception's classifier drops by 4.5% when it is given the mapped version of ResNet's features. The most important conclusion to draw is that much of the information required by Inception's classifier is present in the ResNet features. Secondarily, this result also 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.

5.2. Detailed Analysis

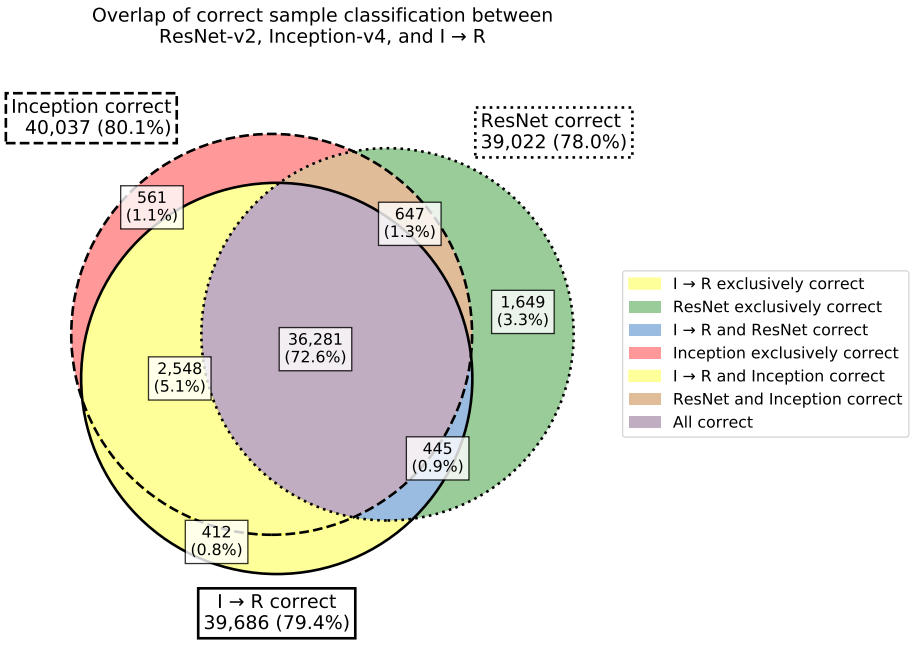

Figure 5: Colored areas represent samples which were labeled correctly by ResNet (dotted circle), Inception (dashed circle), and ResNet's classifier using Inception's features (solid circle). Each of the three circles represents one of these three variants correctly classified samples. I→R is shown with the solid circle, Inception dashed, and ResNet dotted. The innermost region (mauve) is scaled down by a factor of 10 for readability; all other areas are to scale.

For further analysis, we divide the set of correctly labeled samples into three categories: those correctly labeled by ResNet, those labeled correctly by Inception, and those labeled correctly by ResNet's classifier based on Inception's mapped features (i.e. $C_R(M_{I\to R}F_I())$. These sets are shown in Fig. 5. By

overlapping each category, we can observe which correct classifications are either common or novel to each network. The mauve innermost region represents those images that are correctly classified by Inception, ResNet and the I→R mapping. This is reflective of the general performance success in Table 1. The yellow region, however, represents images that are correctly labeled by Inception and mislabeled by ResNet, but that are somehow correctly classified by ResNet's classifier when given I→R mapped Inception features. Since we did not retrain ResNet's classifier, it presumably cannot take advantage of information that was not present in ResNet's features. Yet for the 10,978 images that ResNet misclassifies, replacing ResNet's features with mapped Inception features "fixes" 2,960 of them so that they get the right label. This suggests that ResNet and Inception are extracting qualitatively the same image features (because the ResNet classifier exploits them), but that Inception's features are somehow more robust.

6. Conclusion

Linear mappings can be constructed between Inception and ResNet features with relatively little penalty in classification accuracy. The mapping from Inception features to ResNet features creates no drop in recognition accuracy, in fact there is a gain. This suggests that all the information contained in ResNet features is also contained in Inception features, and that Inception may extract information of value to the ResNet classifier slightly better than ResNet itself does. This hypothesis is consistent with the observation that mapping from ResNet features to Inception features creates a small but significant (4.5%) drop in performance. This could either be because Inception captures a few image features that are not captured by ResNet, or because (as in the other mapping) Inception captures the same features only "better".

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 suggest that the two networks are, in fact, extracting qualitatively the same features. 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., and Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6541–6549.

[2] Dosovitskiy, A. and Brox, T. (2016). Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4829–4837.

[3] Erhan, D., Bengio, Y., Courville, A., and Vincent, P. (2009). Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1.

[4] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

[5] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

[6] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[7] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

[8] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

[9] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

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

[11] Kornblith, S., Shlens, J., and Le, Q. V. (2018). Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*.

[12] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[13] Lenc, K. and Vedaldi, A. (2019). Understanding image representations by measuring their equivariance and equivalence. *International Journal of Computer Vision*, 127(5):456–476.

[14] Li, G. and Yu, Y. (2016). Visual saliency detection based on multiscale deep cnn features. *IEEE Transactions on Image Processing*, 25(11):5012–5024.

[15] Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. (2018). Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34.

[16] Liu, F., Lin, G., and Shen, C. (2015). Crf learning with cnn features for image segmentation. *Pattern Recognition*, 48(10):2983–2992.

[17] Narayana, P., Beveridge, R., and Draper, B. A. (2018). Gesture recognition: Focus on the hands. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5235–5244.

[18] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

[19] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

[20] Silberman, N. and Guadarrama, S. (2016). Tensorflow-slim image classification model library.

[21] Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

[22] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[23] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[24] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

[25] Zeiler, M. D., Taylor, G. W., Fergus, R., et al. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, number 2, page 6.