Enhancing Neural Networks with Locality-Sensitive Clustering of Internal Representations

Richard A. McAllister
Gianforte School of Computing
Montana State University
Bozeman, MT, USA
richard.mcallister@student.montana.edu

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

***Abstract*—Some data exhibit natural divisions where the application of a single neural network leaves some accuracy on the table, thereby making a multi-network approach more appropriate. We develop an approach to preserving knowledge encoded in the hidden layer of several ANN's and assemble that knowledge in new, composite networks based on spatial clustering that more effectively make predictions over subdivisions of the entire dataspace. We show that this method has an accuracy advantage over the single-network approach.**

I. Introduction

Currently, substantial work is being done in the Artificial Neural Network (ANN) community that addresses the black box nature of neural networks. Clarifying the inner workings of ANN's would yield knowledge about why these networks make the decisions that they make. If this knowledge could be harvested during model training then this knowledge could be used in other areas, enhancing accuracy, training time, or versatility. Our study concentrates on preserving accumulated knowledge as an intermediate step in clarifying the inner workings of ANN's rather than interpreting the model. Here, we identify subsets of internal nodes from networks that have been trained to make effective predictions on spatially related sets of data.

A variety of problems exist where one might consider training a single model to perform predictions of the associated dataspace. Unfortunately, situations exist where a single model to cover the space is suboptimal; it would be better to train several models covering subsets of the space, albeit not as an ensemble where predictions are combined. These situations often arise when the data is spatial or temporal in nature. The challenge is then to determine the relevant subsets of the data for a particular model to cover, efficiently training the set of models, and then determining which model to apply at runtime.

In this paper, we proceed from the perspective of working specifically with spatial data[1] We explore how to preserve information encoded in the hidden layer of ANN's trained in specific spatial regions and then identify key hidden nodes in networks to be used to train networks in spatially similar regions. The main question being addressed is whether it is possible to preserve knowledge encoded in the hidden layer of several ANN's and collect that knowledge in a network that can effectively make predictions over a more granular subdivision of the entire dataspace.

The remainder of this paper is organized as follows: In the next section we discuss some background information relevant to our task, including some related work and a characterization of the data that were used here. Section III details the approach we take as well as summarizing the common concepts that we utilize in our approach. Our experiments are described after this in Section IV with the results presented in Section V. Finally, in the last section, we present our conclusions based on the results and outline directions for future work.

II. Background

A. Related Work

In this work we use autoencoders to pre-condition our hidden layer via greedy unsupervised pre-training [1], [2]. Autoencoders perform feature extraction by exploiting a property of being "undercomplete," meaning that they ideally capture the most salient features of the training data at a lower dimensionality [3]. Networks pre-trained in this manner have been credited with causing the renewed interest in deep neural networks starting in 2006 [3]–[6].

The data that we are interested in is spatiotemporal functional data, which is data that exist in a continuous space whose changes happen as a function of space and time. The field of functional data analysis is characterized and explored by Silverman and Ramsay in [7]. A critical characteristic of this data is its tendency to vary in a continuous, non-abrupt manner.

Layerwise Relevance Propagation (LRP) is a procedure for removing some of the "black box" characteristics of Artificial Neural Networks [8]–[10]. Originally, LRP was intended to attribute contributions of single pixels in an image to classification decisions that a neural network made. It has also been used to compute scores for regions of images "denoting the impact of the particular image region on the prediction of a classifier" [11].

Partitioning of the dataspace was addressed by Rakhmatova, et al. in [12]. The authors cite differences in relief, soil type, terrain, and anthropogenic effects causing the data to

[1]Technically, the data is spatiotemporal; however, we train over the entire temporal range, thus abstracting out the temporal relationships.

TABLE I
FEATURES FOR THE HURRICANE SANDY DATASET

Reading Source	Reading Name
Radiometry Measurement	Temperature
	Pressure
	Cloud Density
	Rain Density
	Ice Density
	Snow Density
	Graupel Density
Wind Speed Indicator	Wind u (East/West)

be "spatially inhomogenous" as a motivation for their work. This is consistent with our motivations in this paper. However, though they address pre-partitioning a dataset for greater accuracy in the application of ANN's, their method mainly addresses optimally splitting the data for training and test purposes, rather than to create a multiplicity of networks to more accurately make predictions on their own subdivisions, as is done here.

In [13] Sergeev, et al. refer to processes that cause "spatial heterogeneity" in studying the spatial distribution of topsoil components. They used spatial components as inputs to their procedure, allowing the ANN's that were trained on the data to learn how the spatial locations affected the function being simulated. Our approach does not rely on the ANN's to do this and instead uses separate networks for the contiguous related regions identified via the two stages of clustering detailed below.

B. Data

For this study, we used the Microwave CubeSat fleet simulation dataset created by Zhang and Gasiewski [14]. This data depicts radiometric and wind readings from one day over Hurricane Sandy, an Atlantic Ocean hurricane in 2012. The average spatial resolution of the data is 5 kilometers, which we down-sampled during the spatial binning process to 15 kilometers. The temporal resolution was 15 minutes. The configuration of the data for training and testing purposes was the same as was used in [15].

The particular task that we use to study our procedure is the same as in [15], which is to infer the u component of the storm's wind vector (East-West component) using radiometric data. The data that we use is a hybrid of two datasets (radiometric and wind) that were taken over the same area at the same time. In [15], the authors explore the functional relationship between radiometric data and wind vector data. This is useful to the meteorological community since radiometry does not measure wind directly.

Table I shows the input features that were used and output for these experiments. These are the same features used in [15]; however, we considered all three components in that work (i.e., u, v, and w) but we restrict our attention here to the u component alone.

Algorithm 1 Locally-sensitive hierarchical agglomerative clustering

procedure LSHAC($L, cluster\_limit$)
  for all $l \in L$ **do**
    $c_l \leftarrow l$ ▷ Each location is its own cluster.
  end for
  while $num(C) > cluster\_limit$ **do**
    $l_r \leftarrow rand(L)$
    $c_{l_r} \leftarrow$ cluster containing l_r
    $c\_locs = []$ ▷ c_locs holds closest locs
    for all $l_{c_{l_r}} \in c_{l_r}$ **do**
      $L_a \leftarrow adjacent(l_{c_{l_r}}) \backslash c_{l_r}$
      $c\_locs \leftarrow c\_locs + \underset{l_a \in L_a}{\text{argmin}}(distance(l_a, l_{c_{l_r}}))$
    end for
    $c_{l_r} \leftarrow c_{l_r} + c\_locs$
  end while
end procedure

III. APPROACH

A. Overview

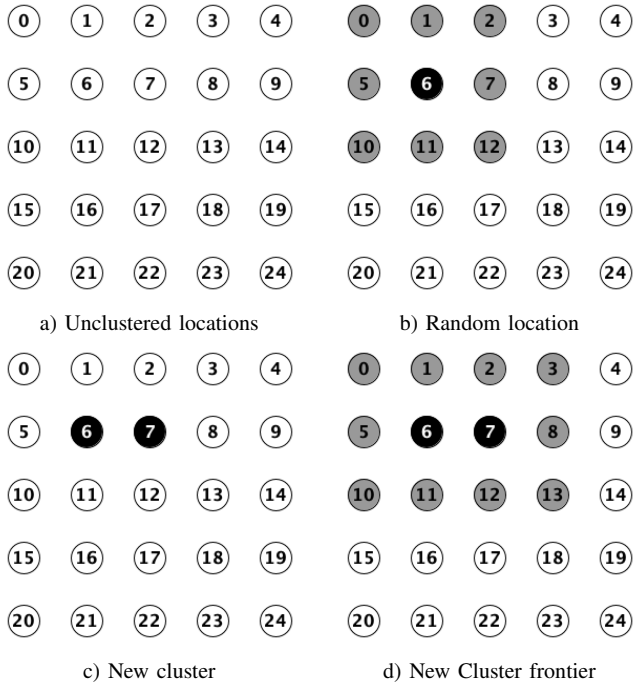
The general approach we take is to divide the dataspace into contiguous clusters based on two types of representations. The clustering is done in two phases: primary and secondary. The primary clustering uses the original training data (data instance clustering) to provide a first-look agglomeration of areas that have similar data. The secondary clustering (LRP clustering) uses the LRP values of the hidden nodes of a stacked autoencoder-based multilayer perceptron (MLP) that has been trained on the data from the clusters in the primary clustering. The LRP clustering captures similar relevance values from these MLP's to provide a basis for hidden-node generalization.

Our approach to knowledge transfer among networks is to identify nodes in the hidden layer that have been most significant in producing correct predictions. These nodes are then combined with other nodes from other networks within their spatial cluster to form a hybrid network. Since we are performing regression, we are interested in studying the effect of both excitatory and inhibitory neurons.

B. Computational Components

1) Spatial Data Clustering: To exploit functional relationships in the data, we use a Locality-Sensitive Hierarchical Agglomerative Clustering algorithm (LSHAC). This proceeds from the assumption that points of interest that are close to one another (spatially) behave functionally with a greater degree of similarity than points that are further apart. As mentioned in Section I these areas may be influenced by common factors that are relatively alien to other areas. Assuming this, we begin by performing LSHAC on the raw training data points.

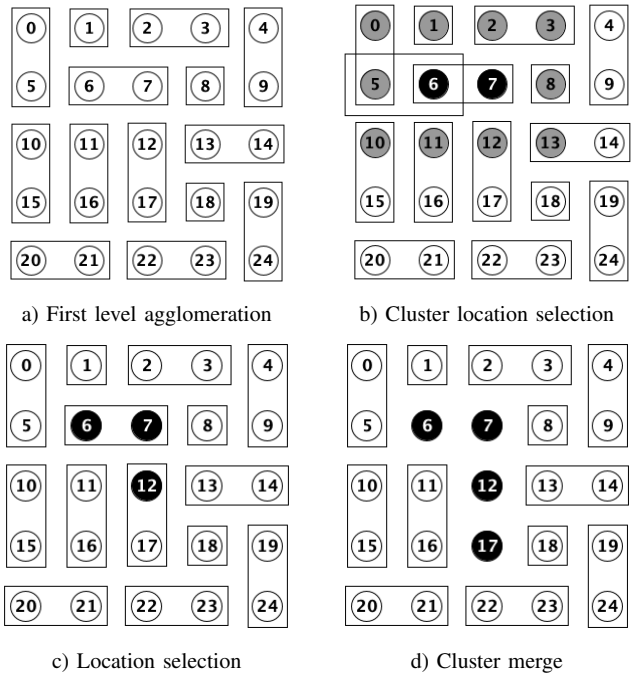
The key difference between LSHAC and normal hierarchical agglomerative clustering is that at each agglomeration stage we only consider the locations neighboring a randomly selected cluster. Note that we refer to these locations as Areas of



a) Unclustered locations b) Random location

c) New cluster d) New Cluster frontier

Fig. 1. Clustering a 5×5 grid of locations



a) First level agglomeration b) Cluster location selection

c) Location selection d) Cluster merge

Fig. 2. Clustering a 5×5 grid of locations (cont'd)

Interest (AOI). We use cosine similarity to compare these neighboring locations to determine which of the neighbors is added to the cluster.

Algorithm 1 shows our procedure for LSHAC. Lines 2 through 4 show that at the beginning each location is its own cluster. The major loop (lines 5 through 14) constrains the number of clusters being formed to the $cluster\_limit$ parameter. In lines 6 and 7 we select a random location and get the cluster corresponding to that location. Lines 9 through 12 iterate through all locations inside this cluster, saving the closest location outside this cluster to each of these interior locations. The cluster being examined (c_{l_r}) is then merged with the external cluster containing the location that is closest to any location inside c_{l_r} on line .

Figure 1 depicts the initial stage of clustering using 25 points in a five by five grid. At this stage, no clustering decisions have been made, and each location is treated as its own cluster. For illustration purposes we select location 6 arbitrarily as a starting point (Figure 1b). The surrounding locations of location 6, shown in grey, are identified, and each of their cosine similarities with location 6 is calculated. Now suppose that location 7 is determined to be most similar. Figure 1c shows location 7 added to the cluster that includes location 6. Figure 1d then shows the next step by considering the surrounding locations of the cluster containing locations 6 and 7.

For the second stage of clustering, Figure 2 shows that all of the locations have either been paired with another location or are orphaned by not being included in another cluster (e.g., locations 8 and 18). The second round of clustering begins with collecting all of the locations neighboring a randomly selected cluster. We continue the previous example by using the cluster containing locations 6 and 7, with the surrounding location shown in grey. Figure 2c shows that location 12 has been selected as the location with the highest cosine similarity. Analogous to a single-linkage approach to clustering, the entire cluster containing location 12 is now merged, creating a new cluster containing locations 6, 7, 12, and 17. This new cluster is excluded from evaluation until the second stage completes.

2) Unsupervised Pre-Training and Fine Tuning: Each ANN trained here begins by unsupervised pre-training (UPT) of a single autoencoder [16]. We use UPT to find favorable initialization points from which to apply supervised learning with the ground truth data [3].

As was done in [17], a single initialization across all locations was used to preserve comparability. During UPT, all of the training data for a given cluster was used to train an autoencoder, presented in the same temporal order across all networks. After training, the decoder portion of the autoencoder was discarded.

3) Layerwise Relevance Propagation: Because we are interested in the contributions of hidden neurons to train a final regression model, We use LRP at the hidden layer level to determine the most relevant nodes. LRP employs a layer-wise conservation principle, which assumes the preservation of a propagated quantity between adjacent layers of a multilayer perceptron [9]. The conservation principle requires

$$\sum_i R_i^{(l)} = \sum_j R_j^{(l+1)}$$

Algorithm 2 Hidden layer LRP

1: **function** HLLRP($n_{AOI}, D_{AOI,training}, l$)
2: $n_{front,l} = n_{AOI}$ with layer α_l as output
3: $A \leftarrow []$ ▷ A gets a new list.
4: **for** $d \in D_{AOI,training}$ **do**
5: $A[d] \leftarrow n_{front,l}(d)$
6: **end for**
7: $n_{back,l} = n_{AOI}$ with layer α_l as input
8: $\Lambda \leftarrow []$ ▷ Λ gets a new list.
9: **for** $\alpha \in A$ **do**
10: $\Lambda[\alpha] \leftarrow LRP(\alpha)$
11: **end for**
12: **return** Λ
13: **end function**

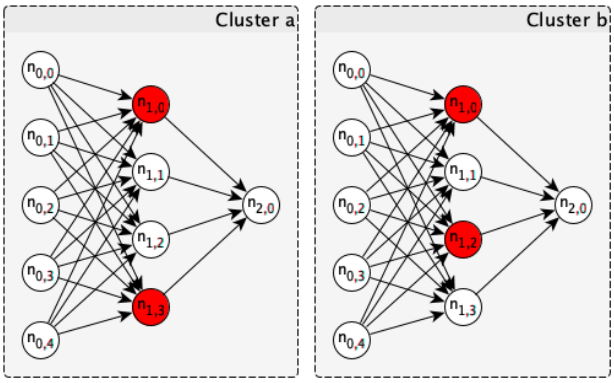


Fig. 3. Intersecting importances

where $R_i^{(l)}$ is the relevance associated to the ith neuron of layer l. The relevance signal is directed proportionally to the subset of inputs that resulted in the corresponding output for each training instance. The relevance score is calculated according to

$$R_i^{(l)} = \sum_j \frac{z_{ij}}{\sum_{i'} z_{i'j} + \epsilon \text{ sign}\left(\sum_{i'} z_{i'j}\right)} R_j^{(l+1)}$$

where

$$z_{ij} = a_i^{(l)} w_{ij}^{(l,l+1)}$$

causing the lower-layer neurons (those closest to the input) that mostly contribute to the activation of a higher layer neuron (closest to the output) to receive a larger share of the associated relevance.

LRP was designed specifically to analyze input data. We were interested in capturing the relevance of the hidden layer activations, so we modified the LRP algorithm to do this (Algorithm 2). On line 2 we create a new network by removing the output layer and using the activations of hidden layer l as the output. We then capture this output A as new input data for the second part of the procedure. On line 7, we create a new network by removing the input layer from the original network n_{AOI} and using A as the input. We perform the LRP procedure on each of the records from the A list to obtain the relevance values Λ for each of the internal nodes in layer l. Notice that we run the complete set of training data through the network once, and then the activations through the network another time.

4) Subset Transfer: We are interested in capturing the most important subsets of the nodes that make up the hidden layers of our networks. Using these subsets, we create a foundation for pre-training new autoencoders. We demonstrate that this yields multi-layer perceptrons that are more accurate predictors for the local area upon whose data the network has been trained.

When we transfer information among networks, we are combining networks that have already been trained to make predictions within their own spatial clusters. Often, corresponding nodes in different clusters' networks have LRP values that are different but still significant, which is demonstrated in Figure 3. Both clusters have produced networks where node $n_{1,0}$ has a significant LRP value.

When we examined the distribution of LRP values, we observed three interesting differences (4). The whisker plot of the LRP values of the nodes from the networks within one cluster (Figure 4a) shows that there is a variety of differences among the nodes, including many with no difference at all (e.g., nodes 0, 3, 6, 9, 15, and 19). Figure 4b shows a histogram of the absolute values of the differences where a large number of nodes have little or no difference with the distribution then trailing off to the right. Figure 4c shows a histogram of both the positive and negative LRP values. To identify nodes with useful LRP values, we employ a knee detection procedure on this final histogram for both increasing (negative) and decreasing (positive) values. Nodes laying outside these knee points were regarded as transferable.

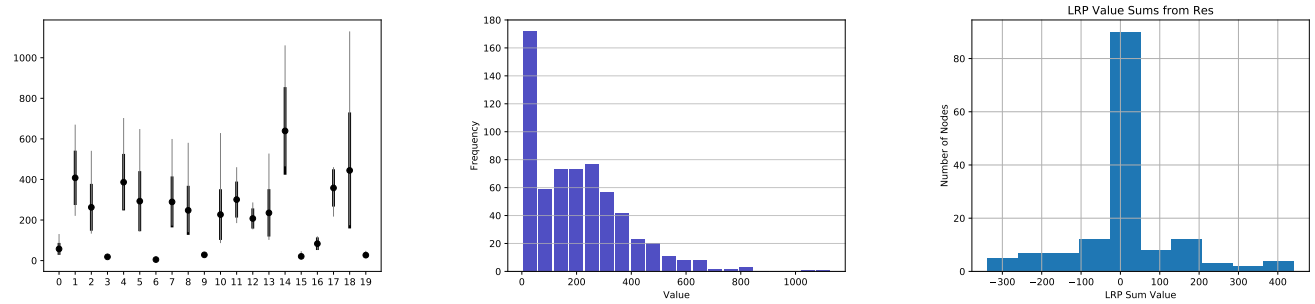
When comparing two networks, if the LRP values of a node in each network are regarded as significant, but one node is inhibitory while the other is excitatory, we split the node in the combined network. Figure 5 illustrates this process. Figure 5a highlights the three relevant nodes. Figure 5b shows that all three have been included in a new autoencoder. The remaining nodes are copied from the untrained template network.

5) Selective Network Application: Figure 6 shows a 5×5 grid that has been divided into four clusters with the centroids of those clusters depicted as the black diamonds.

Suppose we have a location about which we wish to make predictions (e.g., the yellow star in Figure 6). Since centroid 4 is the closest, the network for cluster 4 will be used to make predictions for that location.

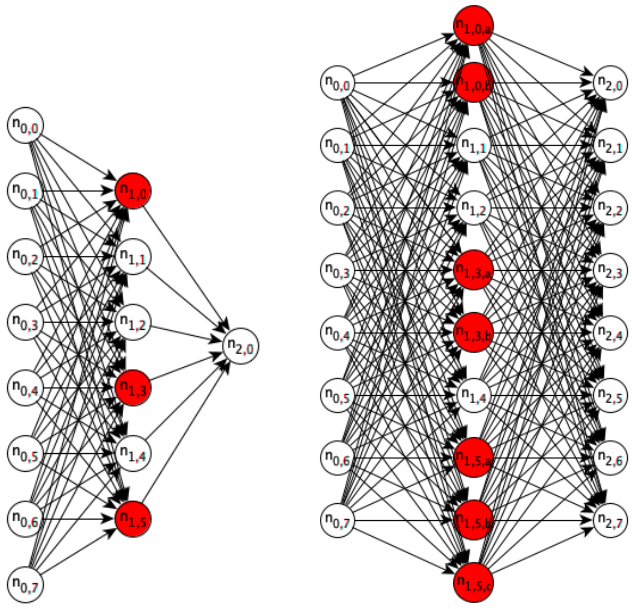
C. Training Procedure

To begin the overall training procedure, spatial data clustering (cf. Section III-B1) is applied to the raw data from each AOI. We call this "data instance clustering." The targeted number of clusters is an input parameter that guides the algorithm, telling it when to stop clustering. The algorithm stops clustering when the number of clusters is at this limit or



a) LRP value differences b) Histogram of absolute values c) Histogram of values

Fig. 4. Examination of LRP value distribution



a) Identification of nodes b) Node expansion

Fig. 5. Node identification and expansion

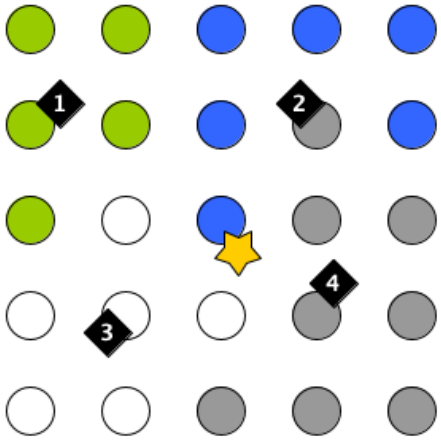


Fig. 6. Spatially clustered areas of interest with query location

when the next agglomeration will cause the number of clusters to be below this limit.

The autoencoders that produce the initial hidden layers for each AOI's network are each copied from a pre-initialized template network. This is done to ensure comparability among each of the networks. This is also important for the aggregation and transfer steps later in the procedure.

Following the Unsupervised Pre-Training, Multilayer Perceptrons (MLP's) are created using the pre-trained layer as the hidden layer. These MLP's are then fine-tuned with the training data. We then apply the LRP procedure to the trained networks to determine the relevant hidden nodes. Following LRP, a second phase of clustering is performed using the relevance values. It is this clustering step that produces the final clusters among which accumulated knowledge is shared. It is also these clusters that are evaluated using the Selective Network Application procedure, which is the final step in the experiment.

IV. Experiments

A. Experimental Design

For training, testing, and verification we employed an 80%/10%/10% strategy where the same time slices are used for each of the splits across all of the networks. These splits were generated via 10-fold cross-validation to obtain a statistical sample of performance. The verification split was used for early stopping to prevent overfitting. The networks were trained to predict wind vector components from the radiometric data collected in the Microwave CubeSat data set.

B. Experiment Scenarios

Table II shows the different scenarios of experiments that were run. The global experiment served as a baseline for performance comparison and consisted of training a single network over all of the AOI's. The dual clustering experiments involved combining both data instance clustering and relevance clustering. The data were clustered using the normalized data instances. For data instance clustering the target cluster

TABLE II
SUMMARY OF EXPERIMENT SCENARIOS

Experiment	Description	Variations
Global	Train one ANN on all data from all locations to predict over the entire dataspace.	N/A
Dual Clustering (DC)	Cluster the dataspace using LSHAC based on the training data. Further cluster the dataspace using HLHAC based on LRP values.	N/A
Dual Clustering with Transfer (DCT)	Same as DC. Combine networks using relevance similarity.	1) Use only above positive knee point. 2) Use only below negative knee point. 3) Use both. 4) Use only relevant nodes.

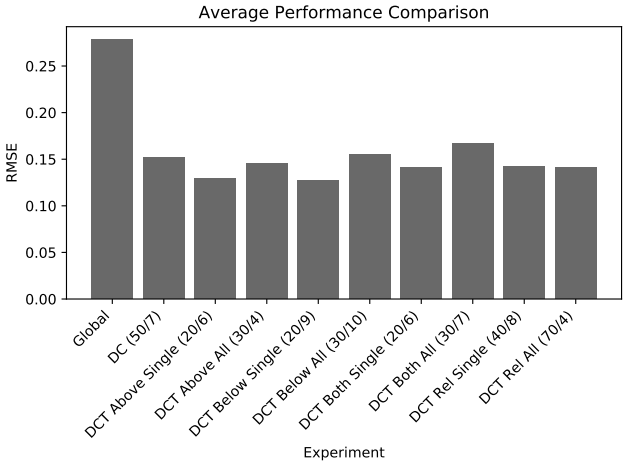


Fig. 7. Summary of performance for the best performing experiments from each major configuration (see Table III for abbreviation meaning)

numbers were 20, 30, 40, 50, 60, and 70. Following this, the LRP clustering used target cluster numbers of 4, 5, 6, 7, 8, 9, and 10. All 36 pairs were examined. Note that no knowledge transfer among networks occurred in the first set of dual clustering experiments, but the knowledge transfer procedure was applied in the second set.

When combining the clusters, there were situations where the candidate clusters for the new combination contained weights that were the same or almost the same. In response to this, the Dual Clustering with Transfer experiments were subdivided further into two node expansion techniques. In the first case, we simply selected one of the nodes at random, treating it as a representative of the set of nodes with similar weight vectors. Thus, in this case, only one node from the group is transferred to the new network. In the second case, we simply transferred all of the nodes from the collection (resulting in the transfer of as many nodes as were in the group). Table III shows the abbreviations that are used in the discussion of the experiment scenarios.

V. RESULTS AND DISCUSSION

Figure 7 summarizes the performance of all experiments in terms of root mean squared error (RMSE) on the wind vector determination task from [18]. As shown, the best performing configurations from each sub experiments all outperformed the Global configuration. The best single configuration was the DCT Both Single (40/4), which had a data instance clustering limit of 40 and an LRP clustering limit of 4.

To test the performance of these scenarios, we ran paired t-tests comparing each scenario depicted in Figure 7 with the Global configuration. The tests were run on the cross validation error from each of the folds. At an $\alpha = 0.1$ level, we found a significant difference between the Global configuration and all of the experimental configurations, with the exception of the DCT Both All configuration. We also found that the DCT Above Single and the DCT Both Single configurations were significantly better than the Global configuration at the $\alpha = 0.05$ level, with the DCT Both Single configuration having greater improvement over Global. However, when comparing both of these configurations to each other, we found no statistically significant difference, even at the $\alpha = 0.1$ level.

A. Global Experiment

Results for the Global experiment are summarized in the heat map in Figure 8. Since this experiment was used as a baseline, we only note the general characteristics of the performance for this network. This figure shows better performance in the North of the dataspace and especially in areas that are over land.

B. Dual Clustering (DC) Experiment

The differences between the RMSE for the DC experiment and the Global experiment are summarized in the heat map in Figure 9. The values used to create the heat map are the mean difference in RMSE over the ten folds for the DC experiment:

$$\text{HM-Val} = \text{Global-RMSE} - \frac{1}{|folds|} \sum_{f \in folds} RMSE_f$$

This means that the heat map shows red when the results favor the experiment and blue when results favor the Global configuration.

TABLE III
EXPERIMENT CONFIGURATIONS FOR DIFFERENT SCENARIOS

Experiment	Sub-Experiment	Expansion	Abbreviation
Global	N/A	N/A	Global
Dual Clustering	N/A	N/A	DC
Dual Clustering with Transfer	Nodes Above Knee Point	Single	DCT Above Single
”	”	All	DCT Above All
”	Nodes Below Knee Point	Single	DCT Below Single
”	”	All	DCT Below All
”	Above and Below	Single	DCT Both Single
”	”	All	DCT Both All
”	Only Relevant	Single	DCT Rel Single
”	”	All	DCT Rel All

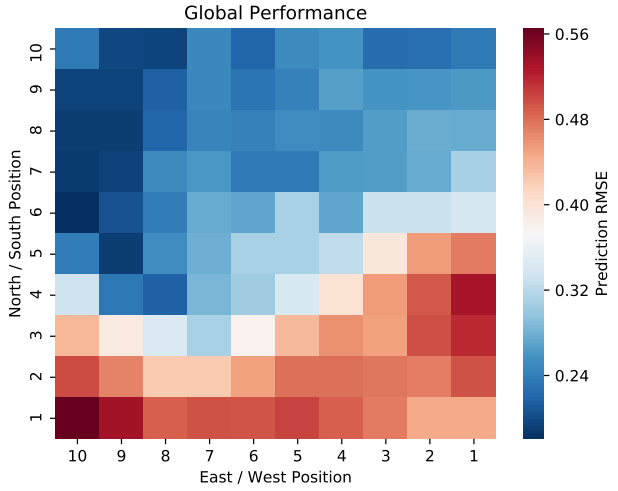


Fig. 8. Global network performance

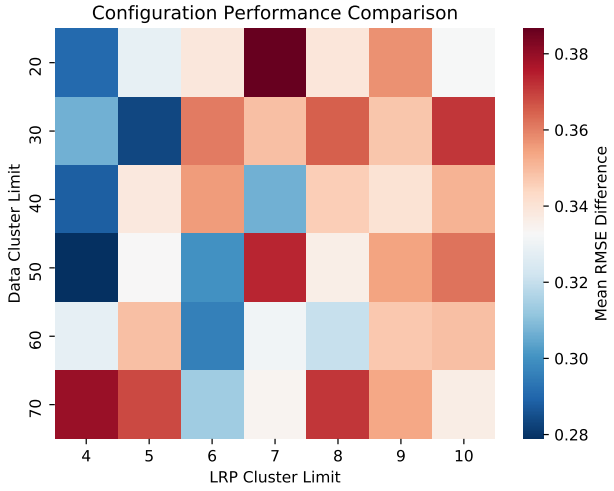


Fig. 9. Mean RMSE difference between DC config and Global config

C. Dual Clustering with Transfer (DCT) Experiment

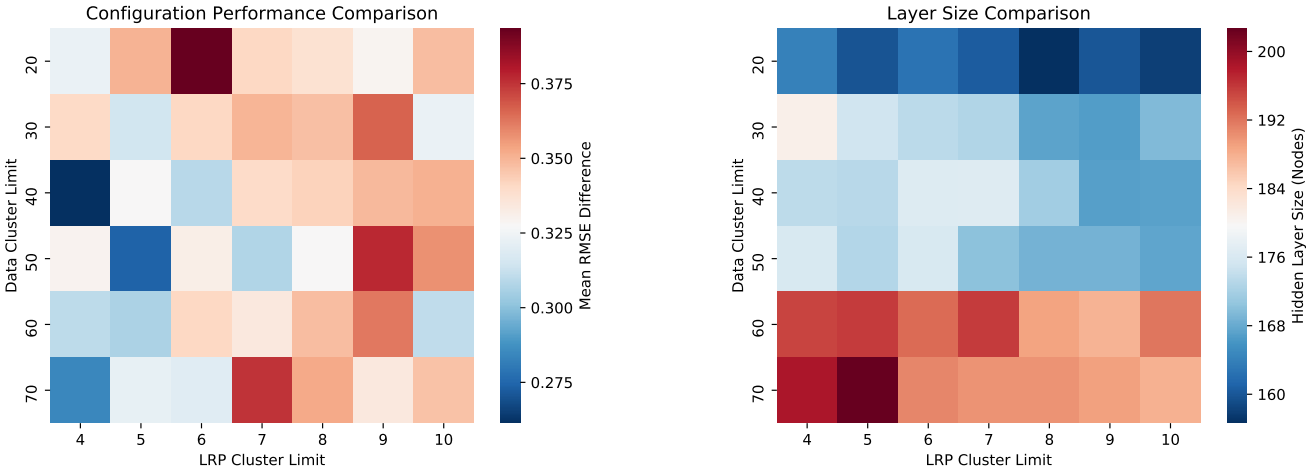
Figure 7 shows the performance of the “DCT Both Single” scenario, which was the best-performing scenario. This heat map is constructed to compare the performance for the 36 different combinations of cluster sizes when configuring data instance clustering and LRP clustering. As shown, the configuration where the cluster limit for data instance clustering was 40 and the cluster limit for the LRP clustering was 4 is the highest-performing version. The heat map in Figure 10a is constructed in the same way as in Figure 9 and compares the winning “DCT Both Single” configuration to the performance of the Global network. The cell corresponding to the best performing configuration in this diagram (Data Cluster Limit: 40, LRP Cluster Limit: 4) conforms to the blue extreme, where the Global average RMSE was 0.3296 and the average RMSE for DCT Both Single was 0.2613.

Figure 10b depicts the the final hidden layer (autoencoder) sized that resulted from the node expansion procedure detailed in Section III-B4. The winning configuration (again, Data Cluster Limit: 40, LRP Cluster Limit: 4) produced on average 173.6780 nodes in the autoencoder, which is near the mean of 176.9263. The lowest average number of nodes produced by “DCT Both Single” was 156.6277 and the highest average number was 202.7015.

VI. CONCLUSIONS AND FUTURE WORK

From our results we conclude that our procedure is a reasonable one to employ if the goal of the neural network is accuracy above training efficiency. Performance was indeed enhanced using our approach, but the cost of using this procedure is certainly a factor, as it is indeed a more lengthy procedure that complicates the training. Perhaps this subdivision is appropriate, especially in situations where one is constrained to neural networks where the expressiveness is constrained. Also, natural divisions in the dataspace may be sought and this is one way of obtaining them.

The number of data clusters and LRP clusters are both parameters that must be tuned with this approach. As of



a) Configuration comparison with Global

b) Hidden layer sizes

Fig. 10. Worst and best performing networks for "DCT Both Single"

now it is not clear how to select these *a priori*, so several configurations must be tried in order to locate an optimal configuration. In the future we would like to see this automated or factored out of the model.

We are also in the process of applying this procedure to new datasets whose functional nature are not predicated on space and time. The new spaces would be predicated on other characteristics that vary continuously at a local level. For the time being, spatiotemporal functional data is a natural fit, especially using selective network application.

Finally, since we are causing the size of the hidden layer to expand and contract through the process of discovering the relevance of certain nodes, we are in effect forcing the expressiveness of the resulting models to adapt to the conditions of the space. An interesting investigation would be to study the behavior of the networks concerning underfitting and overfitting as these expansions and contractions take place. This motivates the idea of searching for "appropriate expressiveness" of a model without generating new models, thus losing all of the intelligence captured in the training of a previous generation of networks.

References

[1] D. Erhan, Y. Bengio, A. Courville, P. Vincent, and S. Bengio, "Why Does Unsupervised Pre-training Help Deep Learning?" *Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[2] T. L. Paine, P. Khorrami, W. Han, and T. S. Huang, "An Analysis of Unsupervised Pre-training in Light of Recent Advances," 12 2014.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[4] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554.

[5] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks," in *Proceedings of the 19th International Conference on Neural Information Processing*, 2006 2006, pp. 153–160.

[6] M. Ranszato, C. Poultney, S. Chopra, and Y. Lecun, "Efficient Learning of Sparse Representations with an Energy-Based Model," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, December 2006, pp. 1137–1144.

[7] B. W. Silverman and J. O. Ramsay, *Functional Data Analysis*. Springer, 2005.

[8] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, p. e0130140, 2015.

[9] W. Samek, T. Wiegand, and K.-R. Muller, "Explainable Artificial Intelligence: Understanding, Visualizing, and Interpreting Deep Learning Models," *ITU Journal: ICT Discoveries*, no. Special Issue 1, 2017.

[10] W. Samek, G. Montavon, A. Binder, S. Lapuschkin, and K.-R. Müller, "Interpreting the Predictions of Complex ML Models by Layer-wise Relevance Propagation," in *Workshop on Interpretable Machine Learning for Complex Systems (NIPS)*, 2016.

[11] A. Binder, S. Bach, G. Montavon, K.-R. Müller, and W. Samek, "Layer-wise relevance propagation for deep neural network architectures," in *Proceedings of the International Conference on Information Science and Applications (ICISA)*, K. J. Kim and N. Joukov, Eds. Singapore: Springer, 2016, pp. 913–922.

[12] A. Rakhmatova, A. Sergeev, A. Buevich, A. Shichkin, and M. Sergeeva, "Partition Procedure of the Initial Data for the Models Based on Artificial Neural Networks," in *Proceedings - 2019 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology, USBEREIT 2019*. Institute of Electrical and Electronics Engineers Inc., apr 2019, pp. 241–243.

[13] A. P. Sergeev, A. G. Buevich, E. M. Baglaeva, and A. V. Shichkin, "Combining spatial autocorrelation with machine learning increases prediction accuracy of soil heavy metals," *Catena*, vol. 174, pp. 425–435, mar 2019.

[14] K. Zhang and A. J. Gasiewski, "Microwave CubeSat fleet simulation for hydrometric tracking in severe weather," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2016, pp. 5569–5572.

[15] R. McAllister and J. Sheppard, "Evaluating Spatial Generalization of Stacked Autoencoders in Wind Vector Determination," in *Florida Artificial Intelligence Research Society Conference (FLAIRS)*, Melbourne, FL, 2018, pp. 68–73.

[16] K. Fukushima, "Cognitron: A self-organizing multilayered neural network," *Biological Cybernetics*, vol. 20, no. 3-4, pp. 121–136, 9 1975.

[17] R. McAllister and J. Sheppard, "Exploring Transferability in Deep Neural Networks with Functional Data Analysis and Spatial Statistics," in *IEEE International Joint Conference on Neural Networks*, vol. 2019-July, 7 2019.

[18] R. A. McAllister and J. W. Sheppard, "Deep Learning for Wind Vector Determination," in *IEEE Symposium Series on Computational Intelligence*, Honolulu, HI, 2017.