OrderNet: Sorting High Dimensional Low Sample Data with Few-Shot Learning

Samuel Hess and Gregory Ditzler
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ 85721
shess@email.arizona.edu, ditzler@arizona.edu

2021 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-3900-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/IJCNN52387.2021.9533766

***Abstract*—Neural networks have shown remarkable classification performances in recent years, often outperforming humans in many tasks. Unfortunately, there are some tasks where conventional neural networks and their training methods have under-performed. One of these areas is learning from a small number of samples which have been partially addressed with the development of few-shot neural networks. Few-shot neural networks contrast traditional classification networks by learning classification tasks with datasets with only a few samples per class; however, these few-shot techniques are trained collectively with a large number of labeled samples. Hence, few-shot learning approaches only address the low sample per class learning problem whereas the task of learning strictly from a low sample size still goes mostly unresolved. In this contribution, we address the challenge of learning from high dimensional low sample data by revising the problem into a data ordering task. Specifically, we have designed OrderNet, a novel network design and training approach that can take a relatively small amount (less than 200 samples) of ordered high dimensional low sample data and organize many more unseen samples. To the best of our knowledge, OrderNet is the first neural network to address the high dimensional low sample data using techniques adopted from few-shot learning. We evaluate OrderNet against its ability to order images of analog clocks by time as well as images of profile pictures by age. Additionally, we demonstrate that OrderNet has superior performance over a conventional regression neural network in the low sample regime.**

***Index Terms*—Few-Shot Learning, Low Sample Learning, Active Learning, Ranking.**

I. Introduction

Deep learning has been successfully applied to many applications with the most typically being classification; however, as the success and popularity of deep learning expands, clever techniques are continuously being developed to take advantage of the technology in unconventional ways [1]–[3]. Naturally, deep learning methods use empirical data to train and develop models, large amounts of data are often needed, making low sample size training particularly difficult to apply these methods. Research in neural networks has recently reformulated the low sample size problem into a few-shot framework where only a small number of samples are available for each class in the supervised learning problem. The goal of a few-shot problem is to be able to classify an unknown sample (i.e., from a class never seen during training) from only one (or a few) exemplary samples from that class. This task of few-shot differs from the conventional classification where the goal is to classify an unknown sample from a class that *was* seen during training.

To circumvent the challenge of a small number of training samples per class, Vinyals et al. [4] created matching networks that use a novel episodic training approach. During episodic training, a small set of samples are repeatedly and randomly selected from the training set to establish support and query batches for comparison during training iterations that more closely match the testing task's goal (i.e., match query samples to support samples). The result is a network training scheme that cleverly uses repeated sub-sampling to learn similarities/differences between samples rather than using a conventional network approach of training on a much more extensive database to learn classification without a support set at testing time. Although significant improvements have been made on the design since matching networks, many approaches still use episodic training and distance or contrastive based design [5].

Unfortunately, the low sample and low sample per class tasks are two separate yet challenging problems and many of the commonly used few-shot approaches do not truly resolve the low sample training problem; few-shot approaches are designed to only address the low sample per class training problem. As a testament to this statement, the benchmark datasets for few-shot performance are *mini*ImageNet and Omniglot, which are commonly trained with 48,000 samples and 24,000 samples, respectively. Only *after* training with lots of samples are the few-shot approaches able to predict from few samples, but this is formally done at testing time. Thus, a natural question evolves from the few-shot literature: can a similar neural network be *trained* with a low number of samples (i.e., on the order of hundreds instead of thousands)?

Our design of OrderNet demonstrates that in some cases, a neural network can be trained with a much smaller number of samples given some novel architectural changes and one core data requirement: *there is a natural order in the training data.* More specifically, if the training samples are naturally ordered then we can use that information by training a modified few-shot network to learn the pairwise order rather than direct classification. This novel approach expands the dataset from N individual samples in the conventional neural network to a much larger set of $N(N-1)/2$ pairwise samples. We have recognized that if the training data can be organized into such an ordered set, then the few-shot approach with episodic

978-0-7381-3366-9/21/$31.00 ©2021 IEEE

TABLE I
COMPARISON OF ORDERNET AND GENERAL FEW-SHOT NETWORK ALGORITHMS

	OrderNet	**Generic Few-Shot Networks**
Training scheme	Episodic	Episodic
Network backbone	Varies	Varies
Loss function	Distance/contrastive	Varies but often distance/contrastive
Dataset assumptions	Small set of linearly ordered samples	Large set of labeled samples and very low samples per class
Application/use	Data Sorting/Linear Regression/Active Learning	Novel sample classification given a few known samples

training and a modified distance function allows for a large number of pairwise comparisons that can be used to train a deep neural network effectively.

OrderNet borrows concepts from the few-shot research community, which makes few-shot the natural choice to establish a research point of reference; however, it should be noted that OrderNet and few-shot have very different uses and applications. Few-shot uses a large set of labeled samples to perform novel sample classification given a few known samples. In contrast, the use of OrderNet is applicable in low sample data sorting and active learning (Table I highlights the similarities and differences). For example, consider the scenario of organizing a large database of peoples' profile photographs by a person's age *without* direct knowledge of any single sample's actual age. A human observer could approximate the age of many photos to establish a training set of labeled images. We argue that it is often easier to make pairwise comparisons, organize the data sequentially by age, and then assign numerical values to the organized data. Further, we show in this work that by using the latter method for establishing ground truth and then applying our novel OrderNet approach, significantly less data is needed to achieve considerable gains in performance (w.r.t. a conventional neural network). Labeling data can be a significant cost to the machine learning process, and the creation of OrderNet is motivated by the need to reduce this cost with better performance on much smaller labeled sample sizes.

To demonstrate our concept of leveraging order for low sample deep learning we provide the following contributions in this work:

- Propose a modified few-shot loss function to evaluate the signed distance between pairs of training data samples.
- Demonstrate how existing episodic training can be used in conjunction with the modified loss function to train a deep neural network that naturally orders data from a relatively small number of training samples.
- Adapt two datasets (i.e., time in analog clocks and age in profile pictures) for the evaluation of our OrderNet model
- Baseline OrderNet against a standard regression network for ordering datasets as a function of sample size.

II. RELATED WORK

The majority of neural networks are parametric models where the parameters are learned empirically from training data, so it is no surprise that neural networks take many samples to train a classification task. Few-shot neural networks have evolved towards classifying data with relatively few exemplary samples per class to mitigate the need for many class samples [3], [6]. Although few-shot has an earlier history in pattern recognition [7], [8], the majority of the modern few-shot research took shape after the development of the benchmark Ominiglot dataset [9]. As networks continued to solve few-shot tasks more efficiently, the performance on Ominiglot became nearly perfect then the more complex *mini*ImageNet dataset was introduced by Ravi and Larochelle [10]. The *mini*ImageNet dataset is now the de-facto benchmark for few-shot networks. The specific few-shot challenge proposed with these datasets is to train a model with a subset of data and then present the model with one (or a "few") labeled samples from a disjoint set of data. The objective is to use the information from the few labeled samples on-the-fly to classify additional unlabeled samples. Few-shot Siamese networks [11], which modified a signature verification approach [12] was one of the first methods to achieve significant performance on the Ominiglot dataset in a one-shot task. These Siamese networks still obtain comparable performance to many methods developed since their initial introduction.

Another significant contribution to the recent few-shot research was made by Vinyals et al. with Matching Networks [4]. The authors matched the training scheme to the testing scheme by repeatedly (and randomly) selecting subsets of the training data to mimic the few-shot test: a process now known in the few-shot literature as episodic training. For example, if the few-shot testing challenge was: given one labeled sample from five classes then classify another set of unknown and unlabeled samples from each of those classes. Then training would contain episodes of one randomly selected labeled sample from five randomly selected classes and learn to match unknown query samples to the right labeled sample. In the case of matching networks, a cosine distance was used and later Snell et al. demonstrated that the Euclidean distance performs empirically better in their similar Prototypical Network design [13]. Since the development of Prototypical Networks many additional approaches have been proposed that include metric based approaches [14], [15], generative based approaches [16], [17], meta learning [18], [19], or some combination of the aforementioned approaches. Surprisingly, Chen et al. has shown that many of the methods appear to have very comparable results when using the same neural network backbone [5].

Unfortunately, many of the few-shot algorithms are restricted to working only within the few-shot paradigm. That is, training is still done on thousands of samples with very few samples per class (e.g. *mini*ImageNet 48,000 training samples) and testing is done with labeled exemplary samples. In comparison to the few-shot literature, OrderNet has inherited a

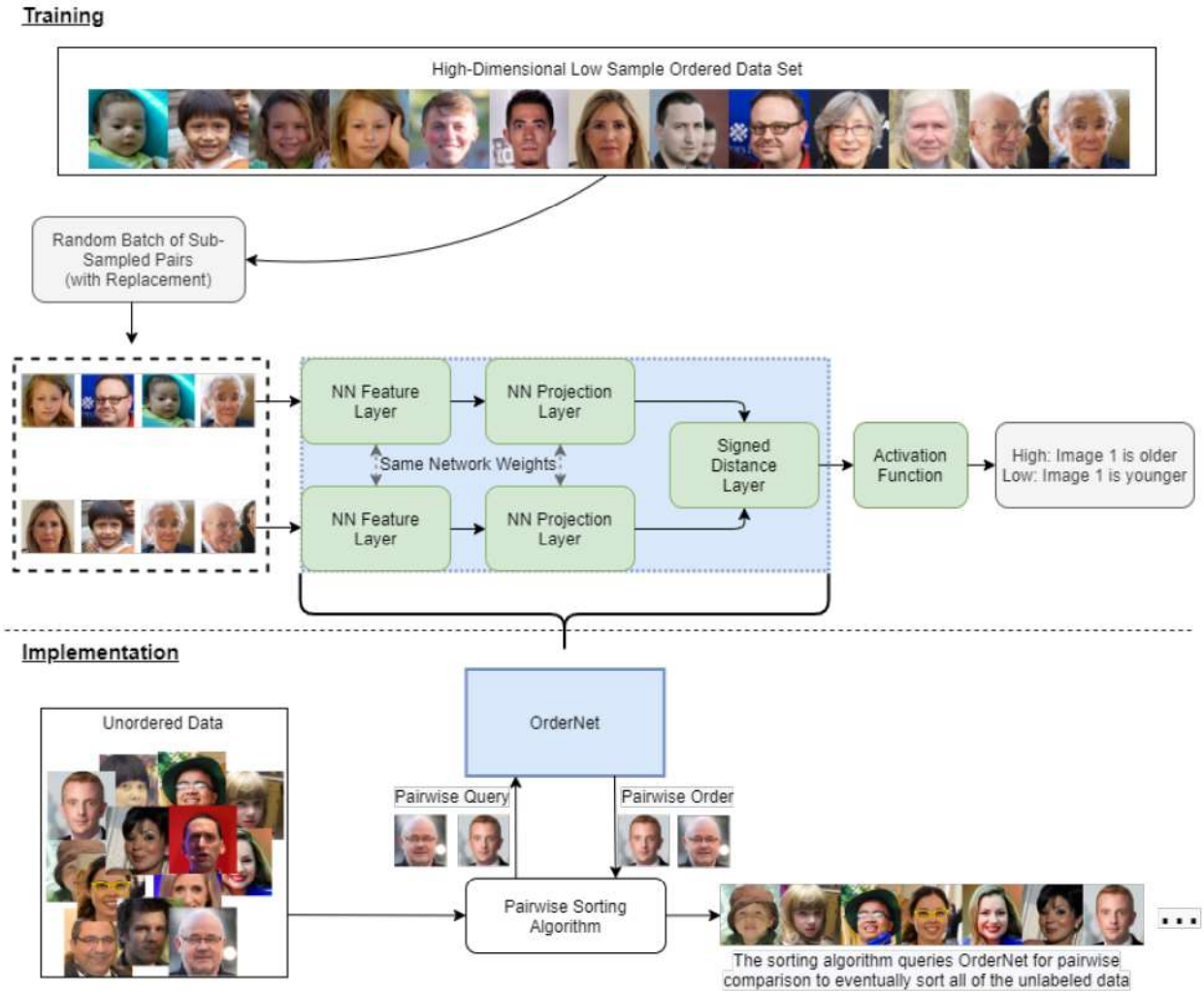

Fig. 1. Overview Block Diagram of the OrderNet Process and Design for Implementation and Training

very similar architecture to the original Siamese Network with a modified distance function. Additionally, OrderNet uses an episodic training, which is similar to approaches like Matching Networks and Prototypical Networks. However, OrderNet and other few-shot approaches' core difference is that OrderNet is designed to order/sort a completely unseen and unlabeled dataset given only a few hundred samples in total. This is different from trying to explicitly classify unknown samples by matching them to a set of given labeled samples.

Concerning ordering data, there is already a prominent area of research focused around ranking systems that operate as search engines [20]. The general objective of these information retrieval systems is to take an unknown query sample (e.g., picture of a bike) and provide a ranked list of available samples in a database that most closely match the query sample (e.g., pictures of other bikes). Unfortunately, many of these approaches set up their training with labeled class data and (unlike OrderNet) cannot effectively train on just an organized ordered list of data. One notable exception is RankIQA [21], where a ranked set of images are synthetically produced by blurring data at increasing levels and the organized datasets are used to train a network. This network is trained to rank a set of unseen images by quality; however, large amounts of training data are still needed and it is unclear how rankIQA would be extended to other rank ordered training sets. In contrast, OrderNet only requires a few hundred ordered samples for training and can be easily applied to any ordered data.

Finally, it is worth mentioning that the primary motivation for the development of OrderNet was to aid in the tedious and costly process of labeling data with a human in the loop. If a training dataset can be effectively ordered, OrderNet can exploit the larger amount of pairwise comparisons instead of merely using individual samples. As a result, using OrderNet with a small labeled dataset can yield much better organizational performance of unlabeled data than a comparative approach that uses regression or classification. Better organized data reduces the evaluation and correction time of a human in the loop and the interaction between the network model and human can be iterative to improve classification performance over time. Such approaches meant to address

tedious labeling are commonly known in the research literature as active learning methods. Although there are many variants to active learning, most approaches start with a minimally trained network and a large amount of unlabeled data [22]. The unlabeled data are classified with accompanying confidence metrics. If the confidence is high for a particular sample, then it can be labeled and used on the subsequent iterations of training. Active learning approaches are often processes surrounding a particular neural network model and therefore complement the proposed OrderNet model.

Further, we argue that it is easier for a human to organize data versus classify in many cases, so OrderNet might prove to be more intuitive in an active learning process. For example, consider the scenario mentioned in the introduction where a human observer has to label unlabeled photos of people by age. It is arguably easier (and more accurate) for a human to organize them from youngest to oldest with pairwise comparison rather than explicitly guessing each person's actual age. In this regard, we see active learning as a potential avenue for the use of our OrderNet model but consider it out of scope for the work presented in this paper.

III. Proposed Method

The design of OrderNet is centered around pairwise comparisons such that a small set of N samples are transformed into $N(N-1)/2$ pairwise combinations for training. The episodic pairwise sub-sampling is common to the few-shot and contrastive literature where the approach is used to identify if an unknown query sample under test is an in-class or an out-of-class sample with respect to a few known samples. Here, OrderNet has modified the loss function to indicate if a query under test is larger than or smaller than another sample. The subtle – albeit important – modification to the loss function in conjunction with the pairwise training scheme allows the network to learn pairwise order from a high dimensional low sample dataset. This tool can be used with any pairwise sorting algorithm to order a large set of data and the entire process is shown in Figure 1.

A. OrderNet Model

More formally, consider an ordered training set $Tr = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ where each $\mathbf{x}_i$ is a D-dimensional vector ($\mathbf{x}_i \in \mathbb{R}^D$) and each y_i is a 1-dimensional value ($y_i \in \mathbb{R}^1$) such that $y_i < y_{i+1}$. y_i can either be a floating-point value that represents some known quantity of $\mathbf{x}_i$ ordering or – if an explicit value of the ordered label is unknown – it can be assigned an arbitrary value such that $y_i < y_{i+1}$ holds (e.g., $y_i = i$). On each episode, a batch of B pairs of high dimensional samples $\mathbf{x}_i$ and $\mathbf{x}_j$ (e.g. profile images of people) are randomly sampled from the training set Tr with replacement such that $y_i \neq y_j$. Each pair of training samples pass through the same neural network (with learnable parameters ϕ) such that it transforms a D-dimensional sample into a K-dimensional projection (i.e. $\mathbf{f}_\phi : \mathbb{R}^D \rightarrow \mathbb{R}^K$). Note that Figure 1 distinguishes between a "feature layer" and a "projection layer."

The feature layer is meant to exploit the datasets' structural features such as a Convolutional Neural Network (CNN) for images or Long Short Term Memory Networks (LSTM) for sequential. In contrast, the projection layer is a much smaller dense neural network that projects the feature layer's output into a consistent multi-dimensional space (e.g., a vector of 512 elements). Although the feature layer and projection layer are part of the same network architecture, a distinction is made between them because the weights of the projection layer are *always* learned through training. In contrast, the feature layer weights have the option of being pretrained (i.e. Xception Network with pretrained imageNet weights). Our experiments use randomized weights and learn both the feature and projection layers from scratch to isolate OrderNet's training performance on low sample size.

After both samples pass through the projection layer, the signed distance between the feature projections of the sample pairs are computed and a single sigmoid activation function ($\sigma(\cdot)$) maps the signed distance to a distribution given by

$$\widehat{p}_\phi(y_i > y_j | \mathbf{x}_i, \mathbf{x}_j) = \sigma(\mathbf{f}_\phi(\mathbf{x}_i) - \mathbf{f}_\phi(\mathbf{x}_j)) \tag{1}$$

The binary cross-entropy loss function for the network can then be represented as

$$J(\phi) = -\sum_{i}^{B} \sum_{j:j\neq i}^{B} \mathbb{1}_{y_i > y_j} \log[\widehat{p}_\phi(y_i > y_j | \mathbf{x}_i, \mathbf{x}_j)] + \mathbb{1}_{y_i \leq y_j} \log[1 - \widehat{p}_\phi(y_i > y_j | \mathbf{x}_i, \mathbf{x}_j)] \tag{2}$$

where the objective of the loss function is to drive the projected difference $\mathbf{f}_\phi(\mathbf{x}_i) - \mathbf{f}_\phi(\mathbf{x}_j)$ to larger positive values when $y_i > y_j$ and larger negative values when $y_i \leq y_j$ over the total batch of B pairwise samples.

Throughout the training process, it is important to assess the validation performance periodically. Since the available amount of training data is small, having a set of disjoint validation data prevents overfitting and allows for an exit criterion. The network is included as the comparator in a sorting algorithm as illustrated in the implementation portion of Figure 1 (bottom). Any sorting algorithm can be used for validation and testing as long as it uses pairwise comparisons. We used a simple comparison sort in our design. The output order of the sorting algorithm is compared against the true order of the data via the Kendall Tau Coefficient as given by [23]

$$\tau = \frac{2}{N(N-1)} \sum_{i<j} \text{sign}(y_i - y_j)\,\text{sign}(z_i - z_j) \tag{3}$$

where the values y_i represent the true sorted samples and z_i represent the sorted samples from the output of the OrderNet enhanced comparison sort. In the ranking literature, the Spearman Rho Coefficient is often used over the Kendall Tau Coefficient because it incorporates the distance between sorted samples, not just the absolute order. However, we have chosen to use the Kendall Tau metric in our work because OrderNet can be applied to unlabeled ordered data by applying

an arbitrary order label (as presented earlier in this section). In scenarios where the labels are arbitrarily chosen, only the order is meaningful and Kendall Tau is more representative of the actual sorting ability.

The full training and validation pseudo-code is shown in Algorithm 1. Since there are a variety of pairwise sorting algorithms that would work with OrderNet, we dictate it in the pseudo-code as function, $g(\cdot)$, that takes the OrderNet pairwise comparator, $\phi(\cdot)$, and the list of input validation samples ($\{\mathbf{x}_{v1}...\mathbf{x}_{vM}\}$).

IV. Experiments

OrderNet was evaluated against two image datasets with complex features: 1) synthetically generated images of analog clocks and 2) real profile images of people. The objective for the analog clocks dataset was to order images by the true time and in the objective for the profile images of people was to order the images by the true age of the person. All experiments were designed to evaluate when training is limited to a relatively small number of samples. Since both of these machine learning datasets had thousands of training samples, we significantly reduced the sample size for training in all experiments and compared performance as a function of the number of training samples.

A. Synthetically Generated Analog Clocks

The synthetically generated analog clock data was developed by Kaggle user Shiva Verma [24]. It consists of a basic set of RGB generated images of clocks with an hour and minute hand. Each image is square with 300 pixels in each dimension and three channels for pixel color. Examples of the data are shown in Figure 2. The analog clocks are all circularly shaped and oriented with 12 o'clock at the top of the image; however, the size, color, and displacement of the clocks are randomly varied as well as the hour and minute hand positions. The images are labeled by the time present on the hour and minute hands with 12:00 represented as 0.0. Since the dataset contains 50,000 synthetically generated samples, we can easily control the number of labeled samples that we use for learning.

For training OrderNet, we randomly sampled between 25 and 800 samples of the 50,000 analog images (where the number of samples varies depending on the experiment). The training set was then ordered by the true time label. For example, since the label for 12:00 in the data was 0.0, the first ordered sample was the sample closest to 12:00 moving in a clockwise rotation. On each training iteration, a batch of 25 pairs of random samples (with different labels) are chosen from the training set with replacement. Every pair of samples from the batch were passed through the neural network feature and projection layer. The feature layer was the Keras implementation of Xception network initialized with randomized weights and the network projection layer is a dense layer with 512 ReLU activation nodes. The entire network has 21.9M trainable parameters. The signed distance is computed between each sample's projection layer's output and passed through a final activation function. The final result

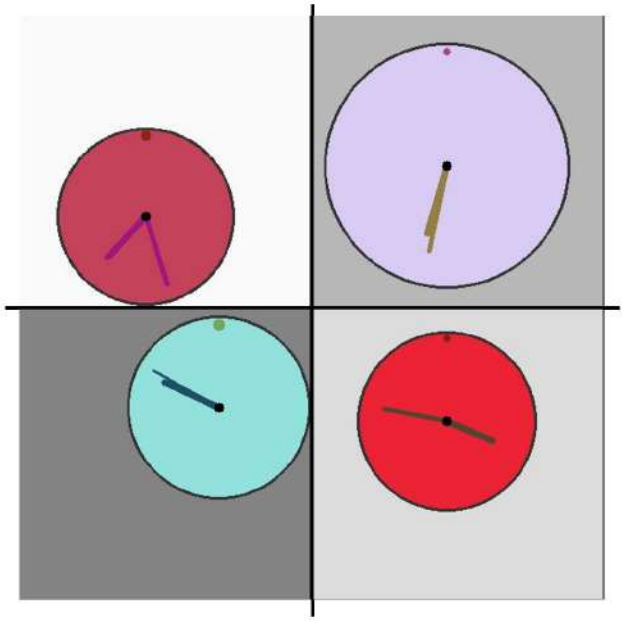

Fig. 2. Example of four data samples from the synthetic analog clock dataset.

is a single output value indicating a binary decision on the two provided samples' order. In our implementation, a sigmoid activation was used and, combined with the loss function in Eq. (2), is designed to produce an output of 1 if the i-th sample is later in time than the j-th sample, else the output value is 0.

For each experiment, training was run for 30,000 episodes with an Adam optimizer and a learning rate of 6×10^{-5}. Every 50 episodes a disjoint set of 100 samples was used to validate ordering performance and the network with the best validation performance was stored for final testing evaluation. A comparison sort was used with the respective OrderNet model as the pairwise comparator to evaluate ordering performance. The sorted result was evaluated against the true sorted indices via the Kendall Tau coefficient (see Eq. (3)).

For comparison against OrderNet, a regression network was trained and evaluated on the same data. The regression network uses the same architecture and training process except for the signed distance layer. Instead of the final activation function indicating the order of two samples, it simply learns to predict the sample's true label. The regression network uses mean squared error between the predicted output and the normalized labeled for the training loss function, but the remainder of the training procedure is congruent with OrderNet training. Additionally, the regression network learns an exact value rather than operating as a comparator so there is no need to integrate the regression model into a comparison sort algorithm. During validation and testing, all samples' regressed output can be calculated and then sorted using any common sorting algorithm available. The sort's final order is then compared against the true sorted indicies via the Kendall Tau coefficient exactly as it was done in the OrderNet design. Again, the network with the best validation performance was

Algorithm 1 OrderNet training and validation pseudo-code

Input: Training Set $Tr = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, disjoint validation set $V = \{(\mathbf{x}_{v1}, y_{v1}), ..., (\mathbf{x}_{vM}, y_{vM})\}$, batch size B, total training episodes E, initial model ϕ, model learning rate η, pairwise sorting algorithm g, and initialize best Kendall coefficient $\tau_{best} = -1.0$
Output: model with best validation performance ϕ_{best}, network loss $J(\phi)$, and ordered of validation data $\{z_{vi}, ..., z_{vM}\}$

for $i = 1, \ldots, E$ **do** // each episode
  $J(\phi) = 0$
  for $b = 1, \ldots, B$ **do** // each sample in the batch
    Create a set of pairwise samples, $Te = \{(\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j)\}$,
    randomly chosen from Tr with replacement and such that $y_i \neq y_j$
    $\widehat{p}_\phi(y_i > y_j | \mathbf{x}_i, \mathbf{x}_j) = \sigma(\mathbf{f}_\phi(\mathbf{x}_i) - \mathbf{f}_\phi(\mathbf{x}_j))$ // Compute the distribution value for each pair of samples
    $J(\phi) = J(\phi) - \mathbb{1}_{y_i > y_j} \log[\widehat{p}_\phi(y_i > y_j | \mathbf{x}_i, \mathbf{x}_j)] - \mathbb{1}_{y_i \leq y_j} \log[1 - \widehat{p}_\phi(y_i > y_j | \mathbf{x}_i, \mathbf{x}_j)]$ // Add the loss function over the entire batch
  end for
  $\phi \leftarrow \phi - \eta \nabla_\phi J(\phi)$ // Update model
  if evaluate model on this episode **then**
    $g(\phi, \{\mathbf{x}_{v1}, ...\mathbf{x}_{vM}\}) = \{(\mathbf{x}_{v1}, z_{v1}), ..., (\mathbf{x}_{vM}, z_{vM})\}$ // pairwise sorting algorithm takes in validation data and provides ordered index labels z_{vi}
    $\tau = \frac{2}{M(M-1)} \sum_{i<j} sign(y_{vi} - y_{vj}) sign(z_{vi} - z_{vj})$ // compute Kendall τ between true order of val. data and output of sort
    if $\tau > \tau_{best}$ **then**
      $\phi_{best} = \phi$
      $\tau_{best} = \tau$ // store best τ
    end if
  end if
end for

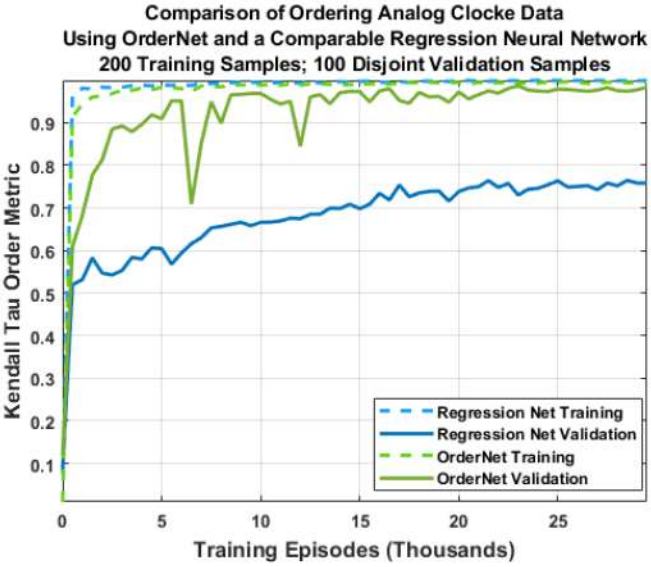

Fig. 3. Comparison of Regression Network and OrderNet Training and Validation performance as a function of training time.

stored for final testing evaluation.

Figure 3 shows the training and validation results when 200 training samples were used. As observed in the plot, the regression network's training performance is nearly perfect early in the training and the corresponding validation performance plateaus. This effect is expected since the number of training samples is small (200) and the regression network quickly overtrains to the data. In contrast, OrderNet's novel pairwise comparison architecture allows the 200 individual data samples to be 19,900 pairwise samples. As a result, the training takes more epochs to converge, and the network's performance can continue to improve (as seen in the validation data).

To test the performance as a function of training samples, we repeated this experiment for 25, 50, 100, 200, 300, 400, 500, and 800 training samples. In each experiment, a disjoint set of validation data was used to monitor performance. The validation data is labeled data available during training that is restricted from the training set. It is used to track performance and/or provide an independent exit criterion. 100 disjoint samples were used for the validation data set and the model with the best (i.e., largest) Kendall ordering metric was used as a final model for testing. Then 200 random samples (disjoint from both the training and validation data) were used for testing the final model. Figure 4 shows the results for OrderNet and the conventional regression network as a function of the number of samples. As expected, both networks' performance increases as a function of the number of training samples and eventually converge; however, OrderNet has considerable performance gain when the training sample size is small. It is important to note that training performance for both networks are almost nearly identical and high due to the small number of training samples; however, upon closer inspection of the raw data, OrderNet performs slightly worse than the regression network only on training data. As mentioned previously about Figure 3, this behavior is expected in the experiment because the regression network has effectively N samples whereas OrderNet expands those N samples into $N(N-1)/2$ pairwise samples. Therefore, reducing over training and improving validation and testing performance but also causing an expected degradation in training performance.

B. Age of Individuals in Profile Images

The UTKFace dataset contains over 20,000 RGB images of aligned and cropped faces labeled by each person's age, gender, and race [25]. Each cropped image is square with 200 pixels in each dimension and three channels for pixel color. Examples of the data are shown in Figure 5. The corresponding labeled age of each person is an integer value from 1 to 116. However, it is highly unbalanced, especially in the range between 91 to 116 years of age where the number of samples drops off considerably. As a result, we

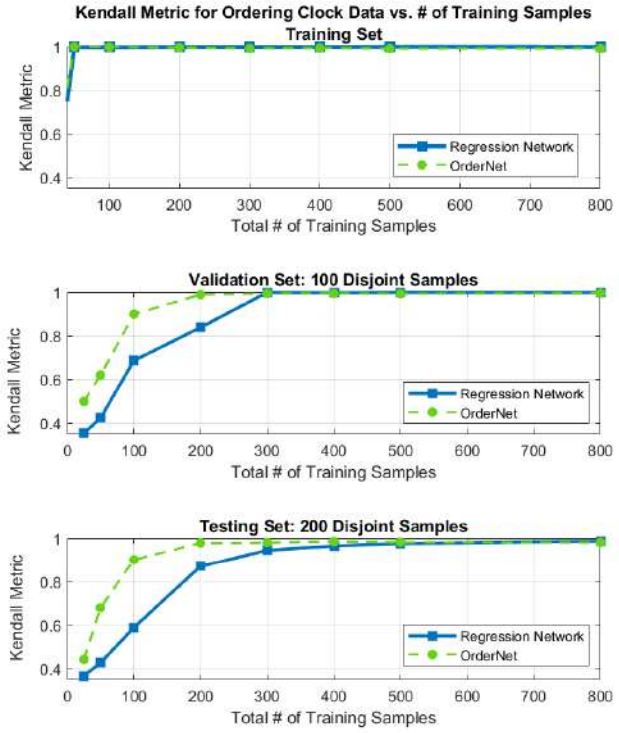

Fig. 4. Final training, validation, and testing performance for ordering analog clocks by time as a function of number of training samples

Fig. 5. Example of four samples from UTKFace age dataset

restricted our training, validation, and testing on ages labeled 1-90 which still contains 23,620 samples. Similar to the analog clocks dataset, we constrained the training and validation sample sizes. The training data is one (randomly selected) sample for each class when experiments have less than 90 training samples to mitigate potentially confusing results from imbalanced training. For experiments greater than 90 training samples, the same number of randomly selected samples from each of the 90 classes are used (i.e., 90, 180, 270, etc.). Note that OrderNet uses batch pairwise sub-sampling with replacement so the balanced dataset is not a prerequisite for OrderNet to learn effectively; however, balancing the dataset establishes a baseline of performance that is easier to compare against.

For training OrderNet with the UTKFace dataset, we used an identical network architecture as presented in the analog clocks subsection (i.e. Keras Xception network feature layer with 512 node projection layer). When training with less than 90 samples, random classes (i.e., ages) were chosen without replacement. When training with greater than 90 samples, a balanced set of samples from each class was used. For validation, a consistent but disjoint set of 2 samples from each age were used (i.e., 180 images in total) and the testing set consisted of 900 samples (10 samples per class) disjoint from both the training and validation sets. Figure 6 shows the results for OrderNet and the conventional regression network as a function of the number of samples for the UTKFace dataset. As in the analog clocks, both networks' performance increase as a function of the number of training samples and also converge as the number of samples increase. Although OrderNet has a small testing performance gain in the lower training sample regime, the performance difference is not nearly as obvious as in the case with analog clocks. We speculate this smaller performance gain could be due to the coarse classification of the UTKFace dataset. That is, the UTKFace dataset we used has 90 class labels (i.e., one label per age from 1-90 years of age) whereas the analog clocks have 720 class labels (i.e. one label per hr/min on a clock). OrderNet explicitly benefits from the pairwise subsampling, so 180 samples with 180 labels effectively allows for more combinations than 180 samples with 90 labels. As a result, we expect some degradation as the dataset becomes more discrete. A potential alternative to improve the performance OrderNet on the UTKFace dataset would be to order all the training data samples from youngest to oldest rather than by the discrete year of age classification used in the UTKFace datasets.

V. Conclusions

The conventional paradigm of few-shot learning is not truly low sample training. Rather, few-shot models assume that the learning setting is high sample training with a low number of samples per class. Training neural networks on high dimensional data with a low number of total samples is a significantly more challenging task due to the empirical training nature of neural networks. Our OrderNet architecture and training scheme allows for truly low sample training if one condition is met. Namely, suppose the training data can be organized by a human operator into a naturally ordered list based on some feature or true data label. In that case, OrderNet can exploit the pairwise comparisons to order a significantly larger dataset. The performance was demonstrated on two sets of data that already contained labeled values: analog clocks and profile images. In both cases, the ordering performance (based on the Kendall Tau Coefficient) of OrderNet improved performance over a conventional regression network in the low sample regime. As the number of samples increased, the performance of the two networks converged as expected. It appears from empirical results that the convergence in performance is around the number of classification labels in the dataset. Although it is uncertain from only two datasets,

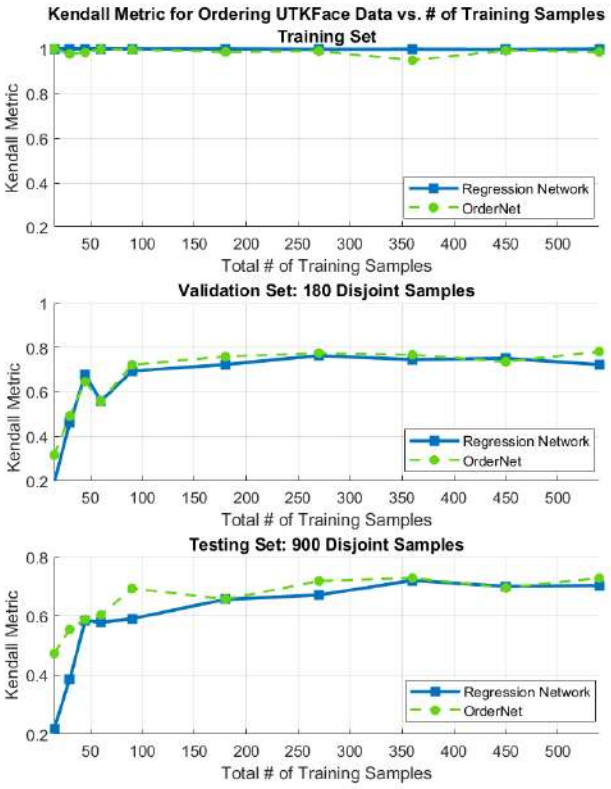

Fig. 6. Final training, validation, and testing performance for ordering the UTKFace dataset by age as a function of number of training samples

this would imply that, as the classification moves from a more discrete set of ordered classes to a more analog set of labels, OrderNet would be increasingly more beneficial than a regression network. We hypothesize this as a potential reason why the analog clocks (which had a fine classification dataset of 720 class labels) benefited much more than the UTKFace dataset (which had a much coarser dataset containing 90 classification labels). It is a possibility that the this effect was compounded by the much higher complexity of the UTKFace dataset features as well.

Future work aims to further refine the operating space of OrderNet by testing the network across a larger set of data. Additionally, we plan to integrate OrderNet into active learning and human in the loop applications where we would expect to see the most gain from the algorithm. Notably, we believe there are many ordering/organizational applications where human data labeling is tedious. In these applications OrderNet may prove to be useful as an interactive tool to reduce cost, time, and directed attention fatigue of humans working toward data organization and labeling.

ACKNOWLEDGMENT

This work was supported by grants from the Department of Energy #DE-NA0003946, National Science Foundation (NSF) CAREER #1943552, and NSF REU #1950359. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.
[2] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (gans): A survey," *IEEE Access*, vol. 7, pp. 36322–36333, 2019.
[3] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.
[4] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.
[5] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," *arXiv preprint arXiv:1904.04232*, 2019.
[6] J. Lu, P. Gong, J. Ye, and C. Zhang, "Learning from very few samples: A survey," *arXiv preprint arXiv:2009.02653*, 2020.
[7] M. Fink, "Object classification from a single example utilizing class relevance metrics," *Advances in neural information processing systems*, vol. 17, pp. 449–456, 2004.
[8] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.
[9] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, "One shot learning of simple visual concepts," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 33, 2011.
[10] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," *International Conference on Learning Representations*, 2017.
[11] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learn. Wrksp*, 2015.
[12] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in Neural Information Processing Systems*, pp. 737–744, 1994.
[13] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, pp. 4080–4090, 2017.
[14] M. Ye and Y. Guo, "Deep triplet ranking networks for one-shot recognition," *arXiv preprint arXiv:1804.07275*, 2018.
[15] T. R. Scott, K. Ridgeway, and M. C. Mozer, "Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning," *arXiv preprint arXiv:1805.08402*, 2018.
[16] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7278–7286, 2018.
[17] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," *arXiv preprint arXiv:1711.04340*, 2017.
[18] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn, "Bayesian model-agnostic meta-learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 7343–7353, 2018.
[19] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," *arXiv preprint arXiv:1806.02817*, 2018.
[20] J. Guo, Y. Fan, L. Pang, L. Yang, Q. Ai, H. Zamani, C. Wu, W. B. Croft, and X. Cheng, "A deep look into neural ranking models for information retrieval," *Information Processing & Management*, vol. 57, no. 6, p. 102067, 2020.
[21] X. Liu, J. Van De Weijer, and A. D. Bagdanov, "Rankiqa: Learning from rankings for no-reference image quality assessment," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1040–1049, 2017.
[22] B. Settles, "Active learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 6, no. 1, pp. 1–114, 2012.
[23] M. G. Kendall, *Rank correlation methods.* Griffin, 1948.
[24] S. Verma, "Analog-clocks," 2020.
[25] Z. Zhang, Y. Song, and H. Qi, "Age progression/regression by conditional adversarial autoencoder," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5810–5818, 2017.