Neural Networks 137 (2021) 63–74

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Words as a window: Using word embeddings to explore the learned representations of Convolutional Neural Networks

Dhanush Dharmaretnam[b], Chris Foster[b], Alona Fyshe[a,*]

[a] *University of Alberta, Department of Computing Science & Department of Psychology, 116 St. and 85 Ave., Edmonton, Alberta, Canada*
[b] *University of Victoria, Department of Computer Science, 3800 Finnerty Road, Victoria, British Columbia, Canada*

A R T I C L E I N F O

Article history:
Received 20 April 2020
Received in revised form 1 November 2020
Accepted 7 December 2020
Available online 22 January 2021

Keywords:
Deep learning
Interpretation
Convolutional Neural Nets
CNNs
Word vectors
Distributional Semantics

A B S T R A C T

As deep neural net architectures minimize loss, they accumulate information in a hierarchy of learned representations that ultimately serve the network's final goal. Different architectures tackle this problem in slightly different ways, but all create intermediate representational spaces built to inform their final prediction. Here we show that very different neural networks trained on two very different tasks build knowledge representations that display similar underlying patterns. Namely, we show that the representational spaces of several distributional semantic models bear a remarkable resemblance to several Convolutional Neural Network (CNN) architectures (trained for image classification). We use this information to explore the network behavior of CNNs (1) in pretrained models, (2) during training, and (3) during adversarial attacks. We use these findings to motivate several applications aimed at improving future research on CNNs. Our work illustrates the power of using one model to explore another, gives new insights into the function of CNN models, and provides a framework for others to perform similar analyses when developing new architectures. We show that one neural network model can provide a window into understanding another.

1. Introduction

Convolutional Neural Networks (CNNs) extract information in the pixels of images by applying a hierarchy of learned functions, and can even outperform humans for some tasks (Karpathy, 2014). While CNNs have become incredibly accurate, they have also become deeper and more complex, making it more difficult to understand why they work, and how they fail. It is not always clear *why* one CNN architecture outperforms another, and when we design new networks, the architectural decisions and innovations can be ad hoc, mostly verified by trial and error. And when CNNs fail (as they do under adversarial attack) it can be difficult to determine where the network went wrong, or even that it went wrong.

In this paper, we present several new methods for studying CNNs, both pretrained and during training, as well as during an adversarial attack. We track the accumulation of information through the layers of a CNN, and offer insights into the function and performance of CNNs. We then describe how our study of CNN hidden representations could be used to operationalize the building of new CNN architectures, or build CNNs that are more robust to adversarial attack. The methodology is inspired by techniques originally developed to study the brain's semantic representations via neuroimaging data, and we have shown it to be useful for understanding CNNs (Dharmaretnam & Fyshe, 2018). Our contributions are:

(1) Evaluation using **5 times** more concepts than our previous work (Dharmaretnam & Fyshe, 2018), including experiments with 3 architectures (ResNet50, Inception-v3, and FractalNet), and two datasets (ImageNet, CIFAR-100).
(2) An exploration of the behavior of hidden layers during training that shows that different layers learn more during early and late training epochs.
(3) An application of the technique to describe hidden representations for adversarial images.

Each of these points illustrate how DS (Distributional Semantic) models can help us to understand CNNs, but also how we might use DS models to train better, more robust CNNs. We illustrate here a framework for understanding the behavior of CNNs in a variety of settings, and then illustrate how this technique could be used to imagine and test better architectures, and possibly protect against adversarial attacks.

2. Related work

Distributional models of word meaning (word embeddings) use patterns of word co-occurrence to estimate vector representations for words. These vectors have proven useful for a

* Corresponding author.
E-mail address: alona@ualberta.ca (A. Fyshe).

https://doi.org/10.1016/j.neunet.2020.12.009
0893-6080/© 2020 Elsevier Ltd. All rights reserved.

variety of Natural Language Processing tasks, and have been shown to correlate strongly to human judgments of word similarity (Bruni & Baroni, 2013; Hill, Reichart, & Korhonen, 2015), and behavioral norms (Hollis, Westbury, & Lefsrud, 2017). In fact, several DS models include both text and images to create one joint model (Anderson, Bruni, Bordignon, Poesio, & Baroni, 2013; Bruni & Baroni, 2013). However, the idea of using DS models to understand CNNs has been largely untouched.

Distributional models have been used in conjunction with CNNs in a variety of ways. CNNs have been trained to predict word vector dimensions as output instead of discrete classification (Frome, Corrado, & Shlens, 2013). Predicting into a space shared with the word vectors allows CNNs to predict for classes not seen during training (zero shot learning) (Lazaridou, Bruni, & Baroni, 2014; Socher, Ganjoo, & Sridhar, 2013). Previously, we used distributional semantic models to explore CNNs (Dharmaretnam & Fyshe, 2018). Compared to our previous work, this paper includes 5 times more concepts, new architectures (FractalNet), additional datasets (CIFAR-100), and additional explorations of the behavior of the hidden layers during training, and for adversarial examples.

The interpretation of CNNs has taken many forms, but most have relied on visual exploration of the images that most "excite" a neuron (Yosinski, Clune, Nguyen, Fuchs, & Lipson, 2015), or the areas of an image that most contribute to a prediction (Alber et al., 2019; Wagner et al., 2019; Zeiler & Fergus, 2014). Semantic parts (e.g. wheels, legs) have also been identified within CNN representations (Gonzalez-Garcia, Modolo, & Ferrari, 2018), another piece of evidence that CNNs capture semantic meaning. There have also been several variants of Canonical Correlation Analysis (CCA) proposed to project the hidden layers into a shared representational space with either the raw image pixels or the layers of another CNN (Morcos, Raghu, & Bengio, 2018; Raghu, Gilmer, Yosinski, & Sohl-Dickstein, 2017; Saini & Papalexakis, 2018).

Because CNNs can be so hard to interpret, some methods actually train additional models on top of the CNN representations to facilitate interpretation. Zhang, Yang, Ma, and Wu (2019) developed a method that uses decision trees to identify semantic parts of the object in the image, assign importance to each of the parts, and relate these parts to the representations within the CNN. Bologna (2019) developed a method which extracts rules from a CNN model, allowing for a more natural interpretation. The rules were generated by a Multi Layer Perceptron, and centroids of the generated rules are quite interpretable, thus assisting practitioners as they explore the origins of a particular CNN prediction.

In this work, we use an independent model trained on very different data (text) as a sort of "third-party" evaluation of the information that exists in the layers of the CNN. This allows us to move beyond the similarities that exist in image space (e.g. many animals are pictured in outdoor scenes) and instead correlate to another notion of semantics built from the usage of the word associated with the concept. In addition, our technique uses raw correlation, which avoids the overhead of training a new model, making our approach quick and easy to implement.

3. Methodology

Our experiments analyze three popular CNNs, two popular image classification datasets, and six DS models. We explore two networks pretrained on ImageNet (Deng et al., 2009): ResNet-50, and Inception-v3; and FractalNet, which we train on CIFAR-100 (Krizhevsky & Hinton, 2009).

3.1. Convolutional neural networks

ResNet-50. The ResNet architecture (He, Zhang, Ren, & Sun, 2015) was introduced as an entry to the 2015 Large Scale Visual Recognition Challenge (Russakovsky et al., 2015; Szegedy et al., 2015). ResNet introduces residual blocks, which contain residual connections that give each block's final layer access to the block's original input. This helps both with stability during training, and also with predictive accuracy. We study ResNet-50, the 50 layer variant that achieved a top-5 error rate of 5.25% on the ILSVRC 2015 test dataset. We studied the 49 activation layers spread across 16 residual blocks in this network.

Inception-v3. Inception-v3 is a variant of the original GoogLeNet architecture (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2015). It has 94 convolutional blocks followed by ReLU activations. The network has nine inception modules, which process the input with differently sized convolutional filters. The outputs of each branch within an inception module are concatenated before they are passed to the next inception layer. We studied both activation and filter concatenation layers (*mixed* layers in *Keras*, Chollet et al., 2015).

FractalNet. FractalNet is an interesting architecture trained with both deep and shallow connection paths made up of fractal expansions of the same base architecture (Larsson, Maire, & Shakhnarovich, 2016). Unlike ResNet (a network with comparable deep and shallow connection paths), the FractalNet architecture continues to improve with added depth (though with diminishing returns).

3.2. Distributional semantic models

We present several Distributional Semantic (DS) models for studying the semantic representations in CNNs. **SkipGram** is part of the Word2vec package, and the vectors we used are 300-dimensional, and are trained on the Google News dataset to predict context words given a central word (Mikolov, Chen, Corrado, & Dean, 2013). **Glove** is a regression-based model that incorporates both local and global co-occurrence information. This 300-dimensional model was trained on the English Wikipedia and Gigaword 5 corpora combined (Pennington, Socher, & Manning, 2014). The word vectors from **Elmo** are derived from a bi-directional language model (Peters et al., 2018). Elmo is designed to take context into account, but can also be used to create vectors for single words, which is what we do here. Elmo is trained on a random sample of Wikipedia and the common crawl, and we use an average of the three 512-dimensional output layers. **Lexvec** uses matrix factorization to compress a matrix of co-occurrence counts for words appearing together within a window of four words (Salle, Idiart, & Villavicencio, 2016). The Lexvec model has 300 dimensions and is trained on the common crawl. **Fasttext** is based on SkipGram, but operates on character n-grams, and thus is able to capture morphological information (Bojanowski, Grave, Joulin, & Mikolov, 2017). Fasttext is trained on a combination of Wikipedia 2017, the UMBC webbase corpus and data from . We use a model with 300 dimensions. The **Non-distributional** model is based on hand-crafted linguistic resources like WordNet (Fellbaum, 1998b) and FrameNet (Baker, Fillmore, & Lowe, 1998). These vectors are of very high dimension (171,839) because they are quite sparse. This is an interesting model to compare against because it is *not* built from a corpus (unlike every other model in this list) (Faruqui, Tsvetkov, Yogatama, Dyer, & Smith, 2015).

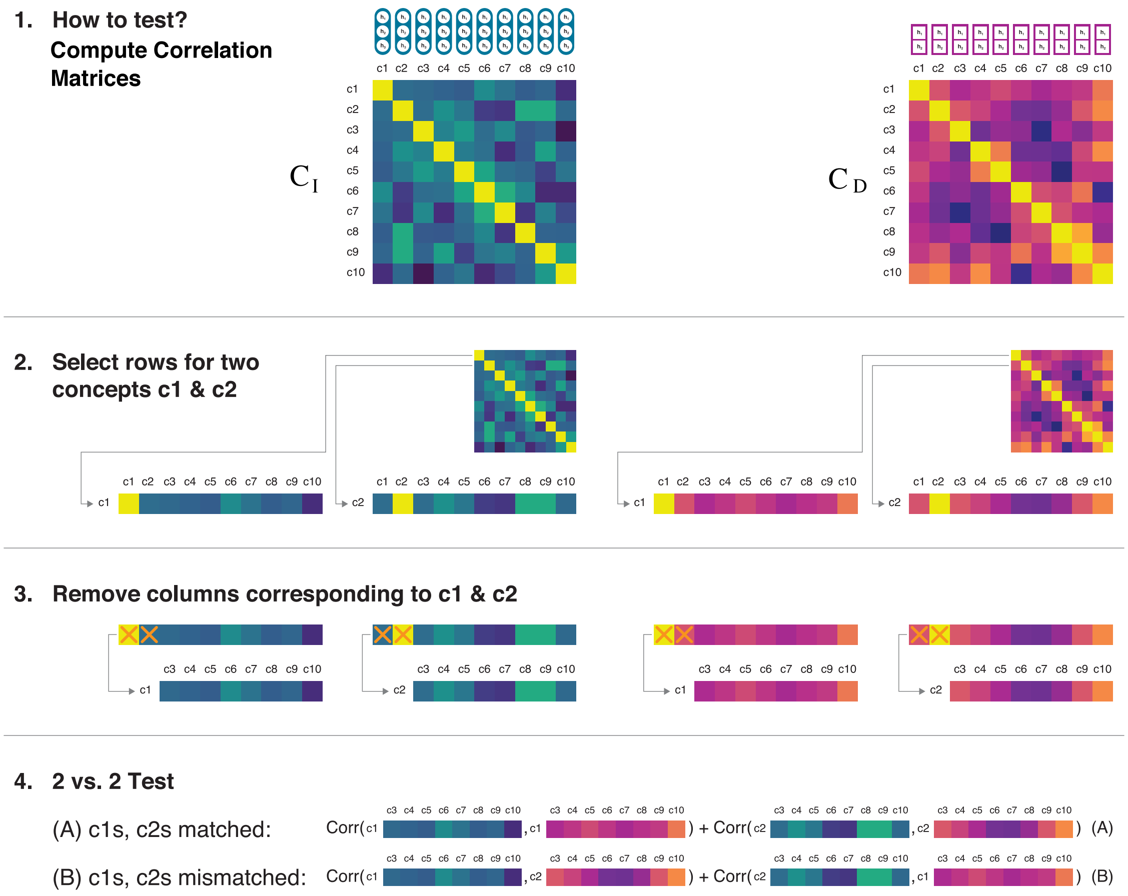

1. How to test?
Compute Correlation Matrices
C I
C D
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10
2. Select rows for two concepts c1 & c2
3. Remove columns corresponding to c1 & c2
4. 2 vs. 2 Test
(A) c1s, c2s matched:
(B) c1s, c2s mismatched:
Corr(c1 , c1) + Corr(c2 , c2) (A)
Corr(c1 , c2) + Corr(c2 , c1) (B)

Fig. 1. An overview of the 2 vs. 2 test, using an example with 10 concepts (c1…c10). **(1)** Using the representations for each concept (illustrated as vectors above the matrices), calculate the correlation of representations in Image and Distributional Semantics space. The blue matrix represents C_I, the correlation of CNN hidden representations computed from images of concepts (c1…c10). The pink matrix is C_D, the correlation of word embeddings for concepts (c1…c10). **(2)** An example of the 2 vs. 2 test for concepts c1 and c2. Select the rows of the correlation matrices C_I and C_D corresponding to c1 and c2. **(3)** Remove the columns corresponding to the self- and cross-correlation of c1 and c2. **(4)** Compare correlation of vectors when matching c1 and c2 vectors (Eq. (A)) to the correlation of mismatched vectors (Eq. (B)). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.3. Concept selection

Two of the CNNs used in our study are pretrained on ImageNet (Deng et al., 2009) which has 1000 labeled image classes organized to align with the WordNet hierarchy (Fellbaum, 1998a). ImageNet classes often are a list of synonymous concepts (e.g. class 286: "cougar, puma, catamount, mountain lion, painter, panther, Felis concolor"). For these cases, we use the vector of the first word that matches a word in the DS model, resulting in hundreds of vectors (e.g. 838 ImageNet classes for SkipGram alone, and 646 classes common to all six DS models). Matching word vectors to CIFAR-100 was more straightforward, as most of the classes are single words (or could be represented as a single word), and all 100 were present in the DS word lists.

3.4. Analyzing learned representations, and the 2 vs. 2 test

Following our previous work (Dharmaretnam & Fyshe, 2018), we randomly selected images for each of the w matched concepts from the ImageNet cross-validation dataset (Russakovsky et al., 2015). All images were rescaled to 224 × 224 for ResNet-50 and VGG-16, and 299 × 299 for Inception-v3. After resizing, the pixel values were mean normalized.

We then generated representations at *each layer*[1] of each CNN, recording the activation values for every image. At each layer, we produce a matrix $I \in \mathbb{R}^{w*k}$, where k is the dimension of the flattened CNN layer and w is the number of concepts. Each row in the matrix I represents the hidden representation of one image extracted from one layer of the CNN. Fig. 1 shows an overview of the 2 vs. 2 test, illustrated with 10 concepts. The vectors atop the two matrices correspond to rows in the matrix I.

We then compute the Pearson correlation of every concept with every other concept in I, resulting in a correlation matrix $C_I \in \mathbb{R}^{w*w}$ (blue matrix in Fig. 1). Thus, every element $C_I(i, j)$ represents the similarity of the hidden representation for an image depicting concept i and the hidden representation for an image depicting concept j. To ensure a fair comparison, we repeat the above steps with 5 randomly selected sets of w images, resulting in 5 CNN correlation matrices C_I per layer.

We extract the DS vectors for the same w concepts, resulting in a matrix $D \in \mathbb{R}^{w*n}$, where w is the number of concepts and n is the dimension of the DS vectors. We then compute the Pearson correlation of every word vector with every other word vector resulting in the correlation matrix $C_D \in \mathbb{R}^{w*w}$ (pink matrix in Fig. 1). $C_D(i, j)$ represents the similarity of word i with word j in the space defined by a specific DS model. Now we have matrices C_I and C_D which represent the similarity of concepts in CNN and DS space.

How can we compare the similarities between representations in C_I and C_D? We could just compute the correlation of the upper triangle of C_I and C_D, but this would obfuscate which concepts are best represented in each layer. Instead, we use the **2 vs. 2**

[1] A layer can be any node in the computation graph. Here, we focus on activation and concatenation layers.

test (Dharmaretnam & Fyshe, 2018), which allows us to explore the representations at the level of individual concepts. For the 2 vs. 2 test, we select the rows corresponding to two concepts (i and j) from both correlation matrices C_I and C_D, for a total of four vectors (Fig. 1 part 2). We then omit columns i and j from all four vectors, as they represent the cross- and self-correlation of the concepts, resulting in vectors with $w-2$ elements (Fig. 1 part 3). These vectors represent the correlation of the representations of concepts i and j to every other concept, both in CNN and in DS space. Let us rename the reduced vectors as $C_I^{(i)}$, $C_I^{(j)}$ from the CNN correlation matrix, and $C_D^{(i)}$, $C_D^{(j)}$ from the DS correlation matrix. In a 2 vs. 2 test, we calculate the correlation of the correlations, testing if the correctly matched pairs (i to i and j to j):

$$\text{corr}(C_I^{(i)}, C_D^{(i)}) + \text{corr}(C_I^{(j)}, C_D^{(j)}) \tag{A}$$

have greater correlation than the mismatched pairs (i to j and j to i):

$$\text{corr}(C_I^{(i)}, C_D^{(j)}) + \text{corr}(C_I^{(j)}, C_D^{(i)}) \tag{B}$$

This is also illustrated in Fig. 1 part 4. A 2 vs. 2 test is considered to pass if Eq. (A) is greater than Eq. (B). The test is repeated for all possible pairs of concepts in our set of w concepts. This results in $\binom{w}{2}$ tests. The 2 vs. 2 accuracy is the percentage of 2 vs. 2 tests passed, and chance is 50%. Note that this method is based entirely on correlation, and so can only detect linear relationships. Still we were impressed with the results this simple approach yielded. Future work might consider learning a (possibly non-linear and/or regularized) mapping to further explore the relationship between representational spaces.

The 2 vs. 2 tests were repeated for the 5 C_I matrices independently, and the scores were averaged to get a single score for a given layer of the CNN. This accounts for variability across images for a single concept in ImageNet (though in practice we found the average variation across the 5 matrices to be very small, e.g. 0.0064 for Inception-v3). The whole process is then repeated for each layer in each CNN, and for every DS model.

4. Studying adversarial examples

We were interested in studying when and how CNNs fail in the face of adversarial attack. For this, we explore adversarial images designed to mislead CNNs. We search through the hierarchical layers of the CNN to identify layers where misclassifications emerge, illustrating the vulnerabilities in CNN architectures, and providing a road map to debug and improve CNNs.

To study adversarial examples, we randomly selected 100 source concepts. For each source concept, we select 6 target concepts for a total of 600 adversarial targets. That is, we selected 100 “true” images, and altered them to become adversarial examples likely to be misclassified into 6 different target classes.

We were interested in how the similarity between source and target classes might affect the performance of an adversarial example. For this reason, we selected the six targets with varying similarity to the source concept, based on the correlation of the source and target word vectors. We select six target concepts including the most similar target concept, the least similar target concept, and four other target concepts spaced evenly between. For each source and target pair, we perform a targeted adversarial attack against Inception-v3 using v2.0.0 of Cleverhans (Papernot et al., 2017). Our attack algorithm is the Momentum Iterative Method with an L_∞ norm perturbation bound of $\epsilon = 0.3$ using a decay factor of $\mu = 1.0$ over 20 iterations (Dong et al., 2017). We then tested each of the 600 adversarial images and discarded four images that were not predicted to be the target class (failed attacks), resulting in 596 total adversarial images. Examples of the adversarial images are available in Appendix A.

4.1. The 1 vs. 2 test

Using the hidden representation generated by an adversarial image, we can generate a vector of correlations that represents the adversarial image in CNN space. This will essentially substitute in for the vector $C_I^{(i)}$ from the 2 vs. 2 test. We cannot simply use a row of C_I because the images we tested are altered to be adversarial. To ensure the validity of the correlation vector for the adversarial image, we need to calculate the correlation using *correctly* classified unaltered (i.e. non-adversarial) images. We randomly sampled 100 correctly classified images, and extracted hidden layer representations for each. This resulted in a matrix $I_{\text{correct}} \in \mathbb{R}^{100*k}$ where k is the dimension of the flattened CNN layer. Similarly, word vectors corresponding to the same 100 image labels were extracted from the DS model. Let us call this matrix $D_{\text{correct}} \in \mathbb{R}^{100*n}$ where n is the dimension of the DS model's word vectors. The 100 concepts represented in I_{correct} and D_{correct} never include the source or target concepts.

Then, using the hidden representations generated from adversarial images, we computed the correlations with all 100 concepts in I_{correct} resulting in the vector $i_{\text{adversarial}} \in \mathbb{R}^{100}$. Now $i_{\text{adversarial}}$ is essentially the equivalent of $C_I^{(i)}$ from the 2 vs. 2 test. The word vectors corresponding to the source and target classes were also extracted, and correlations computed with every concept in D_{correct} resulting in two vectors d_{source} and d_{target}, both of dimension 100. The vector $i_{\text{adversarial}}$ represents the correlation of concepts in CNN vector space, whereas d_{source} and d_{target} represent correlation of concepts in word vector space. We then check to see if the $i_{\text{adversarial}}$ is more correlated to d_{source} or d_{target}:

$$\text{corr}(i_{\text{adversarial}}, d_{\text{source}}) \overset{?}{>} \text{corr}(i_{\text{adversarial}}, d_{\text{target}}) \tag{1}$$

which tells us if the hidden representation of the adversarial image “looks” more like the source or target concept. This is the **1 vs. 2 test** (Dharmaretnam & Fyshe, 2018), and like the 2 vs. 2 test, chance accuracy is 50%. The test is repeated for all adversarial images, which gives us a measure of if the semantic information of the source class exists anywhere in the CNN's hierarchy during the processing of the adversarial image.

4.2. Testing for statistical significance

The 2 vs. 2 and 1 vs. 2 tests were designed to study the relationship between concepts in different vector spaces. The chance accuracy for both these tests is 50%, but we need to calculate a confidence interval around 50%, outside of which results are significant. We do this by running 1000 permutation tests (Wasserman, 2004), approximating p-values for the observed 2 vs. 2 and 1 vs. 2 accuracy, and correcting for multiple comparisons using Benjamini–Hochberg–Yekutieli (BHY) false discovery rate correction (Benjamini & Yekutieli, 2001). This is standard practice for estimating significance, while making very few statistical assumptions.

5. Results and discussion

Figs. 2 and 3 show the 2 vs. 2 accuracy (and standard deviation over the five independent samples) through the layers of both the convolutional networks (see also the annotated architecture diagrams in Appendices B and C). These results are similar to our previous findings (Dharmaretnam & Fyshe, 2018). However, here we tested more nodes in the CNN hierarchy, and were able to see that the increase in 2 vs. 2 accuracy through the CNN hierarchy is not monotonic, especially for Inception-v3. Rather there are peaks and troughs, discussed more below.

We also observed that the 2 vs. 2 accuracy of all six DS models follows a similar trend, though LexVec dominates, and Elmo

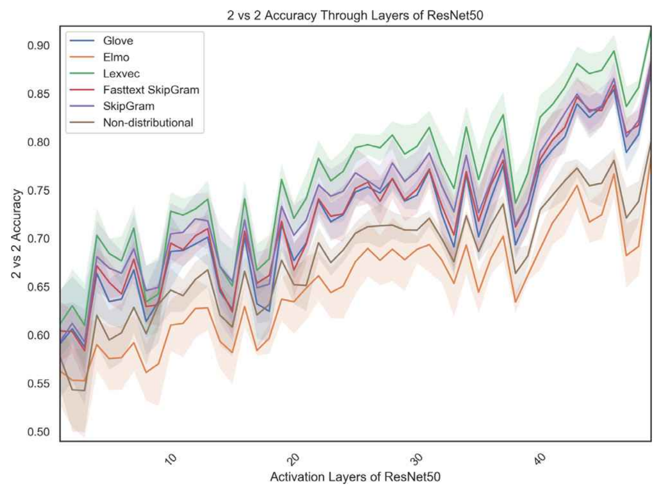


Fig. 2. 2 vs. 2 accuracy for the activation layers of ResNet-50 and several DS models. Shaded area represents one standard deviation over the five independent samples.

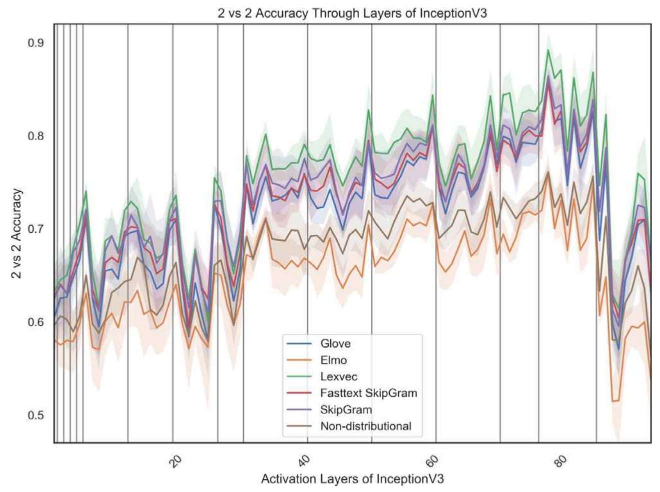


Fig. 3. 2 vs. 2 accuracy for the activation layers of Inception-v3 and several DS models. Inception module boundaries are marked with vertical lines. Between adjacent vertical lines the values plotted may represent parallel representations from within the same module. Shaded area represents one standard deviation over the five independent samples.

and the Non-distributional models trend lower. Elmo vectors are meant to operate on words in context (e.g. in a sentence), and we did not supply any context to our single word concepts. Thus, it is not surprising that Elmo performs so poorly. It should be noted, however, that finding the highest scoring DS model is *not* a goal of this paper. Rather, we are looking for patterns in the 2 vs. 2 accuracy that tell us something about information accumulating in the CNN. As all the 2 vs. 2 results have similar shapes and are all within one standard deviation of each other, it would be reasonable to use any of the top few models. For this reason, the experiments in subsequent sections proceed with SkipGram vectors.

In Fig. 2, we observed that the 2 vs. 2 accuracy in the layer immediately before or after a residual block is always higher than the layers inside a residual block. For example, this can be seen in layer 19 of Fig. 2 (the end of a residual block) when the 2 vs. 2 accuracy vastly improves. This may be explained by residual learning theory. The ResNet-50 architecture is composed of *residual blocks* which contain convolutional layers internally and a skip-connection that connects the input of the residual block to the final layer (He et al., 2015). Conceptually, each residual block is a module that calculates a small change $F(x)$ for a given input x to the residual block. The *add* layer at the end of residual blocks combines $F(x)$ with original input x via the skip connections. Combining $F(x)$ with x provides additional information to the activation layer after a residual block. This effect is also very clear in the annotated ResNet-50 architecture diagram (Fig. C.8). These results provide a semantic argument for the effectiveness of residual learning theory, and illustrate the power of the 2 vs. 2 technique.

Fig. 3 shows the 2 vs. 2 accuracy of activation layers of Inception-v3. The Inception-v3 architecture consists of three different types of inception modules occurring in series (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2015). Within each module, multiple convolutional and pooling operations happen in parallel that are concatenated at the end of the module. Because of these parallel connections, the layers (x axis in Fig. 3) cannot be linearly ordered. For this reason, we indicate the module boundaries using vertical lines (i.e. all activations within a module appear in between adjacent vertical lines, and thus between adjacent vertical lines, the plotted points may actually appear in parallel). Note that within an inception module, the 2 vs. 2 accuracy decreases and increases again as the block ends. This implies that the power of the module may be from combining multiple computational streams.

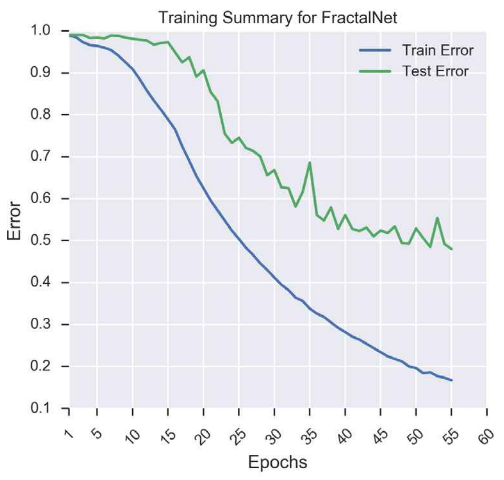
5.1. Training CNNs

Our ability to track the accumulation of semantic information during *model training* also offers additional insights. We trained *FractalNet* (Larsson et al., 2016) on CIFAR-100 (Krizhevsky & Hinton, 2009), using methods and hyperparameters described in the original FractalNet paper. At every five epochs during the training process, we extracted the hidden layers of the network and performed 2 vs. 2 tests using SkipGram and the images from the CIFAR-100 training set. The results for the first 60 epochs appears in Fig. 4, and Appendix D shows all 400 epochs. Note that the initial layers of the CNN learn semantics before the later layers, and that they learn semantics within the first few epochs of training. The 2 vs. 2 accuracy for the initial layers remains constant throughout the remainder of training, and the 2 vs. 2 accuracy for the later layers continues to increase until the end of the first 60 epochs. Thus, a significant part of later learning is driven by the middle and later layers of the CNN. We also measured the 2 vs. 2 accuracy for test images, and found it to be 2%–3% lower than the accuracy on train images.

We also noted that as the network starts to overfit, the 2 vs. 2 accuracy curve becomes noisy (see Fig. C.10), raising the question of what the 2 vs. 2 accuracy would look under the regime of permuted labels (Zhang, Bengio, Hardt, Recht, & Vinyals, 2016). We trained *FractalNet* after randomly permuting both the train and test image labels, and trained the until we achieved 99.9% training accuracy. As expected, the test accuracy was close to chance (1%). We conducted the 2 vs. 2 tests for various layers of CNN. We found that, even though the network achieves close to perfect classification accuracy on train images, the 2 vs. 2 accuracy stays consistently low. This points to another method for identifying overfitting during CNN training: *a network fitting to noise does not learn semantics.*

5.2. Analysis of misclassifications

No CNN is perfect, so every CNN misclassified some images. We were interested in quantifying the magnitude of the mistakes made by a particular CNN. Again, we found we could use DS models to assist us in this task.



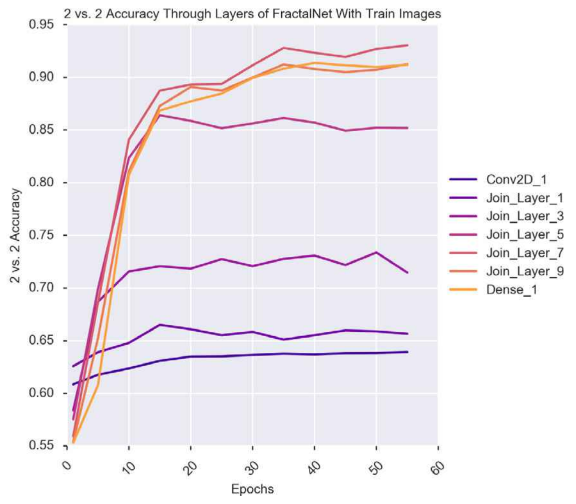


Fig. 4. The emergence of semantic information during training of FractalNet on CIFAR-100. Left: Training and test error for the first 60 epochs training. Right: The 2 vs. 2 accuracy using SkipGram vectors for layers of FractalNet during training. We show 2 vs. 2 accuracy at the first convolutional layer, the end of every second fractal network block, and the last fully connected layer. The initial layers of the CNN learn semantics before the later layers, but later layers continue to improve in later epochs.

For each misclassified image, we computed the cosine similarity of word vectors representing the true and predicted class. Word vectors for similar concepts will have high cosine similarity. Thus, if the cosine similarity between the true and predicted class is large (i.e. the true and predicted classes are similar), then the mistake is less egregious (and vice-versa). We computed the aggregate cosine similarity over all images misclassified by a given CNN. We found that VGG-16 makes more egregious classification mistakes (average of 0.22 cosine similarity) as compared to Inception-v3 and ResNet-50 (0.27 and 0.29 cosine similarity, respectively).

A small number of mistakes made by these CNNs were selected manually for further analysis, and are shown in Fig. 5. In Fig. 5. The misclassifications represented by images 1–4 are less serious than those represented by images 5–8. For example, in image 1, both catamaran and trimaran are similar in visual appearance, and their semantic similarity means they are likely used in similar contexts in the corpora on which the DS models are trained.

On the other hand, the *swing* classified as a *prison* by VGG16 in image 8 is a major mistake. Swing and Prison are very dissimilar concepts which are unlikely to be used in the same text context, and thus have a very small cosine similarity score of 0.02.

Some examples of mistakes made by ResNet50 and Inception-v3 are shown under images 1–4. Interestingly, in image 3, the CNN misclassified the image of a *wheelbarrow*, labeling it *shovel*. However, there is a shovel present in the image along with a wheelbarrow. Furthermore, the CNN does indeed predict *wheelbarrow* correctly in the top 5 predictions. This implies that semantic signals related to both *wheelbarrow* and *shovel* were present in the CNN activations.

Another type of scenario typically seen in the misclassification by VGG16 is shown in image 4 (true: *washer*, predicted: *tabby*). Here the predicted and true class are very dissimilar, but both the concepts are present in the image, and both appear in the top 5 predictions. Again, this could mean that semantic signals related to both *washer* and *tabby* were present in the CNN activations.

5.3. Adversarial examples

Adversarial examples are one of the more serious threats to the adoption of CNNs for practical use, and the vulnerability of neural systems to relatively minor perturbations is a growing concern. We used SkipGram to explore the hidden representations of Inception-v3 when exposed to adversarial examples. This allowed us to better understand the internal representation of CNNs during adversarial attacks. We focus on the Inception-v3 model here as it has the highest ImageNet challenge top-1 accuracy of the three models (Canziani, Paszke, & Culurciello, 2016; Russakovsky et al., 2015). We use the 1 vs. 2 test, which passes if the source class (d_{source}) is closer than the adversarial class (d_{target}) to the CNN correlation vector ($i_{adversarial}$).

Fig. 6 shows the 1 vs. 2 accuracy through layers of Inception-v3 for the 596 adversarial examples, broken down into six target concept classes (based on the correlation between the target and source word vectors). For simplicity here, we show only the 1 vs. 2 accuracy for "Mixed" layers, which join the result of several parallel paths within a module. On average, in earlier layers of the network, the 1 vs. 2 accuracy is higher, but by the later layers of the network, the hidden representations have become more similar to the adversarial class (d_{target}), pushing the 1 vs. 2 accuracy below 50. Below 50, the hidden representations are more similar to the target concept than to the source concept.

Compared to the five other target concepts, the most similar target concept has 1 vs. 2 accuracy that hovers around 0.5 through most layers of the CNN. This is likely because the semantic representation for two highly related concepts is more difficult to disambiguate, so the 1 vs. 2 accuracy will be closer to chance. For all other target concepts, which are less related to the source concept, we see a pattern much more similar to the average: high 1 vs. 2 accuracy in early layers, and lower 1 vs. 2 accuracy for later layers. This implies that later layers are being targeted by adversarial attacks, probably by an accumulation of small perturbations through the network. We will explore one possible use of this discovery in Section 6.2.

6. Utility and examples

Much of the work presented here is expository, and so here we describe several possible practical applications for the information shared here.

6.1. Improving CNN architecture design

CNNs have made an impressive mark on the computer vision community, but progress in creating new CNN architectures

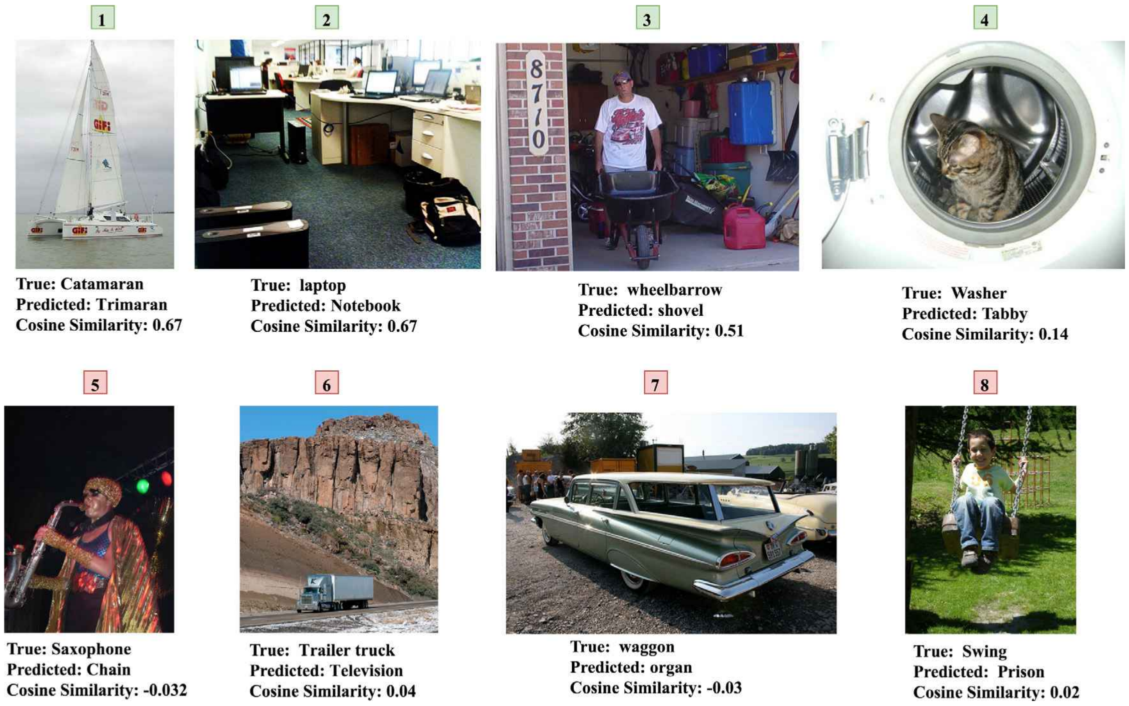


Fig. 5. A qualitative analysis of the classifications mistakes of CNNs. The images above were manually selected to illustrate misclassifications from VGG16, Inception-v3, and ResNet50. We consider a classification error to be more serious if the cosine similarity between the word vectors for the true and predicted class is small. Images (1–4) are sampled from mistakes made by Inception-v3 and ResNet50. They are considered to be small mistakes due to high cosine similarity between true and predicted class. Images (5–8) are sampled from mistakes made by VGG16, and represent more serious misclassifications.

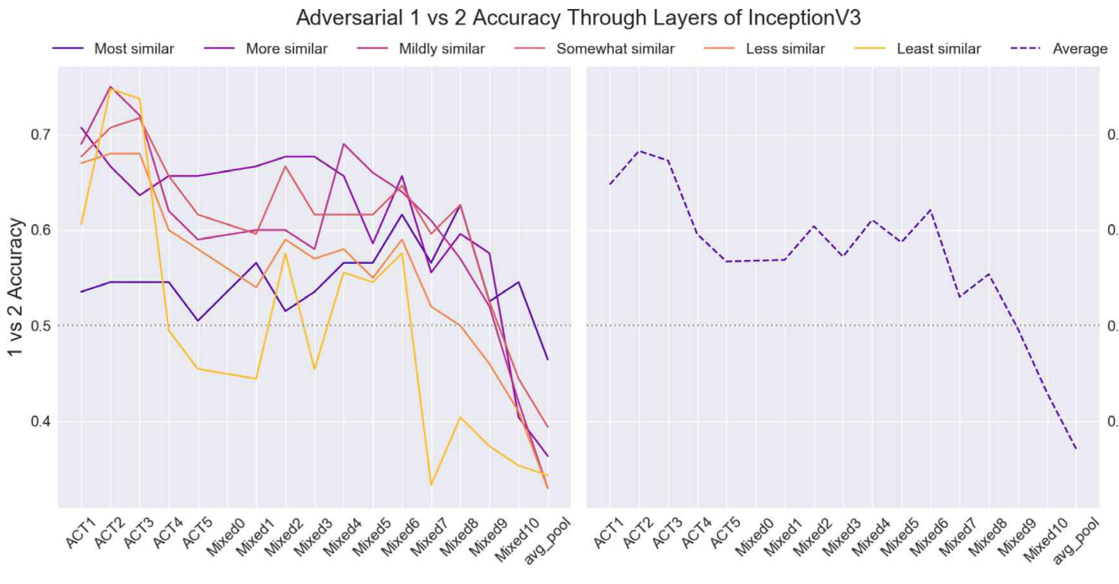


Fig. 6. 1 vs. 2 accuracy through layers of Inception-v3 for adversarial examples. When the accuracy is above 0.5, there is more evidence for the true class than the adversarial class. Left: Average 1 vs. 2 accuracy for the six categories of adversarial targets. Right: average 1 vs. 2 accuracy for all six categories of adversarial targets. ACT: activation layer, Mixed: concatenation block at the end of a module.

is still often a very experimental process guided by intuition and trial-and-error. It can be difficult to know what tricks to try next, and hard to determine which features of an architecture are contributing most to the accuracy of a model. Here we explore the architecture of ResNet-50 and Inception-v3 using our methodology to identify those architectural features that are likely contributing to the success of the architecture, and identify some features which may not be as beneficial. Importantly, the techniques here are easy to operationalize, and do not require the manual examination of images in the training set. Thus, the suggestions included here may be more practical, especially during the rapid prototyping stages of network development.

Recall our results on ResNet-50 (Figs. 2 and C.8) that the 2 vs. 2 accuracy in the layer immediately before or after a residual block tends to be higher than the layers inside the residual block. Recall also that each residual block is supplied with some input representation x. Each residual block ends by combining both $F(x)$ (the function learned as part of that block) as well as the original x, and both pieces of information are supplied as input into the next residual block. Because the intermediate stages in

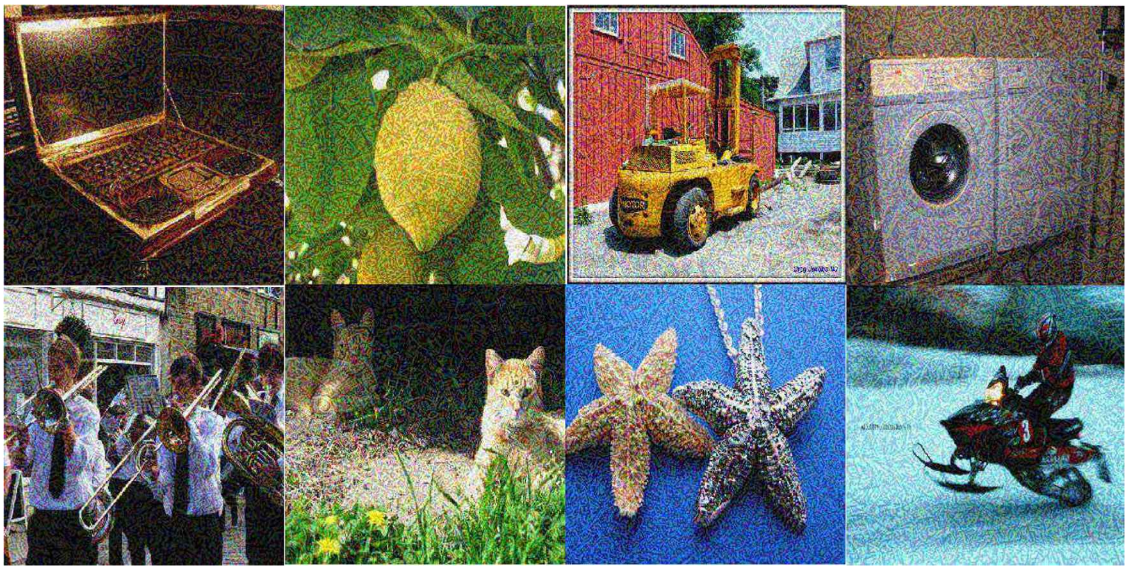


Fig. A.7. A sample of eight targeted adversarial examples.

computing $F(x)$ do not appear to add much to the 2 vs. 2 accuracy, it is likely that the residual connections in ResNet-50 are very important, and are a good candidate feature to include in future CNN architectures. Researchers designing new CNN architectures should consider including such residual connections, not only because they improve gradients at train time, but also because they appear to improve the representations at test time.

Figs. 3 and C.9 show the 2 vs. 2 accuracy of activation layers of Inception-v3. Similar to ResNet-50, within an inception module we see 2 vs. 2 accuracy decrease and increase again at the end of the module when mixing the various parallel convolution operations together (this is most clear in the architecture diagram, Fig. C.9). Interestingly, shallower parallel connections seem to maintain 2 vs. 2 accuracy better than paths with many convolutions, which implies that the lower-dimensional convolutions may be driving some of the early performance. Thus, a researcher might experiment with omitting the higher-dimensional convolutions in early layers of the CNN, as they do not appear to be adding to the early increases in accuracy. It should also be noted that the average pooling at the end of the last inception module also provides a tremendous boost in network performance while reducing the number of parameters. These two case studies show how our framework allows us to quantitatively measure which CNN architectural innovations are truly benefiting the performance of a particular CNN.

6.2. Defense against adversarial attack

In Fig. 6 we saw that even under adversarial attack, early layers of the network contain the information necessary to identify the source class (that is, the 1 vs. 2 accuracy is above 0.5). In fact, for very dissimilar target classes, the evidence for the source class outweighs the target class until quite late in the CNN hierarchy. Recall also the results for non-adversarial examples in Figs. 2 and 3, which show that 2 vs. 2 accuracy increases as we move through the CNN hierarchy. Because adversarial attacks work through minor perturbations that leverage the details of the network's decision boundary, the correlation of the hidden layers may display a pattern of 2 vs. 2 accuracy that differs from non-adversarial images. Thus, researchers may be able to develop techniques to defend against adversarial attack by looking for changes in the correlation of the predicted word vector to the image's hidden representations at different layers of the CNN. Because the DS models are an external and independent source of information, defense that incorporate a DS model may be difficult to evade.

Conversely, this also suggests researchers pursue a mechanism for *strengthening* adversarial attacks. Current adversarial attacks suffer from transferability problems, possibly because they exploit the noisiness of decision boundaries, which can vary widely between networks (Liu, Chen, Liu, & Song, 2017). All three neural networks we studied learn representations that produce high 2 vs. 2 results against a variety of DS models. Thus, researchers may be able to implement an adversarial attack which is regularized towards producing hidden representations that are more correlated to the word vector of the *target* class, rather than one purely optimized to exploit the network's decision boundary. Such an attack may prove to generalize better across networks.

6.3. Improved training

In Section 5.1, we showed that early in training the first layers of the CNN learn quickly, and after the first 10 epochs they change very little. The majority of movement towards semantic representations at the end of training happens in the later layers of the CNN. This finding has many implications, including possible new regularization schemes where DS models are used to guide CNN training (similar to the regularization schemes in Federer, Xu, Fyshe, & Zylberberg, 2020). This result also implies that we could improve training time by freezing the weights of early layers after the first few epochs of training.

7. Implications for artificial intelligence and neuroscience

Our work has uncovered what is a surprising and otherwise unreported convergence of two very different neural network research areas: language modeling and computer vision. The networks we study here differ in many ways: prediction task, data, and architecture. Yet they produce representations that are remarkably similar. This pattern holds across multiple CNNs and multiple DS models. From a scientific standpoint, this is evidence that the information being extracted and computed by CNNs represents true semantic information that faithfully reflects patterns in the natural world. As we move towards incorporating CNNs into more everyday technology, this sort of convergence of evidence will help us trust and further validate our models.

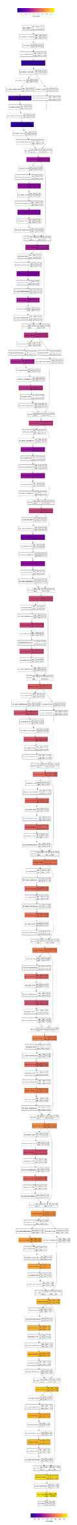


Fig. C.8. The architecture diagram of ResNet-50 colored to indicate the 2 vs. 2 accuracy using SkipGram word-vectors (He et al., 2015). This is a high resolution image and is best viewed electronically at great magnification. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

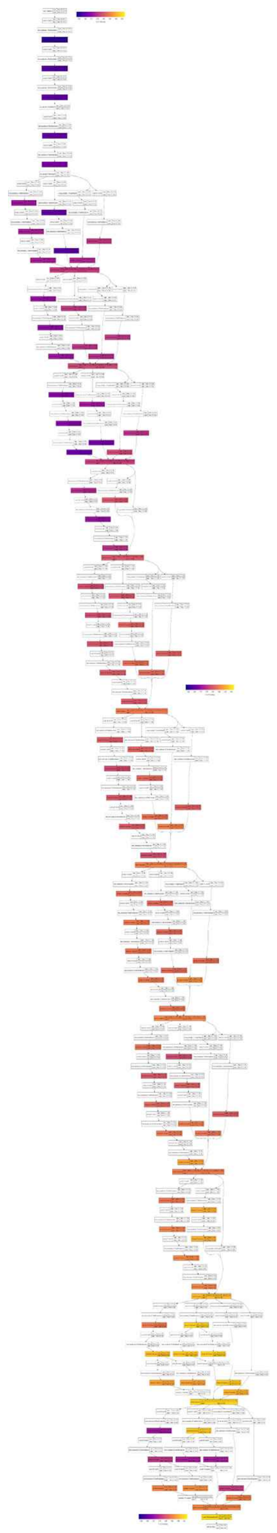


Fig. C.9. The architecture diagram of Inception-v3 (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2015) colored to indicate 2 vs. 2 accuracy using SkipGram word-vectors. This is a high resolution image and is best viewed electronically at great magnification. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We can use Marr's levels of analysis to more deeply understand our findings. Marr breaks down the understanding of an information processing system into three main levels: computational (the thing to be computed), algorithmic (how it is computed), and implementational (the physical realization of that computation, i.e. the hardware). The two systems in question (a DS model and a CNN) trivially share their implementational levels in that they both can run on typical computers. However, the systems do not share a top level computational goal; they are trained for different prediction tasks. It is also clear, based on the architectural differences between the models considered, much of the algorithmic level is also not shared. Nevertheless, the two systems actually do have *some* overlap at the algorithmic level, because their computed representations encode similar relationships between concepts. Though so much of the systems are different, they compute intermediate representational states that are strikingly similar.

But, Marr developed his levels of analysis because he was interested in understanding the human brain as an information processing system. How can our work assist in that goal? There are several lines of research showing that the human brain also shares representational states with DS models (Jain & Huth, 2018; Pereira et al., 2018) and with CNNs (Horikawa & Kamitani, 2017; Khaligh-Razavi & Kriegeskorte, 2014). Clearly these models do not share their implementational level with the human brain, which computes using neurons, not transistors. Though primates certainly are doing more with their visual systems than simple object detection, like in a CNN, object detection is one of the visual systems goals. So, we can grant that the top level computational goal is shared in some instances. Though we do not yet understand the intricacies of the primate visual system, it is likely that to accomplish object recognition, each of the CNNs we considered uses an algorithm that differs from the brains. And yet, again, there is a connection between the representations that a CNN learns, and those recorded in the brain using both fMRI and direct recordings. Something about what the brain is doing is shared with what CNNs learn to do. Furthermore, Horikawa and Kamitani (2017) find that early layers in a CNN correlate to the V1–V3 area of the human brain, and layers closer to the final classification layer correlate more strongly with the higher-order areas of the human brain (e.g. fusiform face area, parahippocampal place area). So, even though there is so much that differs architecturally, these CNN models are producing representations in a hierarchical fashion that resembles what the hierarchy of the human visual system is computing.

But what about the DS models? The DS models we considered here are trained on a variety of tasks (e.g. SkipGram: predicting context words, Glove: regression-based model to predict local and global patterns of word co-occurrence), none of which is a good match for the tasks the human brain engages in while understanding language. So, they do not share a top-level computational goal. Several of the DS model architectures are so simple that it is highly unlikely that they share much algorithmic overlap with the human brain. And yet, again, there is a strong resemblance between the representations in these DS models and the human brain's representations. Even for their simplified language goals, the information these models extract is organized using representations that look like the human brains.

Our work brings together these two disparate lines of work, and shows that CNNs and DS model representations are correlated. There is some evidence that the visual system is involved in the understanding of language (Sudre et al., 2012), and our work lends credence to that finding. Our work hints that vision and language are linked, and that there are deep connections between the neural processing of the two information streams. Because language is so new, evolutionarily speaking, it has been argued that human language processing is performed by brain areas that evolved for other tasks (Lieberman, 2002). Our work is evidence that we ought to consider vision when studying language, and vice versa, because information processing systems for the two tasks show large degrees of overlap.

8. Summary

Here, we studied the representations learned by CNNs, using the representations from several DS models. We measured the behavior of fully trained networks, a network during training, and during the processing of adversarial images. Our results point to several new avenues for training CNNs, and also for characterizing their behavior during failure modes. This is a new approach to understanding CNNs that brings quantifiable interpretability without requiring the visual inspection of images or activation patterns. This method of analysis could further operationalize the development of deep learning architectures by providing a framework within which to reason about changes to an architecture, and what each new architectural innovation brings in our quest to help computers understand our world.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Chris Foster was supported by the NSERC (Natural Sciences and Engineering Research Council) CGSM program. Alona Fyshe is a fellow in the CIFAR (Canadian Institute for Advanced Research) Learning in Machines and Brains program, holds a Canada CIFAR AI Chair, and is funded by NSERC (Natural Sciences and Engineering Research Council). The computational work was supported in part by infrastructure made available by WestGrid (https://www.westgrid.ca/) and Compute Canada (https://www.computecanada.ca/).

Appendix A. Generated adversarial examples

We generated a series of targeted adversarial examples against InceptionV3 using adversarial attacks as described in Section 4. In Fig. A.7 we show eight example images from the attack with our chosen parameters. While the pixel changes to the image are visually detectable, they are minor and the true semantic concepts of the images are clearly retained.

Appendix B. Annotated ResNet architecture diagram

We created a network architecture diagram graphic of ResNet-50 which is shown in Fig. C.8. We annotated the activations on the architecture diagram using a color gradient correlating to the 2 vs 2 accuracy for that layer. Layers which are more strongly represent the semantics of the true label score higher on the 2 vs 2 test. We can see here that the semantic representation becomes stronger as the ResNet network becomes deeper.

However, a unique feature of interest is the skip connections in ResNet. A consistent pattern through the layers augmented with skip connections is that the semantic representation will decrease through the branch, until the final *add* layer with the skip connection which leads to a higher semantic representation than before the branch. This provides evidence that skip connections are useful techniques for passing along semantic representation in deeper networks.

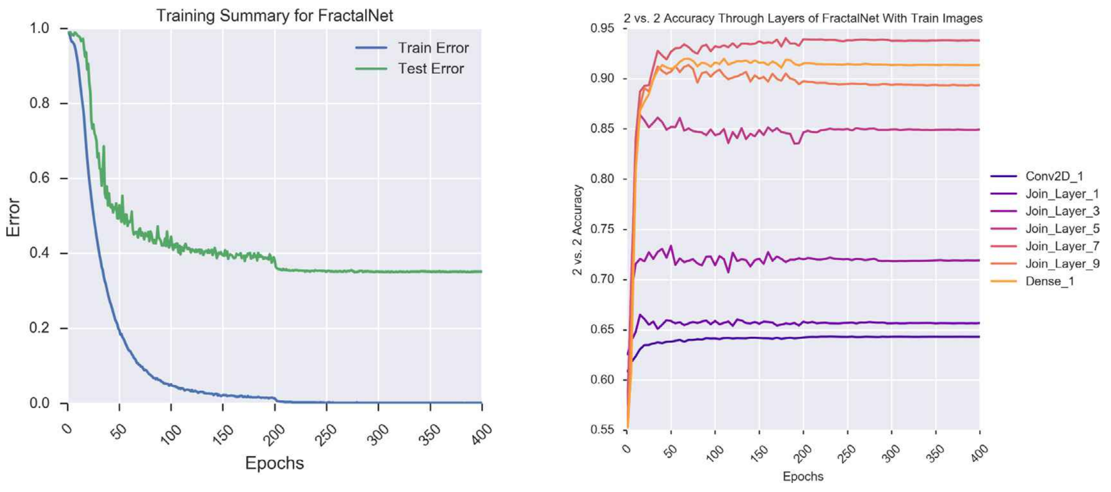


Fig. C.10. The emergence of semantic information during training of FractalNet on CIFAR-100. Left: training and test error for the 400 epochs of training. Right: the 2 vs. 2 accuracies for layers of FractalNet during training.

Appendix C. Annotated InceptionV3 architecture diagram

We created a network architecture diagram graphic of Inception-v3 which is shown in Fig. C.9. We annotated the activations on the architecture diagram using a color gradient correlating to the 2 vs 2 accuracy for that layer. Layers which are more strongly represent the semantics of the true label score higher on the 2 vs 2 test. As with the prior two networks, we continue to see a general pattern of increasing semantic representation through later layers of the network.

Inception-v3 has a complicated architecture, which makes it a good candidate for exploration using this semantic annotation method. We see a similar pattern to ResNet-50, where parallel layers within Inception-v3 blocks may decrease in semantic representation before increasing again at the mixing layer of the block. However, not all parallel layers decreasing in semantic representation suggesting some parallel layers may provide more semantic value than others within the block.

Of particular note in this network is the very large decrease in semantic representation in the final layers for some parallel components of the block. This annotation method may be a useful tool for identifying steps in the network which are not improving semantic representation, and removal or adjustment or those layers may provide a positive affect on classification accuracy.

Appendix D. FractalNet 2 vs. 2 accuracy for 400 epochs

Fig. C.10 shows the train/test error and 2 vs. 2 accuracy for all 400 epochs during FractalNet training. The change in behavior around 200 epochs corresponds to a reduction in learning rate.

References

Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K. T., Montavon, G., et al. (2019). Innvestigate neural networks! *Journal of Machine Learning Research*, *20*, 1–8, arXiv:1808.04260.

Anderson, A. J., Bruni, E., Bordignon, U., Poesio, M., & Baroni, M. (2013). Of words, eyes and brains: Correlating image-based distributional semantic models with neural representations of concepts. In *Empirical methods in natural language processing* No. October. (pp, 1960–1970).

Baker, C. F. C., Fillmore, C. J., & Lowe, J. B. (1998). The Berkeley FrameNet project. In *Proceedings of the 36th annual meeting on association for computational linguistics: Vol. 1*, (p. 86). Association for Computational Linguistics.

Benjamini, Y., & Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, *29*(4), 1165–1188.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135—-146, arXiv:arXiv:1607.04606v2.

Bologna, G. (2019). Propositional rules generated at the top layers of a CNN. In *International work-conference on the interplay between natural and artificial computation*.

Bruni, E., & Baroni, M. (2013). Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, *48*.

Canziani, A., Paszke, A., & Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *CoRR*, *abs/1605.07678*, arXiv:1605.07678.

Chollet, F., et al. (2015). Keras. https://github.com/keras-team/keras.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Li, F. (2009). ImageNet: a large-scale hierarchical image database. In *2009 IEEE computer society conference on computer vision and pattern recognition* (pp. 248–255). http://dx.doi.org/10.1109/CVPRW.2009.5206848.

Dharmaretnam, D., & Fyshe, A. (2018). The emergence of semantics in neural network representations of visual information. In *Proceedings of the 16th annual conference of the North American chapter of the association for computational linguistics: Human language technologies. New Orleans, Louisiana*.

Dong, Y., Liao, F., Pang, T., Su, H., Hu, X., Li, J., et al. (2017). Boosting adversarial attacks with momentum. arXiv preprint arXiv:1710.06081.

Faruqui, M., Tsvetkov, Y., Yogatama, D., Dyer, C., & Smith, N. A. (2015). Sparse overcomplete word vector representations. In *ACL-2015* (pp. 1491–1500). arXiv:1506.02004.

Federer, C., Xu, H., Fyshe, A., & Zylberberg, J. (2020). Improved object recognition using neural networks trained to mimic the brain's statistical properties. *Neural Networks*, *131*, 103–114, arXiv:1905.10679.

Fellbaum, C. (1998a). A semantic network of english: The mother of all wordnets. *Computers and the Humanities*, *32*(2), 209–220. http://dx.doi.org/10.1023/A:1001181927857.

Fellbaum, C. (1998b). *WordNet: An electronic lexical database*. Cambridge, MA: MIT Press.

Frome, A., Corrado, G., & Shlens, J. (2013). Devise: A deep visual-semantic embedding model. *Advances in Neural Information Processing Systems*, 1–11, arXiv:arXiv:1312.5650v3.

Gonzalez-Garcia, A., Modolo, D., & Ferrari, V. (2018). Do semantic parts emerge in convolutional neural networks? *International Journal of Computer Vision*, *126*(5), 476–494, arXiv:1607.03738.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*, arXiv:1512.03385.

Hill, F., Reichart, R., & Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, *41*(4), 665–695, arXiv:1408.3456.

Hollis, G., Westbury, C., & Lefsrud, L. (2017). Extrapolating human judgments from skip-gram vector representations of word meaning. *Quarterly Journal of Experimental Psychology*, *70*(8), 1603–1619.

Horikawa, T., & Kamitani, Y. (2017). Generic decoding of seen and imagined objects using hierarchical visual features. *Nature Communications*, *8*(May), 1–15, arXiv:1510.06479.

Jain, S., & Huth, A. (2018). Incorporating context into language encoding models for fMRI. *Advances in Neural Information Processing Systems*, 6628–6637.

Karpathy, A. (2014). What I learned from competing against a ConvNet on ImageNet. http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/. (Accessed 27 September 2018).

Khaligh-Razavi, S. M., & Kriegeskorte, N. (2014). Deep supervised, but not unsupervised, models may explain IT cortical representation. *PLoS Computational Biology*, *10*(11).

Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images*. Citeseer.

Larsson, G., Maire, M., & Shakhnarovich, G. (2016). FractalNet: Ultra-deep neural networks without residuals. *CoRR*, *abs/1605.07648*, arXiv:1605.07648.

Lazaridou, A., Bruni, E., & Baroni, M. Is this a wampimuk? Cross-modal mapping between distributional semantics and the visual world. In *Proceedings of the annual meeting of the association for computational linguistics* (pp. 1403–1414).

Lieberman, P. (2002). On the nature and evolution of the neural bases of human language. *Yearbook of Physical Anthropology*, *45*, 36–62.

Liu, Y., Chen, X., Liu, C., & Song, D. (2017). Delving into transferable adversarial examples and black-box attacks. In *Proceedings of 5th international conference on learning representations*.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, *abs/1301.3781*, arXiv:1301.3781.

Morcos, A. S., Raghu, M., & Bengio, S. (2018). Insights on representational similarity in neural networks with canonical correlation. arXiv:1806.05759.

Papernot, N., Carlini, N., Goodfellow, I., Feinman, R., Faghri, F., Matyasko, A., et al. (2017). Cleverhans v2.0.0: An adversarial machine learning library. arXiv preprint arXiv:1610.00768.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe : Global vectors for word representation. In *Conference on empirical methods in natural language processing*. Doha, Qatar.

Pereira, F., Lou, B., Pritchett, B., Ritter, S., Gershman, S. J., Kanwisher, N., et al. (2018). Toward a universal decoder of linguistic meaning from brain activation. *Nature Communications*, *9*(1), 963.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., et al. (2018). Deep contextualized word representations. In M. A. Walker, H. Ji, A. Stent (Eds.), *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: Human language technologies* (pp. 2227–2237). Orleans, Louisiana: Association for Computational Linguistics.

Raghu, M., Gilmer, J., Yosinski, J., & Sohl-Dickstein, J. (2017). SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *NIPS* (Nips), (pp. 1–10). arXiv:1706.05806.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252. http://dx.doi.org/10.1007/s11263-015-0816-y.

Saini, U. S., & Papalexakis, E. E. (2018). A peek into the hidden layers of a convolutional neural network through a factorization lens. arXiv:1806.02012.

Salle, A., Idiart, M., & Villavicencio, A. (2016). Enhancing the lexvec distributed word representation model using positional contexts and external memory. ArXiv preprint, arXiv:1606.01283.

Socher, R., Ganjoo, M., & Sridhar, H. (2013). Zero-shot learning through cross-modal transfer. In *International conference on learning representations* (pp. 1–7). arXiv:arXiv:1301.3666v2.

Sudre, G., Pomerleau, D., Palatucci, M., Wehbe, L., Fyshe, A., Salmelin, R., et al. (2012). Tracking neural coding of perceptual and semantic features of concrete nouns. *NeuroImage*, *62*(1), 463–451.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Computer vision and pattern recognition*. http://arxiv.org/abs/1409.4842.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the inception architecture for computer vision. *CoRR*, *abs/1512.00567*, arXiv:1512.00567.

Wagner, J., Kohler, J. M., Gindele, T., Hetzel, L., Wiedemer, J. T., & Behnke, S. (2019). Interpretable and fine-grained visual explanations for convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Wasserman, L. (2004). *All of statistics. Vol. C* (p. 455). arXiv:arXiv:1011.1669v3.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding neural networks through deep visualization. In *Deep learning workshop, 31 st international conference on machine learning*. Lille, France. arXiv:1506.06579.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833). arXiv:1311.2901.

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *CoRR*, *abs/1611.03530*, arXiv:1611.03530.

Zhang, Q., Yang, Y., Ma, H., & Wu, Y. N. (2019). Interpreting CNNs via decision trees. In *The IEEE conference on computer vision and pattern recognition* (pp. 6261–6270). arXiv:1802.00121.