Spatially Biased Random Forests

Benjamin R. Mitchell
Department of Computing Sciences
Villanova University
Villanova, PA 19085
benjamin.r.mitchell@villanova.edu

John W. Sheppard
Gianforte School of Computing
Montana State University
Bozeman, MT 59714
john.sheppard@montana.edu

Abstract

Recent successes in deep learning have led to explorations of what makes these techniques so powerful. One goal of such studies is to determine whether such properties can be transferred to alternative learning methods and yield similar benefits. Since the generalization power of any learning algorithm depends upon the inductive bias(es) of that algorithm, we hypothesize that utilizing a bias incorporated by CNNs and other deep methods—spatial locality—can benefit other learning methods as well. We test this hypothesis by incorporating spatial structure when constructing random forests. Our experiments demonstrate that incorporating a spatial locality bias improves the performance of random forests on several image classification tasks.

Introduction

Historically, most machine learning algorithms have made no assumptions about the organization of features within a feature vector, other than assuming the ordering is consistent across all samples. This weak assumption makes algorithms more flexible, allowing them to be applied to many kinds of data in a uniform way, regardless of the source or meaning of the data. It has generally been believed that useful statistical relationships implicit in feature ordering will be discovered automatically by the learning algorithm.

In recent years, this belief has been challenged by several results, including the impressive performance of deep learning. Currently, the dominant deep learning systems are derived from convolutional neural networks (CNNs) (LeCun et al. 1990) and long-short term memories (LSTMs) (Hochreiter and Schmidhuber 1997). One of the distinguishing characteristics of these methods is the structured operation of the networks. The basic architecture of CNNs assumes the data come from sampling a lattice with a particular spatial organization (i.e. images or image-like data). LSTMs make a similar assumption, only using temporal rather than spatial structure.

The success of these deep learning methods is reliant on the data obeying these assumptions; if the data lack spatial structure these methods lose their power, as we will demonstrate. It is also easy to demonstrate that most classic machine learning algorithms are not impacted by the ordering of features, a property sometimes referred to as *permutation invariance*. This means that while the more traditional algorithms perform well on a wider range of problems, they are at a disadvantage compared to deep learning methods on problems with structurally organized features.

Permutation invariance can be seen as a result of strong assumptions made about the sampling process, including the assumption that the samples are drawn independently and identically distributed (IID). In order to take advantage of spatial structure we must relax these assumptions, as is done in the field of Spatial Statistics (Cressie 1993) by treating samples differently based on their respective 'sampling locations.' The result is that data drawn from a spatially varying distribution, such as geological or epidemiological data, can be modeled more accurately using spatial statistics.

Given the clear utility that this type of structural information exploited by techniques like CNNs have, it is important to find ways of enabling all types of machine learning algorithms to make use of structure that may be latent in the organization of the feature space. In the past, work has been done to make use of the spatial nature of images to improve the performance of PCA (Mitchell and Sheppard 2012), Restricted Boltzmann Machines (Mitchell, Tosun, and Sheppard 2015), and Deep Belief Networks (Tosun, Mitchell, and Sheppard 2016). All of these techniques are useful in their own right, but because they are not classifiers it is difficult to compare their performance directly to that of techniques like CNNs, which perform both feature extraction and classification. In this paper we demonstrate how spatial structure can be used to improve the performance of classifiers directly, without doing any type of intermediary spatially-aware feature engineering.

We use Random Forests as our base classifier, and show how we can take advantage of spatial structure with relatively minor modifications to the algorithm. Our goal is to improve our ability to take advantage of spatial information if it is present without harming performance if it is not. We test this ability using image data, which we will show contains useful spatial structure by means of statistical analysis and empirical testing. The results show that the introduction of a spatial locality bias results in a significant increase in the performance of Random Forests for image classification, that is not vulnerable to the performance penalty seen in CNNs in the permutation invariant setting.

Background and Related Work

The topic of deep learning has seen an explosion of interest in the past few years, both in the machine learning community and in the media at large. Historically, deep learning has mostly been applied to computer vision problems (i.e., learning from digital images), but these days deep learning is being applied to problems in a wide range of other fields, including speech recognition (Dahl et al. 2012), linguistics (Collobert et al. 2011), bioinformatics (Baldi 2014), and game AI (Silver et al. 2016).

Deep learning developed as part of the field of connectionist models as a way of harnessing the increased representational power that arises from the composition of nonlinear functions, and also as an attempt to model the neural architecture of the primary visual cortex. By adding extra layers to a multilayer perceptron, for instance, a greater range of functions can be encoded efficiently, and more abstraction is possible. Unfortunately, this theoretical advantage has historically proven difficult to realize in practice.

The techniques that have come to be labeled *deep learning* are methods of getting around this problem, enabling networks with many layers to be trained efficiently. There are now a variety of different deep learning algorithms that have been proposed, but the vast majority still fit this simple description (LeCun, Bengio, and Hinton 2015).

Random Forests are an ensemble method using decision trees trained with some sort of constraint that makes them function as "weak" classifiers (Ho 1995; 1998; Kleinberg 1996). There are a number of ways to build a random forest; the only fundamental requirement is that the decision trees be built in a stochastic way to ensure sufficient diversity to gain the benefits of an ensemble. In Breiman's classic formulation (Breiman 2001), a random forest is an ensemble of decision trees, each of which is trained using some random "parameter" vector Θ. In his basic formulation, Breiman used a bagging scheme such that each tree was trained on a subset of examples, and each node was trained using a subset of the features. Breiman described several variants, including Random Index Forests (Forest-RI) in which each split uses a single feature, and Random Combination Forests (Forest-RC) in which each split used a random combination of features

Tomita et al. (Tomita, Maggioni, and Vogelstein 2015) presented a generalized algorithm for Projective Random Forests (Forest-RP) that both encompasses most previously described random forest variants and allows for new variants. The basic idea is that, rather than selecting a single feature (as in Forest-RI) or a fixed number of features (as in Forest-RC) to use as a splitting criterion, we can use any projection down to a scalar value. Algorithm 1 shows pseudocode for this algorithm.

Algorithm 1 Forest-RP (Tomita, Maggioni, and Vogelstein 2015)

Input: data: $\mathbf{D} = (\mathbf{x_i}, y_i) \in (\mathbb{R}^p \times Y)$ for $i \in [n]$, tree rules, distributions on projection matrices: $\mathbf{A} \sim f_A(D)$,
Output: decision trees, predictions

for each tree **do**
  Sample training data to obtain a bag $(\bar{X}, \bar{y})$
  Create a root node with this bag as its sample set
  for each leaf node **do**
    Let $\tilde{\mathbf{X}} = \mathbf{A}^\top \bar{\mathbf{X}}$, where $\mathbf{A} \sim f_A(D)$
    Find the "best" split coordinate k^* in $\tilde{X}$
    Find the "best" split value $t^*(k^*)$ for k^*
    Split $\mathbf{X}$ according to whether $\tilde{\mathbf{X}}_i > t^*(k^*)$
    Assign child nodes as a leaf or terminal
  end for
end for

Spatially Biased Random Forests

To test our hypothesis that a spatial bias can help classifiers, we chose Random Forests (RFs) as the base learning algorithm. In particular, we modified the type of random-projection random forest described by Tomita (Tomita, Maggioni, and Vogelstein 2015) as Randomer Forests. For the sake of clarity and consistency, we use Breiman's terminology (Breiman 2001) and call the basic, single-index random forest algorithm Forest-RI (i.e., Forest using Random Indexes). We refer to the Randomer Forest algorithm as Forest-RP (Forest using Random Projections), and our novel spatially-biased algorithm as Forest-RS (Forest using Random Structured projections).

Forest-RP (Algorithm 1) works by using random projections of the data to generate node-splitting candidates. Specifically, candidate projections are created by choosing features at random from an n dimensional feature vector, with each feature having a likelihood of $1/k$ of being chosen for some $k \ll n$. Each feature is then assigned a random weight in the range $[-1, 1]$. The scalar projection value is then considered as a candidate for splitting the tree node.

Forest-RS is similar to Forest-RP but modifies the way in which the random projections are generated. Specifically, where in Forest-RP, features are chosen according to a uniform distribution, in a Forest-RS, features are chosen according to a spatially-biased distribution.

For spatial data (such as images), each candidate projection has a "center" location sampled from a uniform distribution across each spatial dimension of the input. This location is then used as the mean of a Gaussian distribution from which features are sampled. The variance of the Gaussian is a function of the tree-depth of the node: as depth increases, so does variance (the variance is also scaled based on the size of each dimension, in the case of not all spatial dimensions being equal in size). Note that samples drawn from the Gaussian are also subject to the bounds of the edges of the image. The effect is that decision nodes near the root of a tree will tend to have projections based on features tightly clustered in a small spatial region. As tree depth increases, the features will tend to become more widely spread out until eventually there is little spatial bias remaining, and the sampling distribution begins to resemble a uniform one again. Since the center locations are still sampled uniformly, on average the overall forest will spread its feature-samples across the data space uniformly, but the sampling becomes heteroscedastic due to the per-node spatial bias.

Algorithm 2 Forest-RS.

Input: Spatial stride vector $\mathbf{v} \in \mathbb{N}^n$, tree depth of the current node d, scaling parameters α and β
Output: Output projection $\mathbf{f}$

1: $\mathbf{f} \leftarrow \mathbf{0}$
2: **for** each non-zero component **do**
3: **for** $i = 1$ to n **do**
4: $\mu \sim \text{Uniform}(1, v_i)$
5: $\sigma^2 \leftarrow (\alpha_i + (\beta_i \cdot d)) \cdot v_i$
6: **repeat**
7: $idx_i \sim \text{Gaussian}(\mu, \sigma^2)$
8: **until** $1 \leq idx_i \leq v_i$
9: **end for**
10: $o \leftarrow idx_1$
11: **for** $i = 2$ to n **do**
12: $o \leftarrow o \cdot v_{i-1}$
13: $o \leftarrow o + idx_i$
14: **end for**
15: $f_o \sim \text{Uniform}(-1.0, 1.0)$
16: **end for**
17: **return** $\mathbf{f}$

Algorithm 2 shows how to create a candidate projection $\mathbf{f}$ (used in line 5 of Algorithm 1), which can be applied to the data vectors to produce a candidate split. As with all random forests, many candidates will be generated for each node, with the best candidate chosen according to a subset purity measure. Note the stride vector $\mathbf{v}$ is determined by the data; for example, the SVHN dataset contains 32×32 pixel images with 3 color values per pixel, so $\mathbf{v} = \{32, 32, 3\}$ and $n = 3$. At the root of the tree, the sampling process in this algorithm tends to focus "attention" on a particular spatial region. This encourages discovery of spatially local statistical relationships in the data. By gradually increasing variance, the spatial region size sampled increases. This allows the trees to capture non-local and conditionally local statistical relationships at deeper nodes.

Experiments

We used the standard image classification benchmarking datasets, MNIST (LeCun, Cortes, and Burges 2019) and SVHN (Netzer et al. 2019) for our experiments, since CNNs are known to perform well on these problems. This allows us to examine the impact of spatial structure on CNNs in a setting favorable to them and to examine the impact of spatial biasing on our Random Forest. Our analysis consisted of three steps. First, we used mean correlation plots to examine the spatial structure in the datasets. Second, we compared the performance of CNNs on original data and permuted data. Third, we compared the performance of different types of Random Forests on original and permuted data.

Randomly Permuted Datasets

Since we wanted to evaluate the ways different learning algorithms interacted with spatial structure, we created "destructured" versions of each dataset. To create these destructured data sets, we generated random permutations and applied them to the feature order of the data vectors. Note that the same permutation is applied to all the vectors in a given dataset, including both training and testing samples (Figure 1).

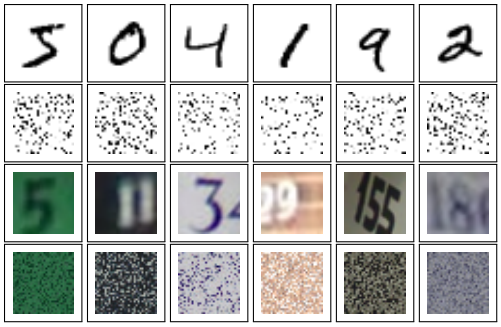

Figure 1: Original and randomly permuted image examples. Rows 1 and 2 show MNIST. Rows 3 and 4 show SVHN.

The purpose of the permutation is to disrupt any spatial structure that might exist while leaving non-spatial structure intact. An algorithm that does not consider spatial structure should perform the same on both the original and the permuted dataset. An algorithm that does rely on spatial structure, however, should show lower performance when applied to the permuted data instead of the original data.

Locality Analysis

For each dataset, we generated spatial information plots as described in (Mitchell, Tosun, and Sheppard 2015). Here, we show the mean-correlation plots for both the original and the permuted versions of each of the three datasets. To generate mean-correlation plots, we calculated $corr(X^i, X^j)$ for all pairs (i, j), where X^i is a vector containing the value of feature i for each training vector. For each distance, we then plotted the mean correlation of feature pairs that distance apart.

Figure 2 shows the mean-correlation plots for both the original data and for the randomly permuted versions of the MNIST and SVHN datasets. Notice in Figure 2 at distance 0, there is perfect correlation since any feature always has the same value as itself. Beyond that, however, the correlation plot for the permuted dataset is basically flat, indicating that after permutation there remains no significant relationship between statistical information and spatial distribution for feature pairs. The deviations from this trend in the last few points in each plot are caused by the reduced sample size available for the extreme distances.

The plots for the original data, on the other hand, exhibit some interesting structure. The first thing to note is that for all datasets, adjacent features have high correlation. As the feature-pairs get farther apart, the degree of correlation drops off, but the rate differs between the datasets.

The MNIST data show the most rapid falloff; by a distance of 5 pixels, there remains on average little correlation between feature pairs. This is likely related to the line-width of the hand-written digits, which is generally 2-3 pixels. The

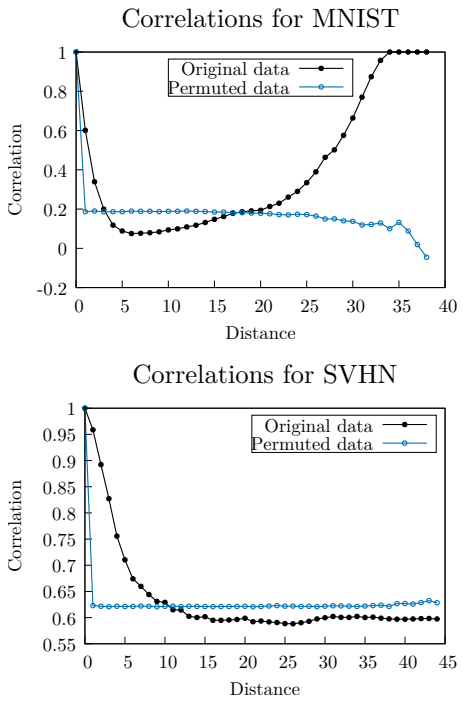

Figure 2: Correlation vs. pixel coordinate distance, before and after random permutation.

fact that correlation improves again as distances increase is an artifact of the MNIST images all having a centered digit surrounded by a white border. Since background pixels have the same value in *all* images, correlation scales with the likelihood that both members of a pair are background.

The SVHN data show a falloff that is slower than for MNIST but still has a relatively steep slope for the first 5-10 pixel-units of distance. The backgrounds tend to be somewhat regular within any given image, since they tend to be mostly paint, shingle, brick, or whatever surface the house numbers are attached to, but they lack the universal background color seen in MNIST. As a result, the mean correlation does not go back up for pairs of background pixels.

Taken together, these results demonstrate that image data exhibits spatially localized structure, that this structure can be measured, and that it can be eliminated by randomly permuting the ordering of the features in a dataset.

Convolutional Neural Networks

We assess the impact of spatially local structure on CNNs by comparing their behavior in the presence and absence of that structure. As a baseline, we used a multi-layer perceptron (MLP), which should be permutation invariant. We compared the behavior of CNNs with that of densely connected networks on both original and permuted versions of the MNIST and SVHN datasets.

A method that makes no use of spatially local information should perform identically on both original and permuted versions of the data. For techniques that rely on spatial information, permuting the data should result in lower performance than on the original data. It should be noted that better performance for these techniques is reported elsewhere in the literature; our goal was not to achieve state-of-the-art performance, but rather to observe how performance is impacted by removing spatial information.

Table 1: Classification accuracy of neural network variants; "-p" indicates permuted data.

	MLP	MLP-p	CNN	CNN-p
MNIST	98.46%	98.11%	99.41%	97.13%
SVHN	79.20%	78.37%	89.47%	72.70%

The CNN we trained for the MNIST dataset used 4 convolutional layers, each followed by a pooling layer. The network was topped with two fully-connected layers, with 512 and 128 units respectively. The MLP trained for MNIST was architecturally identical to the two-layer output network used for the CNN: 512 units in the first hidden layer, 128 in the second.

For SVHN, we used a VGG-like CNN (Simonyan and Zisserman 2015), with a total of 11 convolutional layers and 5 pooling layers, topped by a fully-connected layer with 512 nodes. The MLP architecture for SVHN used two hidden layers, containing 4092 and 2048 units.

All models were trained using dropout (Hinton et al. 2012) and batch normalization (Ioffe and Szegedy 2015). We trained each model on both the original and permuted version of each data set. In each case, the provided training and testing sets were used for training and evaluation. Each experiment was replicated 10 times, and the reported results represent means over these 10 samples. In Table 1, we see that CNNs are able to outperform MLPs on original image data by wide margins, as expected. To test for statistical significance, we applied a Wilcoxon rank sum test to each pair of experimental conditions, with the null hypothesis in each case being that the two conditions were equivalent.

With all data sets, we find that the MLPs behave the same whether the data have been permuted or not ($p \gg 0.05$). This is a strong indication that the MLPs are not making use of any spatial structure. The CNNs, on the other hand, show significantly worse performance on the permuted data than on the original data ($p < 0.002$ in all cases). Indeed, the CNN on permuted data performs significantly worse than the MLP ($p < 0.003$ in all cases). In the case of the MNIST data, where the same architecture as the MLP was used as the output network for the CNN, this indicates that not only is the CNN making use of spatial information when it is present, it is reliant on the spatial information to such an extent that removing the spatial data makes the convolutional layers of the CNN perform worse than the identity function.

Spatially Biased Random Forests

Experiments were performed using basic, single-index random forests (Forest-RI), projective random forests (Forest-RP), and spatially biased random forests (Forest-RS); several meta-parameters were examined, and values were chosen empirically. Experimental results are reported for the MNIST and SVHN datasets, and original and permuted ver-

Table 2: Classification accuracy for random forest variants.

	MNIST	SVHN
Forest-RI	0.967	0.701
Forest-RP	0.971	0.706
Forest-RS	0.974	0.722

sions of the MNIST dataset. For each dataset, we used the provided test-train split, to allow easy comparison of our results to those obtained by others. In all cases our performance measure was classification accuracy on the test set.

All versions of the forest learning algorithm used a set of c candidate splitting projections, using either single randomly-selected features (Forest-RI), unbiased random projections (Forest-RP), or spatially biased random projections (Forest-RS). For the latter two, each projection had k non-zero components on average. For each candidate, the data were sorted according to the projected value, and midpoints between each pair of adjacent values were considered as possible split points. The optimal split point was determined by computing information gain.

We used a grid search to find hyperparameter values for our algorithms. For the parameters shared by multiple algorithms, all algorithms showed the same trends. This allowed us to use the same values for all versions, resulting in a fair comparison without disadvantaging any algorithm.

For all types of forests, each tree was trained using 70% of the training examples selected at random. This is a similar number to what others have suggested is reasonable (Breiman 2001). For all types of forests, increasing the number of candidates splits c at each node tended to improve performance up to some point of diminishing returns; we used a value of $c = 200$.

For the MNIST dataset, the minimum number of examples needed before a node was split ($n_{\min}$) was set to 1, meaning the algorithm ran until nodes were pure or no candidate split was able to reduce entropy. Higher values were not found to help generalization. For the SVHN dataset we used a value of 3 for reasons of computational efficiency.

For both Forest-RP and Forest-RS, the average number of non-zero projection components k was set to 3 as this seemed optimal for both techniques on our data. Higher values were computationally more expensive, and significantly higher values were found to negatively impact the generalization of both techniques.

For Forest-RS, the variance was set as:

$$\sigma^2 = (0.05 + (0.015 \cdot d)) \cdot w$$

where d is the tree depth of the current node, and w is the width of the current image dimension.[1] Coordinates sampled outside the image boundaries were re-sampled until valid coordinates were obtained.

Table 2 shows the classification accuracy for each ensemble. The ensemble sizes used were 1008 for MNIST and 384 for SVHN, enough for performance to saturate (see Figure 3). In all cases, Forest-RS outperforms Forest-RP, which in turn outperforms Forest-RI. For the null hypothesis that two techniques were equivalent, a Wilcoxon test yielded $p \leq 2 \times 10^{-5}$ for all pairs.

[1] All images used were square, so width = height.

Table 3: Classification accuracy for random forest variants on original and permuted data.

	MNIST-orig	MNIST-perm
Forest-RI	0.9687	0.9684
Forest-RP	0.9698	0.9688
Forest-RS	0.9711	0.9687

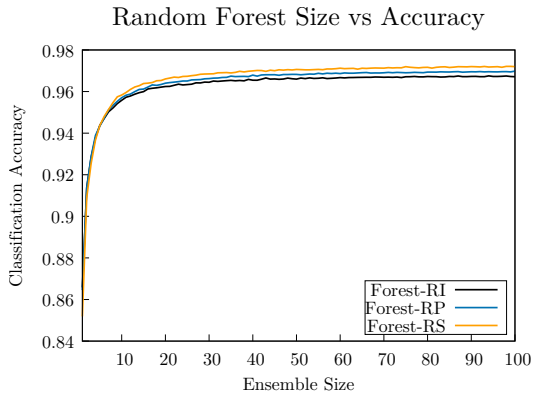

Figure 3: Classification accuracy vs number of trees in the forest for the MNIST dataset.

Table 3 shows the classification accuracy for each RF variant when trained on original and permuted versions of MNIST; ensembles of 64 trees were used since only relative performance is of interest. In contrast to the results using CNNs (Table 1), it can be seen that in the absence of spatial structure, Forest-RS does no worse than Forest-RP; in fact, the two become equivalent ($p > 0.8$ over 10 replications). This makes sense mathematically, since with randomized feature locations both methods are effectively picking features from a uniform distribution with respect to the original feature locations.

Figure 3 shows the performance of ensembles of varying sizes on MNIST. For each ensemble size, 20 sample forests were tested, and their performance was averaged. This plot demonstrates that the ranking of the three methods is consistent. It can also be seen that mean accuracy is an increasing function of ensemble size, but the benefit tails off after a certain point. This is in line with other work on random forests, which often suggests that there is no upper bound to how many trees can productively be combined. In particular, the claim is sometimes made that random forests are "immune to overfitting," though this refers to the number of members in the ensemble, not the strength of each base classifier.

It is worth noting that for very small ensembles (less than 5 trees), Forest-RS does not outperform the other methods. This suggests that part of the performance advantage of the Forest-RS technique may be that it helps act as a regularizer; individual trees are slightly weaker but have greater indepen-

dence, resulting in more power from combining them.

Conclusions

Overall, our results demonstrate that it is possible to introduce a spatial bias into random forests, and that incorporating such a bias improves the performance of random forests on spatial data. Additionally, the weaker bias relative to CNNs means that applying this method to randomly permuted data is essentially equivalent to the Forest-RP algorithm; regardless of what structure is present, its performance will never fall below that of the baseline method. This suggests that this type of bias is more robust to the possible absence of spatial structure, which is a potential advantage over CNNs and other fixed-partition methods.

Even with random projections, we acknowledge that RFs, with or without spatial bias, are not currently good candidates for tasks requiring certain types of invariance such as some types of vision tasks. For example, CNNs display a degree of spatial shift invariance that RFs cannot currently provide. Our goal here was to show that RFs can be made to take advantage of spatially local information, thus potentially broadening their applicability to other types of problems. We view the introduction of additional types of invariance and regularization to be an important direction for future work.

Acknowledgments

This work was made possible by the resources of the Maryland Advanced Research Computing Center (MARCC), and a donation from NVidia. We are grateful to conversations with Joshua Vogelstein and Tyler Tomita for helping to motivate this work. Parts of this work first appeared in the first author's dissertation (Mitchell 2017).

References

Baldi, D. C. P. S. P. 2014. Deep autoencoder neural networks for gene ontology annotation predictions. In *Proceedings of ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM BCB)*, 533–540.

Breiman, L. 2001. Random forests. *Machine Learning* 45(1):5–32.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 2493–2537.

Cressie, N. 1993. *Statistics for Spatial Data*. Wiley-Interscience.

Dahl, G.; Yu, D.; Deng, L.; and Acero, A. 2012. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. In *IEEE Transactions on Audio, Speech, and Language Processing*, 30–42.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *Computing Research Repository (CoRR)*.

Ho, T. 1995. Random decision forests. In *Proceedings of International Conference on Document Analysis and Recognition (ICDAR)*, 278–282.

Ho, T. 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 20(8):832–844.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, 448–456.

Kleinberg, E. 1996. An overtraining-resistant stochastic modeling method for pattern recognition. *The Annals of Statistics* 24(6):2319–2349.

LeCun, Y.; Bengio, Y.; and Hinton, G. E. 2015. Deep learning. *Nature* 521:436–444.

LeCun, Y.; Matan, O.; Boser, B.; Denker, J.; Henderson, D.; Howard, R.; Hubbard, W.; Jackel, L.; and Baird, H. 1990. Handwritten zip code recognition with multilayer networks. *Advances in Neural Information Processing Systems*.

LeCun, Y.; Cortes, C.; and Burges, C. 2019. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

Mitchell, B. R., and Sheppard, J. W. 2012. Deep structure learning: Beyond connectionist approaches. In *Proceedings of International Conference on Machine Learning and Automation (ICMLA)*, 162–167.

Mitchell, B. R.; Tosun, H.; and Sheppard, J. W. 2015. Deep learning using partitioned data vectors. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*.

Mitchell, B. R. 2017. *The Spatial Inductive Bias of Deep Learning*. Ph.D. Dissertation, The Johns Hopkins University.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. 2019. The street view house numbers (svhn) dataset. http://ufldl.stanford.edu/housenumbers.

Silver, D.; Huang, A.; Maddison, C.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529:484–489.

Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of Interantional Conference on Learning Representations (ICLR)*.

Tomita, T. M.; Maggioni, M.; and Vogelstein, J. T. 2015. Randomer forests. *Computing Research Repository (CoRR)*.

Tosun, H.; Mitchell, B. R.; and Sheppard, J. W. 2016. Assessing diffusion of spatial features in deep belief networks. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, 1625–1632.