Deep Convolutional-Shepard Interpolation Neural Networks for Image Classification Tasks

Kaleb E. Smith[1(✉)], Phillip Williams[2], Tatsanee Chaiya[1], and Max Ble[1]

[1] Florida Institute of Technology, Melbourne, FL 32901, USA
ksmith012007@my.fit.edu
[2] University of Ottawa, Ottawa, Canada

Abstract. With the power of deep learning taking over image classification and computer vision problems, it is no wonder that many algorithms look to their architecture to leverage better results. With Shepard Interpolation Neural Networks (SINN), there is no need for this deep architecture, but rather a shallow and wide approach is taken. SINNs fall short in the ability to take raw input information and extract meaningful features from them. This task is excelled however by deep learning approaches, more specifically, a deep convolutional neural network (CNN) which naturally learns important features from the raw input data for better discrimination. For this paper, we look to collide the power of deep learning features with the speed and efficiency of the shallow learning framework into one cohesive architecture that produces competitive results with a tenth of the computational cost. We start by using different CNNs to extract features from three popular image classification data sets (MNIST, CIFAR-10, and CIFAR-100), and then use those features to efficiently and effectively train a shallow SINN to classify the images accordingly. This method has the ability to not only produce competitive results to the state-of-the-art in image classification, but also blow their computational cost of running an efficient network out of the water by nearly ten times the speed.

1 Introduction

Although deep learning has changed many of the benchmark data sets' "state-of-the-art" accuracy, the overall concept of image classification hasn't changed. That is, to classify well on any data set you need a crucial component - features. However, not just any feature will suffice. For instance, the pixel colors in the RGB color space are indeed features, but with just those three numbers in a feature vector the task ability is limited. This is where deep learning actually shines, in its ability to naturally extract meaningful and useful features from any image and then have those features be unique to certain aspects of the image class it's describing [5,6]. It is here where we look to utilize deep learning for its ability to extract better features for image classification.

A. Campilho et al. (Eds.): ICIAR 2018, LNCS 10882, pp. 185–192, 2018.
https://doi.org/10.1007/978-3-319-93000-8_21

The reason for not using stacked convolutional neural networks for the image classification task is shown in the computational complexity. Though these algorithms excel in classification tasks the computational costs are extremely high and complex [4]. This leads to current state-of-the-art models having an extreme training time due to the billions of tunable parameters. The network is then limited to only being trained on large massive computer servers or clusters in order to get a network that can perform a classification task. Our proposed algorithm looks to advance this computational complexity to a feasible amount that can be achieved on a local CPU or laptop. Shepard Interpolation neural networks (SINN) map the input feature vectors to a set of hypersurfaces which approximates the functions in the feature space to a certain amount of precision. SINN make an impact in image classification tasks in the fact that they can compete in accuracy, but at a huge reduction in the memory footprint and computational cost of the network to be trained (to an order of ten). For images, the input vector is the flattened RGB data of the image, and from there the information is encoded in the relative position of the pixels [7]. This flattening does not help the SINN because it fails to extract meaningful features from these RGB input vectors. This is where CNNs come into play, using their feature extracting ability, meaning features can be utilized in the SINN architecture and achieve high accuracy with efficient speeds [6].

In this paper, the issue of obtaining great performance results (both accuracy and computational) on image classification is addressed. We obtain features from several small CNN architectures and then extract features from three popular image classification data sets: MNIST, CIFAR-10, and CIFAR-100. Then these features are utilized in the SINN framework to classify the images into the appropriate categories. Experiments show that at a tenth of the cost of large bulky CNNs, our approach achieves competitive results in accuracy across all data sets. The rest of the paper first looks at previous related works of Shepard interpolation methods in machine learning and image classification. Second, the CNN features are briefly explained while looking more into how a SINN is formed. Next, experimental results are presented from the three data sets mentioned above. Finally, a brief conclusion looking back on the work done and future outlook for our approach.

2 Related Works

The Shepard Interpolation Neural Network architecture is new, having just being published within the last year. As a result, there are little to no advancements made on this architecture or improvements to the classification results [3]. SINN use a Shepard and metric node inside the hidden layer, which can be thought of as one node essentially since the Shepard node can not activate without a metric node's output. By utilizing the interpolation technique of Shepard [9], this network can design hyper-parameters which require very little tuning, resulting in a decrease of the computational complexity of the network. As mentioned, this network design is still fresh in the community, therefore looking deeply into

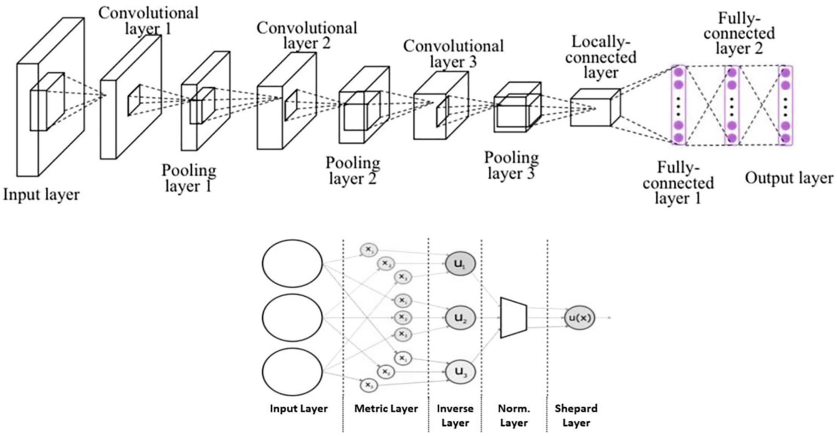

Fig. 1. Illustration of a CNN (top) and SINN (bottom) Architecture.

SINN for literature is difficult; however, this method of Shepard interpolation has been used for tasks outside of image classification.

Park and Sandberg [2] describe a Radial Basis Function Network that functions in much the same manner as the Shepard Interpolation in that they are both based off of exact interpolation and function using a single hidden layer. The Shepard Interpolation Neural Network architecture departs from Radial Basis Function Networks through implementation as well as a few characteristics of the activation functions used. For example, Radial Basis Function Networks require that the activation function be radially symmetric [2], while Shepard Interpolation Neural Networks do not place such a constraint on the metric and Shepard activation functions [3].

Ren et al. [8] outline the application of Shepard Interpolation to convolutional neural networks. The proposed method is the addition of a Shepard Interpolation layer as an augmentation on architectures used for low-level image processing tasks such as inpainting and super resolution. This showed promise in the work, but was never advanced from looking at more literature reviews.

We all know of the vast amount of literature relating to deep learning and convolutional neural networks [1]. To summarize, the concept was brought to light by Lecun in the 1990's for a new way of classifying images [13]. It didn't gain extreme popularity until ImageNet was won by Krizhevsky in 2012 using a deep CNN, blowing all other algorithms away in accuracy [16]. Since then, his paper has been cited over eighteen thousand times and deep learning took off.

3 Deep Feature Based Shepard Interpolation Neural Networks

For any image classification task, the SINN transforms a feature vector to a space of classification vectors. If the Shepard Interpolation Neural Network is

treated as a transform from the feature space to the output classification space, a second transform is needed from the image space to the feature space. Prior to the introduction of the feature-based input layers to the Shepard Interpolation architecture, the mapping from the image space to the feature space is simply flattening the image matrix to a vector; the key innovation proposed is to replace the image flattening step with more sophisticated feature extraction techniques from the deep CNN, resulting in a much better overall accuracy for the model. An overview of a CNN architecture and a SINN architecture can be seen in Fig. 1. Notice, the SINN shows an independent Metric and Shepard layer. These activation functions work in unison and can be visualized as just one layer in a higher level architecture. This leads to the idea of a shallow learning process, which is much quicker, more efficient, and competitive in accuracy versus standard deep methods.

$$\boldsymbol{R}^{i\times j} \rightarrow \boldsymbol{R}^{m} \rightarrow \boldsymbol{R}^{n} \tag{1}$$

where $R^{i\times j}, R^m, R^n$ are the image space, feature space and classification space respectively.

3.1 Network Compression

In the Shepard Interpolation Neural Network architecture [3], the metric nodes contain the vast majority of learnable parameters. Furthermore, it is possible to reduce the total memory footprint of the metric layer by a third after training while maintaining exactly the same behavior. The metric activation function is as follows:

$$\boldsymbol{\Phi}(\boldsymbol{x}) = |\boldsymbol{\alpha}(\boldsymbol{w}\boldsymbol{x} + \boldsymbol{b})| \tag{2}$$

By allowing all three free parameters (α, w and b) to be updated during training, the network converges much more quickly and achieves superior validation accuracy once the training is completed. However, the activation can be written under a slightly different form:

$$\boldsymbol{\Phi}(\boldsymbol{x}) = |\boldsymbol{\alpha}\boldsymbol{w}\boldsymbol{x} + \boldsymbol{\alpha}\boldsymbol{b}| \tag{3}$$

This shows two input digits from MNIST before and after training. You can see the difference in what the Metric node and Shepard node learns depending on their input. In comparison, the features learned from a CNN can also be seen in the same figure.

3.2 K-means Initialization

The performance of the model is directly related to the coverage of the nodes across the areas of interest in the feature space. This concept leads to interesting possibilities for the initiation of the metric nodes. Therefore, to achieve better efficiency in training, the k-means clustering algorithm is used on the feature vectors in the feature space to initialize the metric nodes for the approximation needed to cover the complete feature space surface [10].

4 Experiments

In order to validate that the our deep feature approaches show significant improvements over existing methods, we explore our method on three popular data sets: MNIST data set [13] of 28×28 grayscale images of handwritten numbers and the CIFAR-10 and CIFAR-100 [14] data sets of 32×32 RGB images of various objects.

We pull features from three smaller CNNs to feed into our SINN of only 50 nodes in the hidden layer. These CNN networks are of the following sizes: CNN-1: $32\times(3\times3)$, $32\times(5\times5)$, $32\times(3\times3)$ CNN-2: $32\times(5\times5)$, $64\times(3\times3)$, $32\times(5\times5)$ and CNN-3: $64\times(3\times3)$, $64\times(3\times3)$, $64\times(3\times3)$, these names denote the architecture in the results tables. Each CNN is trained using a ReLU activation function and having a dropout of 0.3 between the convolutional layers; the output is a fully connected layer going to a softmax activation function. These networks were trained over 200 epochs and the middle convolutional layer features were used in the SINN. The middle layer was used from comparison of the three layers, showing a small increase in accuracy while using the middle layer. The training parameters shown in the results below take into account the parameters needed to train the CNN used for pulling features. This is to compare the parameters needed for feature extraction and testing that a regular deep learning approach takes. You can see a reduction in parameters from the original SINN using the raw images to the SINN using deep features; the differences are larger in the CIFAR data because of the third dimension of color given in the original data.

4.1 MNIST

For the MNIST data set, we see an average of 2% increase in performance using the deep CNN features. This is good, considering our later model shows results almost rivaling that of the state-of-the-art Deep Multi Column (DMC) network which has roughly four times the amount of trainable parameters in the network [11]. DMC tries to better mimic the complex human brain and have several smaller deep neural networks that are trained on different preprocessed data. These networks then combined in a multi column approach making the recognition of the DMC boosted compared to the individual networks. This is similar to our approach since we use a smaller deep neural network for features, however, this is done once and several features from multiple trained networks are

Table 1. Classification accuracy on MNIST data set using various variants of feature extraction architectures

Architecture	SINN	SINN-1	SINN-2	SINN-3	DMC [11]
Accuracy	96.50%	97.92%	98.45%	99.71%	99.77%
Parameters	189,065	123,811	216,210	242,180	839,650
Features	None	CNN-1	CNN-2	CNN-3	Deep

not combined into one. Further results for each network can be seen in Table 1. To note, the best result obtained was a testing accuracy of 99.79%, while most models attained a performance of 99.6% to 99.7%. It is also clear looking at the results for MNIST that using a deep CNN to pull features is an improvement over the original SINN with no feature extraction used.

4.2 CIFAR-10 and CIFAR-100

Deep features are then learned and fed into the SINN for training on the CIFAR data sets. The resulting accuracy of CIFAR-10 can be seen in Table 2 while the accuracy of CIFAR-100 can be seen in Table 3. As seen in Tables 2 and 3, the deep features are a drastic improvement from the original image being flattened and trained in the SINN. Even with such a small CNN to extract features, these improvements are almost two times that of the original in both data set cases. This solidifies our approach of the SINN being better at separating the feature space to become a more efficient classifier than a feature extractor. Though it did not reach state-of-the-art accuracy on either data set, it is good to note the size difference in trainable parameters having a decrease of a large order of magnitudes. The state-of-the-art method for CIFAR-10, Aggregated Residual Transformation for Deep Neural Networks (ART), looks to improve the residual network design by being able to have multiple transformation in a block but having the same complexity as the original ResNet [12]. This network contains parameters in the millions which they needed 8 GPUs with only 300 epochs to perform their training. While SINN contains significantly less parameters to achieve decent accuracy, the network can be trained to several thousand epochs with only a single GPU in epoch times less than one second per. Wide Residual Networks (WRN) hold the leading accuracy for the CIFAR-100 data set and once again look to better the implementation architecture of residual networks [15]. This method looks to widen each block of the residual network for better accuracy instead of cascading multitudes of layers on top of one another. Even with this improvement on stacked layers in large deep residual networks, WRN boasts nearly fifty-six million parameters for training. They also are restricted to convolutions of size (3×3) due to the computational performance of large convolutions. These issues do not occur in our SINN architecture, due to its widening of the hidden layer (similar to ART and WRN) we are able to use meaningful features from any convolution size in the CNN to be extracted to the SINN with limited computation loss. It is also interesting to note the size changes of our model depending on the data set. In CIFAR-10, the SINN is a smaller model since there are only ten classes; this means each Shepard node only contains ten metric nodes. This causes the overall width of the network to stay the same, but the internal node difference to be rather easy to tune. Since CIFAR-100 is just ten times the amount of classes per metric node, it fits that the parameters are of an order ten higher than CIFAR-10. Now to mention again, the total number of parameters in the SINN models do include those which were needed to find the features in the CNN. This is the only fair way to assess the feature extraction to classification that these deep models do in

Table 2. Classification accuracy on CIFAR-10 data set using various variants of feature extraction architectures

Architecture	SINN	SINN-1	SINN-2	SINN-3	ART [12]
Accuracy	48.50%	84.84%	89.35%	90.50%	96.44%
Parameters	729,541	254,103	285,230	295,852	68,000,000
Features	None	CNN-1	CNN-2	CNN-3	Deep

comparison to our shallow model. These parameter reductions show a promise in application, having an efficient algorithm which holds high classification accuracy while maintaining low computational cost. It achieves near top performance and translates to training speed having some of our experiments training and testing in under an hour.

Table 3. Classification accuracy on CIFAR-100 data set using various variants of feature extraction architectures

Architecture	SINN	SINN-1	SINN-2	SINN-3	WRN[15]
Accuracy	36.50%	68.52%	70.25%	72.86%	81.7%
Parameters	7,436,967	2,652,052	2,978,502	3,143,025	56,000,000
Features	None	CNN-1	CNN-2	CNN-3	Deep

5 Conclusion

In this paper, we look at utilizing and harnessing the power of deep learning for feature extraction using a CNN. These features were then used in our SINN architecture to show high accuracy on the data sets tested. Though the SINN is still rather new, the results show promising accuracy in image classification against the state-of-the-art. What shows even more promise of SINN is that of lowering the computational cost for this caliber of accuracy, by thirty - three hundred times less than that of the leading deep learning frameworks. This shows it is not always necessary to scale a deep framework to immense extents when smaller frameworks with a SINN work in comparison. Possible future work could be to expand the architecture of the SINN to see if accuracy performance can increase (while of course decreasing computational performance) with more nodes. There could be some use to see SINN in other machine learning tasks, such as natural language processing, time series analysis and forecasting, or medical image analysis. Since the approach is still new, there are many open areas for research interest. Also to note, a stacked method could be possible, however, this would neglect our rebuttal that deep isn't always the best way, and our shallow approach is fast and efficient.

References

1. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) ICANN 2011. LNCS, vol. 6791, pp. 52–59. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21735-7_7
2. Park, J., Sandberg, I.W.: Universal approximation using radial-basis-function networks. Neural Comput. **3**(2), 246–257 (1991)
3. Williams, P.: SINN: Shepard Interpolation Neural Networks. In: Bebis, G., et al. (eds.) ISVC 2016. LNCS, vol. 10073, pp. 349–358. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50832-0_34
4. Le, Q.V.: Building high-level features using large scale unsupervised learning. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8595–8598. IEEE, May 2013
5. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (ELUS). arXiv preprint arXiv:1511.07289 (2015)
6. Wan, L., Zeiler, M., Zhang, S., Cun, Y.L., Fergus, R.: Regularization of neural networks using dropconnect. In: Proceedings of the 30th International Conference on Machine Learning (ICML 2013), pp. 1058–1066 (2013)
7. Petrov, Y., Zhaoping, L.: Local correlations, information redundancy, and sufficient pixel depth in natural images. JOSA A **20**(1), 56–66 (2003)
8. Ren, J.S., Xu, L., Yan, Q., Sun, W.: Shepard convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 901–909 (2015)
9. Donald, S.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM National Conference. ACM (1968)
10. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035. Society for Industrial and Applied Mathematics, January 2007
11. Ciregan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3642–3649. IEEE, June 2012
12. Xie, S., et al.: Aggregated residual transformations for deep neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2017)
13. LeCun, Y., Bengio, Y.: Convolutional networks for images, speech, and time series. In: The Handbook of Brain Theory and Neural Networks, vol. 3361, no. 10 (1995)
14. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report, University of Toronto (2009)
15. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (2012)