SIAMESE NEURAL NETWORK FOR ONE SHOT IMAGE RECOGNITION

Shalini Dixit
MCA Student
School of Computing Science and Engineering
Galgotias University Greater Noida, India

Pratik Garg
MCA Student
School of Computing Science and Engineering
Galgotias University Greater Noida, India

Shivam Chauhan
MCA Student
School of Computing Science and Engineering
Galgotias University Greater Noida, India

Abstract

The process of learning good features for ma- chine learning applications can be very compu- tationally expensive and may prove difficult in cases where little data is available. A prototyp- ical example of this is the *one-shot learning* set- ting, in which we must correctly make predic- tions given only a single example of each new class. In this paper, we explore a method for learning *siamese neural networks* which employ a unique structure to naturally rank similarity be- tween inputs. Once a network has been tuned, we can then capitalize on powerful discrimina- tive features to generalize the predictive power of the network not just to new data, but to entirely new classes from unknown distributions. Using a convolutional architecture, we are able to achieve strong results which exceed those of other deep learning models with near state- of-the-art perfor- mance on one-shot classificationtasks.

Humans exhibit a strong ability to acquire and recognize new patterns. In particular, we observe that when presented with stimuli, people seem to be able to understand new concepts quickly and then recognize variations on these concepts in future percepts (Lake et al., 2011).

Machine learning has been successfully used to achieve state-of- the-art performance in a variety of applications such as web search, spam detection, caption generation, and speech and image recognition.

However, these algorithms often break down when forced to make predictions about data for which little supervised information is available.

We desire to generalize to these unfamiliar categories without neces- sitating extensive retraining which may be either expensive or impossible due to limited data or in an online prediction setting, such as web retrieval.

One particularly interesting task is classification under the restriction that we may only observe a single example of each possible class before making a prediction about a test instance.

This is called *one-shot learning* and it is the pri- mary focus of our model presented in this work (Fei-Fei et al., 2006; Lake et al., 2011).

This should be distinguished from *zero-shot learning*, in which the model cannot look at any examples from the target classes (Palatucci et al., 2009).

One-shot learning can be directly addressed by develop- ing domain-specific features or inference procedures which possess highly discriminative properties for the target task.

As a result, systems which incorporate these methods tend to excel at similar instances but fail to offer robust solutions that may be applied to other types of problems.

In this pa- per, we present a novel approach which limits assumptions on the structure of the inputs while automatically acquir- ing features which enable the model to generalize success- fully from few examples.

We build upon the deep learn- ing framework, which uses many layers of non-linearities to capture invariances to transformation in the input space, usually by leveraging a model with many parameters and then using a large amount of data to prevent overfitting (Bengio, 2009; Hinton et al., 2006).

These features are very powerful because we are able to learn them without imposing strong priors, although the cost of the learning algorithm itself may beconsiderable.

1. Approach

In general, we learn image representations via a supervised metric-based approach with siamese neural networks, then reuse that network's features for one-shot learning without any retraining.

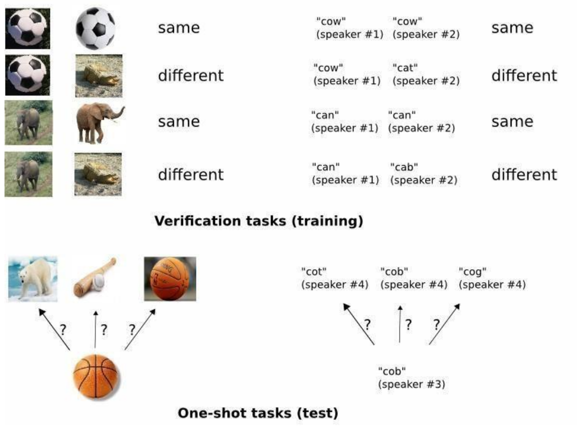

Figure 1. Our general strategy. 1) Train a model to discriminate between a collection of same/different pairs. 2) Generalize to evaluate new categories based on learned feature mappings for verification.

In our experiments, we restrict our attention to character recognition, although the basic approach can be replicated for almost any modality (Figure 1). For this domain, we employ large siamese convolutional neural networks which **a)** are capable of learning generic image features useful for making predictions about unknown class distributions even when very few examples from these new distributions are available; **b)** are easily trained using standard optimization techniques on pairs sampled from the source data; and **c)** provide a competitive approach that does not rely upon domain-specific knowledge by instead exploiting deep learning techniques.

To develop a model for one-shot image classification, we aim to first learn a neural network that can discriminate between the class-identity of image pairs, which is the standard *verification* task for image recognition. We hy- pothesize that networks which do well at at verification should generalize to one-shot classification. The verifica- tion model learns to identify input pairs according to the probability that they belong to the same class or differ- ent classes. This model can then be used to evaluate new images, exactly one per novel class, in a pairwise manner against the test image. The pairing with the highest score according to the verification network is then awarded the highest probability for the one-shot task. If the features learned by the verification model are sufficient to confirm or deny the identity of characters from one set of alpha- bets, then they ought to be sufficient for other alphabets, provided that the model has been exposed to a variety of alphabets to encourage variance amongst the learned features.

2. Related Work

Overall, research into one-shot learning algorithms is fairly immature and has received limited attention by the machine learning community. There are nevertheless a few key lines of work which precede this paper.

The seminal work towards one-shot learning dates back to the early 2000's with work by Li Fei-Fei et al. The au- thors developed a variational Bayesian framework for one-shot image classification using the premise that previously learned classes can be leveraged to help forecast future ones when very few examples are available from a given class (Fe-Fei et al., 2003; Fei-Fei et al., 2006). More re- cently, Lake et al. approached the problem of one-shot learning from the point of view of cognitive science, ad- dressing one-shot learning for character recognition with a method called Hierarchical Bayesian Program Learning (HBPL) (2013). In a series of several papers, the authors modeled the process of drawing characters generatively to decompose the image into small pieces (Lake et al., 2011; 2012). The goal of HBPL is to determine a structural ex- planation for the observed pixels. However, inference un- der HBPL is difficult since the joint parameter space is very large, leading to an intractable integration problem.

Some researchers have considered other modalities or transfer learning approaches. Lake et al. have some very recent work which uses a generative Hierarchical Hidden Markov model for speech primitives combined with a Bayesian inference procedure to recognize new words by unknown speakers (2014). Maas and Kemp have some of the only published work using Bayesian networks to pre- dict attributes for Ellis Island passenger data (2009). Wu and Dennis address one-shot learning in the context of path planning algorithms for robotic actuation (2012). Lim fo- cuses on how to “borrow” examples from other classes in the training set by adapting a measure of how much each category should be weighted by each training exemplar in the loss function (2012). This idea can be useful for data sets where very few examples exist for some classes, pro- viding a flexible and continuous means of incorporating inter-class information into the model.

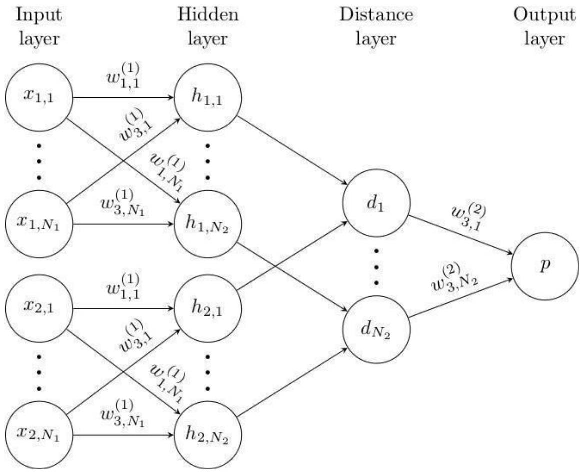

Figure 2. A simple 2 hidden layer siamese network for binary classification with logistic prediction *p*. The structure of the net- work is replicated across the top and bottom sections to form twin networks, with shared weight matrices at each layer.

3. Deep Siamese Networks for image Verification

Siamese nets were first introduced in the early 1990s by Bromley and LeCun to solve signature verification as an image matching problem (Bromley et al., 1993). A siamese neural network consists of twin networks which accept dis- tinct inputs but are joined by an energy function at the top. This function computes some metric between the highest- level feature representation on each side (Figure 2). The parameters between the twin networks are tied. Weight ty- ing guarantees that two extremely similar images could not possibly be mapped by their respective networks to very different locations in feature space because each network computes the same function. Also, the network is symmet- ric, so that whenever we present two distinct images to the twin networks, the top conjoining layer will compute the same metric as if we were to we present the same two im- ages but to the opposite twins.

In LeCun et al., the authors used a contrastive energy func- tion which contained dual terms to decrease the energy of like pairs and increase the energy of unlike pairs (2005).
However, in this paper we use the weighted L_1 distance between the twin feature vectors $\mathbf{h}_1$ and $\mathbf{h}_2$ combined with a sigmoid activation, which maps onto the interval [0,1] Thus a *cross-entropy* objective is a natural choice for train- ing the network. Note that in LeCun et al., they directly learned the similarity metric, which was implictly defined by the energy loss, whereas we fix the metric as specified above, following the approach in Facebook’s DeepFace pa- per (Taigman et al., 2014).
Our best-performing models use multiple convolutional layers before the fully-

connected layers and top-level energy function. Convolutional neural networks have achieved exceptional results in many large-scale computer vision applications, particularly in image recognition tasks (Bengio, 2009; Krizhevsky et al., 2012; Simonyan & Zis- serman, 2014; Srivastava, 2013).

Several factors make convolutional networks especially ap- pealing. Local connectivity can greatly reduce the num- ber of parameters in the model, which inherently provides some form of built-in regularization, although convolu- tional layers are computationally more expensive than stan- dard nonlinearities. Also, the convolution operation used in these networks has a direct filtering interpretation, where each feature map is convolved against input features to identify patterns as groupings of pixels. Thus, the out- puts of each convolutional layer correspond to important spatial features in the original input space and offer some robustness to simple transforms. Finally, very fast CUDA libraries are now available in order to build large convolu- tional networks without an unacceptable amount of train- ing time (Mnih, 2009; Krizhevsky et al., 2012; Simonyan & Zisserman, 2014).

We now detail both the structure of the siamese nets and the specifics of the learning algorithm used in our experiments.

Model

Our standard model is a siamese convolutional neural net- work with *L* layers each with N_l units, where $\mathbf{h}_{1,l}$ repre- sents the hidden vector in layer *l* for the first twin, and $\mathbf{h}_{2,l}$ denotes the same for the second twin. We use exclusively rectified linear (ReLU) units in the first *L* 2 layers and sigmoidal units in the remaining layers.

The model consists of a sequence of convolutional layers, each of which uses a single channel with filters of varying size and a fixed stride of 1. The number of convolutional filters is specified as a multiple of 16 to optimize perfor- mance. The network applies a ReLU activation function to the output feature maps, optionally followed by max- pooling with a filter size and stride of 2. Thus the *k*th filter map in each layer takes the following form:

$$a_{1,m}^{(k)} = \text{max-pool}(\max(0, \mathbf{W}_{l-1,l}^{(k)} * \mathbf{h}_{1,(l-1)} + \mathbf{b}_l), 2)$$
$$a_{2,m}^{(k)} = \text{max-pool}(\max(0, \mathbf{W}_{l-1,l}^{(k)} * \mathbf{h}_{2,(l-1)} + \mathbf{b}_l), 2)$$

where $\mathbf{W}_{l-1,l}$ is the 3-dimensional tensor representing the feature maps for layer *l* and we have taken * to be the *valid* convolutional operation corresponding to returning

only those output units which were the result of complete overlap between each convolutional filter and the input feature maps.

then one more layer com puting the induced distance metric between each siamese twin, which is given to a single sigmoidal outputunit.

More precisely, the prediction vector is given as **p** =σ(Σj(j)1,L−1(j)2,L−1|),

This final layer induces a metric on the learned feature space of the (L 1)th hidden layer and scores the similarity between the two feature vec- tors. The αj are additional parameters that are learned by the model during training, weighting the importance of the component-wise distance. This defines a final Lth fully-connected layer for the network which joins the two siamese twins.

We depict one example above (Figure 4), which shows the largest version of our model that we considered. This net- work also gave the best result for any network on the verification task.

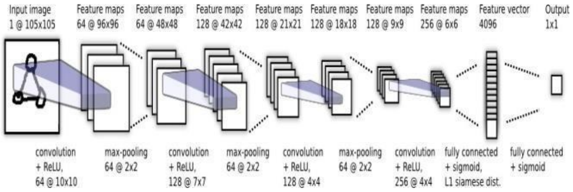

Figure 4. Best convolutional architecture selected for verification task. Siamese twin is not depicted, but joins immediately after the 4096 unit fully-connected layer where the L1 component-wise distance between vectors is computed

Learning

Loss function. Let M represent the minibatch size, where i indexes the ith minibatch. Now let $\mathbf{y}(x_1^{(i)}, x_2^{(i)})$ be a length-M vector which contains the labels for the mini-batch, where we assume $y(x_1^{(i)}, x_2^{(i)}) = 1$ whenever x_1 and x_2 are from the same character class and $y(x_1^{(i)}, x_2^{(i)}) = 0$ otherwise. We impose a regularized cross-entropy objec- tive on our binary classifier of the following form:

$$\mathcal{L}(x_1^{(i)}, x_2^{(i)}) = \mathbf{y}(x_1^{(i)}, x_2^{(i)}) \log \mathbf{p}(x_1^{(i)}, x_2^{(i)}) +$$
$$(1 - \mathbf{y}(x_1^{(i)}, x_2^{(i)})) \log (1 - \mathbf{p}(x_1^{(i)}, x_2^{(i)})) + \boldsymbol{\lambda}^T |\mathbf{w}|^2$$

Optimization. This objective is combined with standard backpropagation algorithm, where the gradient is additive across the twin networks due to the tied weights. We fix a minibatch size of 128 with learning rate η_j, momentum μ_j, and L_2 regularization weights λ_j defined layer-wise, so that our update rule at epoch T is as follows:

$$\mathbf{w}_{kj}^{(T)}(x_1^{(i)}, x_2^{(i)}) = \mathbf{w}_{kj}^{(T)} + \Delta\mathbf{w}_{kj}^{(T)}(x_1^{(i)}, x_2^{(i)}) + 2\lambda_j |\mathbf{w}_{kj}|$$
$$\Delta\mathbf{w}_{kj}^{(T)}(x_1^{(i)}, x_2^{(i)}) = -\eta_j \nabla w_{kj}^{(T)} + \mu_j \Delta\mathbf{w}_{kj}^{(T-1)}$$

where w_{kj} is the partial derivative with respect to the weight between the jth neuron in some layer and the kth neuron in the successive layer.

Weight initialization. We initialized all network weights in the convolutional layers from a normal distribution with zero-mean and a standard deviation of 10^{-2}. Biases were also initialized from a normal distribution, but with mean 0.5 and standard deviation 10^{-2}. In the fully-connected layers, the biases were initialized in the same way as the convolutional layers, but the weights were drawn from a much wider normal distribution with zero-mean and stan- dard deviation 2×10^{-1}.

Learning schedule. Although we allowed for a different learning rate for each layer, learning rates were decayed uniformly across the network by 1 percent per epoch, so that $\eta^{(T)} = 0.99\eta^{(T-1)}$. We found that by annealing the learning rate, the network was able to converge to local minima more easily without getting stuck in the error sur- face. We fixed momentum to start at 0.5 in every layer, increasing linearly each epoch until reaching the value μ_j, the individual momentum term for the jth layer.

We trained each network for a maximum of 200 epochs, but monitored *one-shot validation error* on a set of 320 one- shot learning tasks generated randomly from the alphabets and drawers in the validation set. When the validation error did not decrease for 20 epochs, we stopped and used the parameters of the model at the best epoch according to the one-shot validation error. If the validation error continued to decrease for the entire learning schedule, we saved the final state of the model generated by this procedure.

Hyperparameter optimization. We used the beta ver- sion of Whetlab, a Bayesian optimization framework, to perform hyperparameter selection. For learning sched- ule and regularization hyperparameters, we set the layer- wise learning rate ηj [10−4, 10−1], layer-wise momentum μj [0, 1], and layer-wise L2 regularization penalty – $\lambda j \in [0, 0.1]$. For ∈ network hyperpar∈ameters, we let the size of convolutional filters vary from ∈ 3x3 to 20x20, while the number of convolutional filters in each layer varied from 16 to 256 using multiples of 16. Fully-connected layers ranged from 128 to 4096 units, also in multiples of 16. We set the optimizer to maximize one-shot validation set accu- racy. The score assigned to a single Whetlab iteration was the highest value of this metric found during any epoch **Affine distortions.** In addition, we augmented the train- ing set with small affine distortions For each image pair $\mathbf{x_1}$, $\mathbf{x_2}$, we generated a pair of affine trans-formations T_1, T_2 to yield $\mathbf{x}'\mathbf{1} = T_1(\mathbf{x_1})$, $\mathbf{x2}' = T_2(\mathbf{x_2})$, where T_1, T_2 are determined stochastically by a multi-dimensional uniform distribution. So for an arbitrary trans- form T , we have $T = (\theta, \rho_x, \rho_y, s_x, s_y, t_x, t_x)$, with θ [10.0, 10.0], ρ_x, ρ_y [0.3, 0.3], s_x, s_y [0.8, 1.2], and t_x, t_y [2, 2]. Each of these components of the transfor- mation is included with probability 0.5.

4. Experiments

We trained our model on a subset of the Omniglot data set, which we first describe. We then provide details with re- spect to verification and one-shot performance.

The Omniglot Dataset

The Omniglot data set was collected by Brenden Lake and his collaborators at MIT via Amazon's Mechanical Turk to produce a standard benchmark for learning from few exam- ples in the handwritten character recognition domain (Lake et al., 2011).1 Omniglot contains examples from 50 alpha- bets ranging from well-established internationallanguages like Latin and Korean to lesser known local dialects. It also includes some fictitious character sets such as Aurek-Besh and Klingon.

The number of letters in each alphabet varies considerably from about 15 to upwards of 40 characters. All charac- ters across these alphabets are produced a single time by each of 20 drawers Lake split the data into a 40 alpha- bet *background set* and a 10 alphabet *evaluation set*. We preserve these two terms in order to distinguish from the normal training, validation, and test sets that can be gener- ated from the background set in order to tune models for verification. The background set is used for developing a model by learning hyperparameters and feature mappings. Conversely, the evaluation set is used only to measure the one-shot classification performance.

Verification

To train our verification network, we put together three dif- ferent data set sizes with 30,000, 90,000, and 150,000 train- ing examples by sampling random *same* and *different* pairs. We set aside sixty percent of the total data for training: 30 alphabets out of 50 and 12 drawers out of 20.

We fixed a uniform number of training examples per alpha- bet so that each alphabet receives equal representation dur- ing optimization, although this is not guaranteed to the in- dividual character classes within each alphabet. By adding affine distortions, we also produced an additional copy of the data set corresponding to the augmented version of each of these sizes. We added eight transforms for each training example, so the corresponding data sets have 270,000, 810,000, and 1,350,000 effective examples.

To monitor performance during training, we used two strategies. First, we created a validation set for verification with 10,000 example pairs taken from 10 alphabets and 4 additional drawers. We reserved the last 10 alphabets and 4 drawers for testing, where we constrained these to be the same ones used in Lake et al. (Lake et al., 2013). Our other strategy leveraged the same alphabets and drawers to generate a set of 320 one-shot recognition trials for the validation set which mimic the target task on the evaluation set. In practice, this second method of determining when to stop was at least as effective as the validation error for the verification task so we used it as our termination criterion.

In the table below (Table 1), we list the final verification results for each of the six possible training sets, where the listed test accuracy is reported at the best validation check- point and threshold. We report results across six different training runs, varying the training set size and toggling dis- tortions.

One-shot Learning

Once we have optimized a siamese network to master the verification task, we are ready to demonstrate the discrimi- native potential of our learned features at one-shot learning. Suppose we are given a test image **x**, some column vector which we wish to classify into one of C categories.

To empirically evaluate one-shot learning performance, Lake developed a 20-way within-alphabet classification task in which an alphabet is first chosen from among those reserved for the evaluation set, along with twenty charac- ters taken uniformly at random. Two of the twenty drawers are also selected from among the pool of evaluation draw- ers. These two drawers then produce a sample of the twenty characters. Each one of the characters produced by the first drawer are denoted as test images and individually com- pared against all twenty characters from the second drawer, with the goal of predicting the class corresponding to the test image from among all of the second drawer's characters. This process is repeated twice for all alphabets, so that there are 40 one-shot learning trials for each of the ten evaluation alphabets. This constitutes a total of 400 one-shot learning trials, from which the classification accuracy is calculated.

The one-shot results are given in Table 2. We borrow the baseline results from (Lake et al., 2013) for compari- son to our method. We also include results from a non-convolutional siamese network with two fully-connected layers.

At 92 percent our convolutional method is stronger than any model except HBPL itself. which is only slightly be- hind human error rates. While HBPL exhibits stronger re- sults overall, our top-performing convolutional network did not include any extra prior knowledge about characters or strokes such as generative information about the drawing process. This is the primary advantage of our model.

MNIST One-shot Trial

The Omniglot data set contains a small handful of samples for every possible class of letter; for this reason, the original authors refer to it as a sort of "MNIST transpose", where the number of classes far exceeds the number of training instances (Lake et al., 2013). We thought it would be in- teresting to monitor how well a model trained on Omniglot can generalize to MNIST, where we treat the 10 digits in MNIST as an alphabet and then evaluate a 10-way one-shot classification task. We followed a similar procedure to Omniglot, generating 400 one-shot trials on the MNIST test set, but excluding any fine tuning on the training set. All 28x28 images were upsampled to 35x35, then given to a reduced version of our model trained on 35x35 images from Omniglot which were downsampled by a factor of

3. We also evaluated the nearest-neighbor baseline on this task.

The nearest neighbor baseline provides similar performance to Om- niglot, while the performance of the convolutional network drops by a more significant amount. However, we are still able to achieve reasonable generalization from the features learned on Ominglot without training at all on MNIST.

5. Conclusions

We have presented a strategy for performing one-shot clas- sification by first learning deep convolutional siamese neu- ral networks for verification. We outlined new results comparing the performance of our networks to an exist- ing state-of-the-art classifier developed for the Omniglot data set. Our networks outperform all available baselines by a significant margin and come close to the best num- bers achieved by the previous authors. We have argued that the strong performance of these networks on this task indi- cate not only that human-level accuracy is possible with our metric learning approach, but that this approach should extend to one-shot learning tasks in other domains, especially for image classification.

Two sets of stroke distortions for different characters from Omniglot. Columns depict characters sampled from differ- ent drawers. Row 1: original images. Row 2: global affine trans- forms. Row 3: affine transforms on strokes. Row 4: global affine transforms layered on top of stroke transforms. Notice how stroke distortions can add noise and affect the spatial relations between individual strokes.

this paper, we only considered training for the verification task by processing image pairs and their distortions using a global affine transform. We have been experimenting with an extended algorithm that exploits the data about the individual stroke trajectories to produce final computed distortions. By imposing local affine transformations on the strokes and overlaying them into a composite image, we are hopeful that we can learn features which are better adapted to the variations that are commonly seen in new examples.

Reference

1. E. G. Pintelas, T. Kotsilieris, I. E. Livieris, and P. Pintelas, “A review of machine learning prediction methods for anxiety disorders,” in Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-Exclusion—DSAI 2018, Thessaloniki, Greece, 2018. View at: Publisher Site | Google Scholar

2. G. Cho, J. Yim, Y. Choi, J. Ko, and S.-H. Lee, “Review of machine learning algorithms for diagnosing mental illness,” Psychiatry Investigation, vol. 16, no. 4, pp. 262–269, 2019. View at: Publisher Site | Google Scholar

3. R. B. Rutledge, A. M. Chekroud, and Q. J. Huys, “Machine learning and big data in psychiatry: toward clinical applications,” Current Opinion in Neurobiology, vol. 55, pp.152–159, 2019. View at: Publisher Site | Google Scholar

4. Y. T. Jo, S. W. Joo, S. H. Shon, H. Kim, Y. Kim, and J. Lee, “Diagnosing schizophrenia with network analysis and a machine learning method,” International Journal of Methods in Psychiatric Research, vol. 29, no. 1, 2020. View at: Publisher Site | Google Scholar

5. S. Srinivasagopalan, J. Barry, V. Gurupur, and S. Thankachan, “A deep learning approach for diagnosing schizophrenic patients,” Journal of Experimental & Theoretical Artificial Intelligence, vol. 31, no. 6, pp. 803–816, 2019. View at: Publisher Site | Google Scholar

6. W. H. L. Pinaya, A. Mechelli, and J. R. Sato, “Using deep autoencoders to identify abnormal brain structural patterns in neuropsychiatric disorders: a large-scale multi-sample study,” Human Brain Mapping, vol. 40, no. 3, pp. 944–954, 2018. View at: Publisher Site | Google Scholar.