How Robots Learn to Classify New Objects Trained from Small Data Sets

Tick Son Wang[1] **Zoltán-Csaba Márton**[1] **Manuel Brucker**[1] **Rudolph Triebel**[1,2]
[1] German Aerospace Center (DLR), Institute of Robotics and Mechatronics
[First.Last@dlr.de]
[2] Technical University of Munich, Department of Computer Science
rudolph.triebel@in.tum.de

Abstract: In this paper, we address the problem of learning to classify new object classes and instances by adapting a previously trained classifier. The main challenges here are the small amount of newly available training data and the large change in appearance between the new and the old data. To address this we propose a new variant of Progressive Neural Networks (PNN), originally introduced by Rusu et al. [1]. We show that by performing a specific simplification in the adapters, the prediction performance of the resulting PNN can be significantly increased. Furthermore, we give additional insights about when PNNs outperform alternative methods, and provide empirical evaluations on benchmark datasets. Finally, we also suggests a way of using it to augment the functionality of a network by extending it with new classes, addressing the problem of unbalanced classes, i.e. where the new classes are under-represented.

Keywords: Progressive Neural Network, Robotic Vision, Transfer Learning

1 Introduction

Many learning tasks in robotics are particularly challenging because the amount of available training data is not sufficient to obtain models that exhibit a low risk of overfitting. Especially if deep learning methods are employed, in order to mitigate that risk and to achieve a good generalization, the demand for annotated ground truth data is huge. On the other hand, the lack of high amounts of training data for a given target task is a practical problem for deep learning applications in robotics. Robots often encounter situations where new observations are significantly different from previous ones, be it new known objects observed under novel circumstances or a completely new object class. In both cases, the previously learned model must be adapted to account for this new information, but often without loosing the ability to recognize the previous cases. An additional problem is due to an unbalanced distribution of object classes, i.e. the new samples are usually under-represented compared to the old ones. In such cases one can employ transfer learning techniques, whose effect is especially intensified in the case of sparse target data, because as the target data becomes more self-sufficient, the benefit of the transfer over e.g. fine-tuning will be less visible.

To address these issues that are not exclusively pertaining to robotic learning, in this paper we investigate the Progressive Neural Network (PNN), a recent methodology introduced by Rusu et al. [1] that extends the existing network through adapter functions that map between the old and the new layers. PNNs have the advantage that the architecture of the target network is independent of the source network, giving room for more flexibility in its size and structure, while in fine-tuning the adaptation to the new data must be done with the old network structure (an overview of differences between PNNs and fine-tuning is given in Table 1). Here, we propose an improved variant of PNNs, as outlined in Figure 1, extending the contributions made by Rusu et al. [1] and increasing the practical applicability of the framework. While [1] concludes that PNNs are on par if not better than the fine-tuning approach, the experiments in this work, albeit using a different adapter design and task at hand, show different trends under different circumstances. To summarize our contributions:

- while reusing knowledge, we also augment the current task with extended functionality;
- for both new and the extended tasks, we investigate the constructions of the adapter;

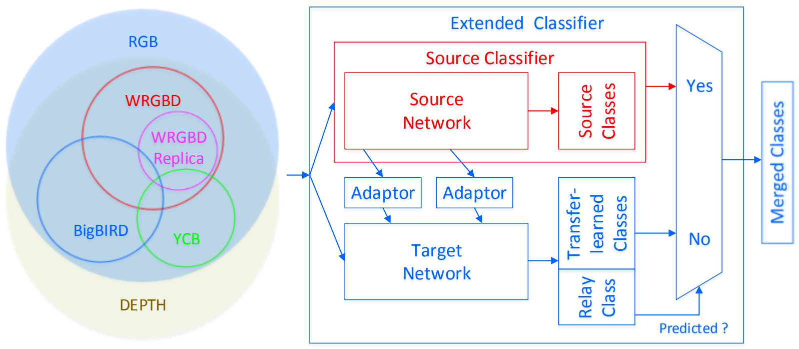


Figure 1: Graphical visualization of learning new objects with PNNs. The source classifier (red) consists of a network that is trained on a large training set, while the target network (blue) is smaller and has thus less parameters to be trained. To maintain the classification of the original classes, in the inference phase, the target network's output is queried first, and only if the predicted as "relay", the source network's output is used.

- we provide general insights into when PNNs work and perform better than fine-tuning;
- their effectiveness in improving performance when trained with sparse data is investigated;
- we focus on the issue of balancing and deciding between source and target classes;
- performing experiments on robotic vision benchmarks, and evaluating them statistically.

2 Related Work

The most common standard transfer learning approach in the field of deep learning is *fine-tuning*, where a previously trained network is re-trained with the new target data. Fine-tuning is used broadly, mainly for its good performance and comparably little effort. For example, Held et al. [2] pretrained a network with a multi-view datasets and then fine-tuned it with a single-view dataset of different object, showing that the new network inherits the multi-view robustness for the new objects as well. This has been an important reference for our experiment design where the different datasets cover different view-ranges and densities. Furthermore, Yosinski et al. [3] reported an extensive investigation into the transferability of CNN features through fine-tuning, showing that the transferability is correlated to the similarity between the source and target datasets. In cases where the two are too different, fine-tuning even yielded negative transfer, thus we chose our evaluation datasets to include both more and less similar data.

While it is possible to fine-tune a model pretrained with big color dataset using depth data, the result is generally unsatisfactory. The problem is to find a suitable 3-channeled embedding for depth data which matches the feature properties of color images. As one among many recent studies about inter-modal transfer learning, Song et al. [4] attempted to fine-tune a Places-CNN [5], which was trained with color images of places, with depth data in the proposed HHA embedding and concluded that it was less effective than training from scratch. The paper explained that the lower level filters are modality-specific and hence, low level color filters are ineffective in capturing important features in depth images. We show that in such cases the PNN approach is a promising potential alternative,

Table 1: Comparison between different aspects of fine-tuning and PNN for transfer learning.

Aspects	Fine-Tuning	Progressive Neural Network
Knowledge transfer mechanism	By directly changing the parameters of the pretrained network for the target task.	By combining transformed feature representations of the source network.
Knowledge preservation	Feature representation of the pretrained network is altered for the target task, information loss is possible.	Feature representation of the source network is preserved.
Flexibility of network architecture	Architecture of the pretrained network is to be fixed.	Architecture of target networks can be arbitrary.
Computation for single target task	Identical to pretrained network up to the last layer used.	Additional computation from the adaptors and the source networks.
Reusability for multiple tasks	Extend the pretrained network with multiple outputs or clone the pretrained network once for each task.	Multiple target networks can depend on one source network in parallel.
Possibility for life-long learning	Knowledge preservation is not guaranteed in successive fine-tuning.	Knowledge is accumulated at the cost of more target networks.

and by choosing RGB-D datasets (commonly used in the robotics domain), we are able to show the performance of fine-tuning and PNN on the same objects using both RGB and depth information.

Our work is closely related to that of Rusu et al. [1], who proposed the Progressive Neural Network (PNN) architecture for transfer learning between two game playing tasks. In Rusu et al. [6] they demonstrated transfer learning between simulations and reality, while this work uses PNNs for the task of learning new objects based on sparse data, with or without forgetting the original objects. Anderson et al. [7] proposed *Stitched Modules*, a very similar concept to PNNs, also along the paradigm of training a target network which receives input on every layer from a source network for transfer learning. The target network is obtained by cloning the source network and then during training, only the target network is trained using the target data but with additional lateral connections from the source network. Unlike PNNs, where the lateral connections allow feature adaptations, the lateral connections in the *Stitched Modules* only serve to pass the original feature maps layer-wise from the source network to the target network. This constraints the architecture of the target network to be identical to that of the source network which reduces the efficiency and the flexibility of the concept. In addition to that, it did not explore the potential of feature adaptation via the lateral connections. While [7] also emphasized the notion of learning new task with small amount of data (which is an important topic in this work), it only provided evidence that the *Stitched Modules* is better than traditional fine-tuning for transfer learning from CIFAR-10 to CIFAR-100. It did not explore different source-target combinations in order to show a consistent trend.

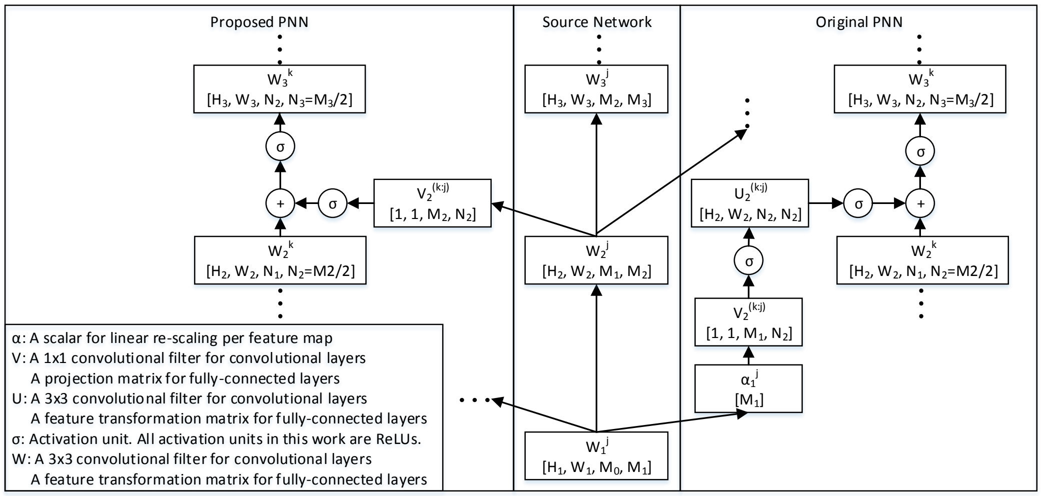
Recently Kaiser et al. [8] presented a *MultiModel* architecture which is designed to learn different tasks across different domains using data of different modalities (images, speech, text). Its uniqueness is that it has an organized framework for the encoding of inter-modal data at the input, the processing of fused information via a mixture of experts in different task domains and then decoding of the experts output back to different human-interpretable modalities. There is also some research specifically focusing on techniques to compel the training process to achieve certain desired domain adaptation effects by adding additional objectives into the loss function [9, 10]. Another popular technique for transfer learning is to use deep networks as fixed feature extractors for traditional learning methods. Sharif Razavian et al. [11] benchmarked this technique using the CNN features from the OverFeat model against methods using handcrafted features which were state-of-the-art during that time. Even though this is a valid technique, it is not considered in this work because fundamentally, this technique does not have the capacity to improve the network.

3 Network Architecture

As the base architecture for the PNN, we employ CaffeNet [12] with some modifications. CaffeNet was selected because among most of the well-known CNN architectures such as the likes of ResNet [13] and VGGNet [14], it requires far less computation to train and far less memory consumption. Sacrificing higher accuracy of deeper and more complex models for shorter training time and lower memory cost was a conscious trade-off made in this work because the main aim of the experiment was to carry out extensive experiments instead of surpassing benchmarks.

In terms of modifications, the max pooling layer after the last convolutional layer is replaced with an global average pooling layer. This modification provides the network the capability to accept input image of arbitrary height and width without the need to change the implementation. This increases the flexibility of the network to potentially be reused for dataset of different image sizes. Next, the numbers of hidden units in the fully connected FC6 and FC7 layers are increased to 2048 each. The same architecture will also be used for the target network with the exception that the size of the network is halved (i.e. both the number of convolutional layer filters of the target network, and the number of hidden units in the fully-connected layers, are halved compared to the source network).

The general purpose of a PNN is to perform a new task by not only learning new feature representations from the given new data but also simultaneously deriving useful ones from the attached source network. The target networks can assume the architecture of any conventional types of DNN, or in the context of this work, any variants of CNN. This part provides the capacity to discover novel patterns or information from the target data. The adapter, which refers to the lateral connections between a source network and a target network, is the key component of a PNN. It is assumed that the feature maps of the source network will not be optimal for the target task, hence the adapters work as a layer of transformation to select and modify these feature maps so that they



α: A scalar for linear re-scaling per feature map
V: A 1x1 convolutional filter for convolutional layers
A projection matrix for fully-connected layers
U: A 3x3 convolutional filter for convolutional layers
A feature transformation matrix for fully-connected layers
σ: Activation unit. All activation units in this work are ReLUs.
W: A 3x3 convolutional filter for convolutional layers
A feature transformation matrix for fully-connected layers

Figure 2: Structure of original and proposed PNN adapter (see legend and main text for details).

become more suitable and specific to the target task. With the bias terms omitted for clarity, the original design of the adapter proposed by [1] can be generally defined as (see Figure 2):

$$h_i^{(k)} = \sigma\big(W_i^{(k)} h_{i-1}^{(k)} + U_i^{(k:j)} \sigma(V_i^{(k:j)} \alpha_{i-1}^{(<k)} h_{i-1}^{(<k)})\big) \quad (1)$$

The original design of the adapter proposed by [1] is functionally sensible, it combines all possible transformations arranged in a logical manner. However, this combination of transformations is very complex and computationally very costly. To improve this, this work proposes a more efficient design for the adapter.

An aspect in the design of the adapter is the compatibility of feature map dimensions. The original adapter design combines $W_i^{(k)} h_{i-1}^{(k)}$ and $U_i^{(k:j)} \sigma(V_i^{(k:j)} \alpha_{i-1}^{(<k)} h_{i-1}^{(<k)})$. In cases where the size of feature map decreases across the layers, which are typical, $U_i^{(k:j)}$ has to have the same filter size as $W_i^{(k)}$ or otherwise it might encounter a mismatch in feature map dimensions. This work proposes to combine $W_i^{(k)} h_{i-1}^{(k)}$ and $U_i^{(k:j)} \sigma(V_i^{(k:j)} \alpha_i^{(<k)} h_i^{(<k)})$ instead. Since $h_i^{(<k)}$ already has the same dimension as $W_i^{(k)} h_{i-1}^{(k)}$, parametrizing the dimensions of the filters in the adapters becomes trivial. This opens up possibilities for a more efficient design the adapter.

Under the assumption that the source network contains useful information for the target task, its original feature representations should be directly useful to the target task without complex transformation. In fact, complex transformations tend to invite higher chances of over-fitting. Based on this view, this paper proposes to use only a projection in the adapter because it is the bare necessity by network construction in order to match the depths of feature maps between the source and the target network. Besides, once the information flows into the target network, there is plenty of capacity for complex transformations. Using the same notation, the proposed adapter is given as:

$$h_i^{(k)} = \sigma\big(W_i^{(k)} h_{i-1}^{(k)} + \sigma(V_i^{(k:j)} h_i^{(<k)})\big) \quad (2)$$

Figure 2 shows a detailed graphical representation and a short description regarding the components of the original and the proposed adapter designs using the same notation as [1]. It intuitively displays how the feature maps from the source network are transformed through the adapters.

4 Experimental Results

Since Rusu et al. [1] considered Deep Reinforcement Learning to play atari games while we use Deep Learning to classify images, we can not use the same benchmark. However, we evaluate both the original and proposed method on standard robotics-related object recognition datasets. The datasets used for evaluation in this work are the Washington University's RGB-D Object Dataset

(WRGBD) [15], the Yale-CMU-Berkeley Object and Model Set (YCB) [16] and the Big Berkeley Instance Recognition Dataset (BigBIRD) [17]. These datasets are similar multi-view datasets, where objects of daily use were placed at the center of a rotating turn-table and images were recorded from different elevation angles. This means that there is a high chance for many low level features to be shared across these datasets. Moreover YCB and BigBIRD used the exact same setup, and we are also using a dataset aimed to replicate WRGBD, which we will refer to as the WRGBD-Replica. It contains images of 51 objects captured using the same setup as WRGBD, each of the objects is one new and distinctive instance of each of the WRGBD categories. For training and validation, we only use a subset of samples from the elevation angles of $\{30°, 60°\}$ for WRGBD/WRGBD-Replica. For training, we sample every 5° and 40° for the cases of source and target data. For validation, we sample every 20° but starting at an offset of 3° and 10° for source and target data. In the case of BigBIRD/YCB, we take samples from elevation angles of $\{0°, 45°, 90°\}$ for training and validation. For training, we sample every 6° and 90° for the cases of source and target data (we also performed experiments with every 30° sample as target data as which are only mentioned in the conclusions due to space limitations). For validation, we sample every 30° and 90° but starting at an offset of 15° and 45° for source and target data. For testing, we use all samples in the unused elevation angles. We deliberately reduce the training and validation data for the target task to increase the gap in information content between source and target. This makes the effect of transfer learning more obvious, and simulates the practical scenarios in robotics where often only sparse data is available.

The quality of the source network is a critical factor for success in transfer learning. To enforce fair competition between different transfer learning techniques, each set of experiments shared the same instance of the source network so that they started from the same state. In order to maximize the performance of a network, it is common to optimize the training in terms of aspects such as the choice of the optimizer, the number of training iterations, the learning rate and the regularization. However, we will use a set of general configurations throughout all experiments, allowing us to analyze the experimental results without the necessity to factor in these aspects. In order to draw conclusions that would most likely also hold if the results were to be tuned, we will focus on the results where differences are largest, and strive for achieving statistically significant differences.

In the interest of simplicity and consistency, RMSProp [18] is chosen for all experiments in this paper. RMSProp is an adaptive optimization method which individually scales the update of each trainable parameter based on the running average of the magnitude of previous gradients. Since this ability automatically controls the magnitude of the updates, it can be considered as a form of adaptive learning rate and thus it reduces the influence of the actual learning rate on the optimization process. Unlike optimization methods such as the Stochastic Gradient Descent where usually a decaying learning rate is needed, it does not require such special care (at the cost of slightly larger variance). As a benefit of using the RMSProp optimizer, the choice of the learning rate becomes less influential to the process of training, which means that it is possible to apply the same learning rate when training with different datasets. Nevertheless, it is still not possible to apply the same learning rate when training with different approaches. Since lower learning rate generally yields better result, this work roughly chooses the lowest possible learning rates for the different training approaches such that all the training is guaranteed to converge within the specified number of training iterations. Since the random elements during training cause deviations in the resulting optimum which could affect the performance of the network. To produce more reliable results, the experiments in this work were repeated for 10 times using the same setting. The results are presented in the form of violin plots or the median (m) and the Median Absolute Difference (MAD) in tables, and perform the nonparametric (i.e. no assumption is made on the distribution of the data) Wilcoxon-Mann-Whitney test [19] for unpaired data (the 10 trials being random) to reject the null hypothesis of no accuracy improvement to be expected when using the proposed approach instead of the best baseline[1].

4.1 Proposed Progressive Neural Network Design

Table 2 shows the performance differences between different adapter designs and it provides a number of observations: (i) the additional scale transformation in the adapter before the projection did not have a positive effect, likely because the 1x1 projection filter is already capable of scaling the input feature maps and thus the extra scaling capacity is redundant; (ii) the Filter-adapter performed slightly worse than the Project-adapter, which suggest that complex feature transformation is

[1]PNN of [1], train-from-scratch or fine-tuning (complete fine-tuning is used as training time is uncritical)

Table 2: Test accuracies [%] using different adapter designs for transfer learning from BigBIRD to WRGBD.

Adapter Design	Original PNN	Filter	Scale + Project	Filter + ReLU	Scale + Project + ReLU	Project	Project + ReLU
$m \pm$ MAD	69.7 ± 0.4	72.3 ± 0.5	72.8 ± 0.6	73.4 ± 0.7	73.5 ± 0.3	73.6± 0.4	74.7 ± 0.4

Table 3: Test accuracies [%] of transfer learning techniques on different combinations of source and target. Pairwise differences between best (**bold**) and second best (*italic*) performance are significant ($p < 0.1\%$).

Dataset	BigBIRD → WRGBD	WRGBD → YCB	BigBIRD → YCB	WRGBD → BigBIRD	YCB → BigBIRD
Scratch	*71.68 ± 1.56*	59.44 ± 1.49		29.91 ± 0.89	
Fine-tune	71.38 ± 0.78	63.79 ± 1.18	73.37 ± 1.01	27.50 ± 0.94	46.95 ± 0.76
Ori. PNN	69.72 ± 0.35	*70.20 ± 0.82*	*75.90 ± 0.95*	*33.74 ± 4.39*	*47.27 ± 0.82*
Prop. PNN	**74.71 ± 0.39**	**74.55 ± 0.74**	**78.97 ± 0.60**	**44.95 ± 1.03**	**51.14 ± 0.57**

undesirable; (iii) the non-linear activation with ReLU seems to be marginally beneficial; and (iv) the original adapter design, being the most complex design amongst all, achieved the worst result with significantly lower performance compared to the other designs. Additionally, by repeating the experiment with all the convolutional layers of the target network left out, we show that the capacity for new discoveries and adaptations of the low and intermediate level features is actually essential. Without them, the performance of the proposed PNN drops significantly from $\approx 74\%$ to $\approx 64\%$.

4.2 Knowledge Transfer to Sparse Target Dataset

PNNs work well on limited target data but becomes less effective as the gap of information content between source and target data decreases because this means that the source will have less useful information to offer that is not already learnable from the target. To show the effectiveness of the proposed PNN, we compare its accuracy with that of training a new network from scratch, fine-tuning and the original PNN. The results are shown in Table 3. Note that only the highlighted combination was used in subsection 4.1, where the adapter design for the proposed PNN was selected, avoiding a model selection bias across the different scenarios. WRGBD seems to be an ineffective source compared to YCB and BigBIRD. This is not surprising considering that YCB and BigBIRD share an identical data acquisition setup, while WRGBD does not (WRGBD covers only one third of the camera elevation angles' range in YCB and BigBIRD). Even though Table 3 shows that PNN performed better, Figure 3 shows a counter result, where fine-tuning performed better at transfer learning from WRGBD to WRGBD-Replica for RGB data. We believe that fine-tuning should be superior at adapting feature representation for a target task because it can directly manipulate the parameters. However, if the target information domain has a big shift from the source information domain (see depiction in Figure 1), learning the target feature representation could severely distort the source feature representation and result in information loss. We argue that this is the reason why PNN performed better at other transfer dataset combinations but not for WRGBD to WRGBD-Replica (in the experiment using depth data PNN performed better probably because our depth measurements had lower quality than in WRGBD).

To illustrate this effect in the case of RGB, we can make use of the widely used CaffeNet pretrained on ImageNet, whose large generality encompasses the variance found in the RGB part of all used

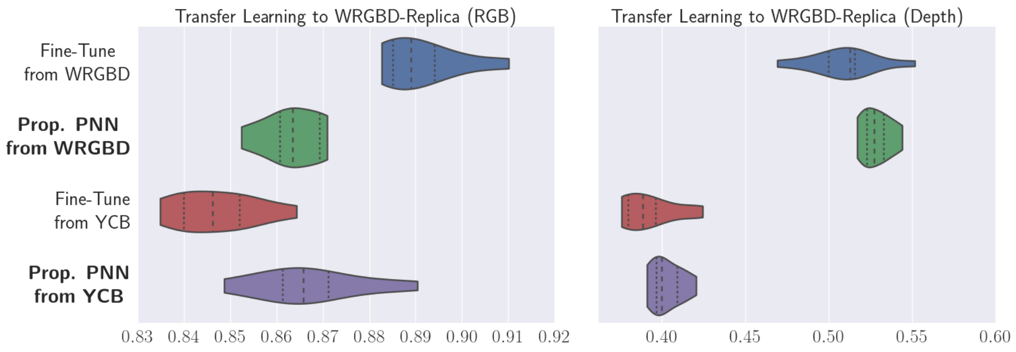


Figure 3: Transfer Learning to WRGBD-Replica. The differences between the pairs are significant ($p < 5\%$).

Table 4: Accuracies [%] of the PNN extended classifier with different number of relay classes.

Nr. Relay Classes	1	7	21	35	70	125
$m \pm$ MAD	**70.9 ± 0.5**	70.7 ± 0.3	69.8 ± 0.4	69.7 ± 0.2	70.4 ± 0.3	70.4± 0.6

datasets. Between fine-tuning versus PNN, the results are $\approx 88\%$ vs $\approx 82\%$ for the case of WRGBD and $\approx 85\%$ vs $\approx 77\%$ for the case of YCB. However, in the case of depth (replicated to 3 channels), they performed similarly ($\approx 53\%$ vs $\approx 52\%$), supporting our hypothesis on when PNNs make sense. Since there are several areas apart from depth images where a generic dataset like ImageNet is missing (e.g. medical, space, hyperspectral, thermal, etc. imaging) and only limited labeled data is available (e.g. industrial applications), PNNs offer a viable way to improve DNN performance.

As an additonal comparison, following the idea presented in [20], see [21], where a traditional classifier was trained on features extracted by an ImageNet pre-trained network. On the same WRGBD instance recognition benchmark they reached $\approx 86\%$, a value between our fine-tuning and PNN results, when using approximately the same ammount of RGB training data (cf. Fig 8 in [21]). In our similar experiments comparing fine-tuning with an SVM or Random Forest on CNN features in [22], we found that fine-tuning performs better when done on similar data, but worse when on differring data. A further improvement was achieved when combining both fine-tuning on similar data and training another classifier afterwards that replaces the softmax layer, re-using the same training and validation data (there a ResNet50 was used on our own object and scene data).

4.3 Extending a Classifier with a Progressive Neural Network

Although PNNs were originally proposed for the purpose of transferring information from a source to a new target network, here we propose a different perspective, that one can also view the new network as a way to extend the functionality of the source network given new data. Specifically, we investigate how a PNN can extend a classifier with new classes while at the same time improve the classification accuracy of the new classes with the information stored in the source network.

This means that the extension has to be reversible by simply detaching the extended network, i.e. keeping the preceding networks fixed. When making new changes, the succeeding network does not necessarily have to be appended in a linear path, it can be branched out at any network along the growth path. This gives the PNN a natural way to cope with dynamic incremental changes to the classification problem while preserving the integrity of the network at each point of the changes. Since the succeeding network is only expected to accommodate for a certain task modification, its size should be relative to the degree of the modification, figuratively speaking. In theory, each successive network can be pruned to its minimal size and this makes a PNN an efficient way of incrementally harboring information for the corresponding incremental modifications.

There are two problems we are investigating here: how to best distinguish if the test data comes from the original or the extended classes, and how to balance the data for training the new classifier.

4.3.1 Balancing the Number of Original and New Classes in the Target Network

Since the source network has to be kept fixed, we are introducing the decision between old and new classes in the new network. Given that the original network's performance on the original classes is already very good, it suffices that the extended network relays the classification decision when a sample is estimated to come from the original classes. Therefore one or more *relay classes* are introduced, whose purpose is to allow the target network to detect source data. If the test data falls into a relay class of the target network, the classification result is taken from the source network's output. These special target network classes could be all original classes grouped into one, kept as individual classes, or grouped together in different number of classes.

To investigate if the number of relay classes will affect the accuracy of the extended classifier, we are employing a hierarchical clustering of the original classes based on the confusion matrix obtained on the validation data of the original network. We present the results on different levels of clustering in Table 4. The single relay class PNN extended classifier yields the best accuracy.

Table 5: Individual accuracies and dataset confusion rates (DCR) [%] of differently trained extended classifiers.

Model	Mixed Data	Overall acc.	Source acc.	Target acc.	Source DCR	Target DCR
Fine-Tuned	Reduced	56.4 ± 0.91	51.0 ± 1.05	65.7 ± 0.63	$\mathbf{15.5 \pm 0.67}$	16.1 ± 0.74
	Original	64.3 ± 0.28	68.6 ± 0.18	57.7 ± 0.77	27.8 ± 2.91	$\mathbf{4.1 \pm 0.69}$
Proposed PNN	Reduced	*70.9 ± 0.54*	*72.0 ± 0.69*	$\mathbf{69.0 \pm 1.34}$	*15.5 ± 0.88*	7.5 ± 1.38
	Original	$\mathbf{70.9 \pm 0.24}$	$\mathbf{73.5 \pm 0.39}$	*66.8 ± 0.89*	16.3 ± 0.10	*5.7 ± 0.77*

4.3.2 Balancing the Training Data from Source and Target for the Extended Classifier

For the training of the extended classifier, the target data is required to recognize the target classes while the source data is required to recognize the relay class. Since the source data is many times denser than the target data, there are different possibilities to compose the mixed training data. If the full source set is used together with the target data (*Original* mixed data) to train the extended classifier, then its predictions are expected to be skewed towards the source classes reducing the accuracy for the target classes. To avoid this prediction skewness, it is also possible to reduce the density per source class of the source data to match the density per target class of the target data (*Reduced* mixed data). Note that since the source network is already pretrained with the entire source data, it should be able to provide all learnable information in it.

As a baseline, we compare the effectiveness of the proposed PNN to fine-tuning, i.e. exchanging the last layer of the source network with a FC layer of more output neurons and fine-tune the whole resulting network with the mixed training data. Table 5 shows the result of the experiment for the cases of training with the *Original* and *Reduced* mixed data. Aside from the accuracies, the table also reports the source and target dataset confusion rate (DCR). Source DCR refers to the portion of target data predicted as any of the source classes while target DCR refers to the opposite case. The source network used in this experiment is trained with BigBIRD, reaching an accuracy of 76.7%. To measure the element of transfer learning in this context, we also trained a network of the same architecture from scratch with the sparse target data (YCB) and it achieved an accuracy of 59.4%. Since both fine-tuning and PNN showed higher target accuracy than that, it implies that the extended classifiers benefited from transfer learning for the target classes (at the cost of the source classes).

The PNN yielded better accuracies than fine-tuning for both types of mixed data. For the fine-tuned model, all accuracies are heavily influenced by the composition of the mixed data. When it is trained with the *Original* mixed data, in which the density of the source data is much higher, it achieves significantly higher source accuracy. However, its prediction skewness towards the source classes, as indicated by its high source DCR, is also the reason why it has lower target accuracy. Importantly for the PNN, even though the composition of the mixed data shows similar influences, the magnitude is significantly smaller. Compared to the source classifier, the fine-tuned model dropped by $\approx 20\%$ in source accuracy when trained with *Reduced* mixed data but PNN only dropped by $\approx 3\%$ (the reason for the drop is the relay class inaccuracy). This demonstrates the increased stability of our PNN in inheriting the source classifier's classification capacity. The reduction can be done whenever the source dataset is larger than the target. While we did not perform incremental progression with several steps, the balancing can also be done in such cases as well.

5 Discussion and Conclusion

In terms of the effectiveness of the experimented transfer learning approaches, the proposed PNN seems to be consistently the best performing transfer learning technique compared to the original PNN design and fine-tuning in all experiments where no dataset replication was attempted (subsection 4.2). The transfer between YCB and BigBIRD in both directions shows a tipping point after the $30°$ sampling mark of the target data. At that point most alternative approaches failed to improve over training from scratch, but the proposed PNN is yet able to maintain a small improvement, in median $\approx 2\%$ on YCB and $\approx 1\%$ on BigBIRD that are still statistically significant.

In terms of extending a classifier with new classes, the proposed PNN outperformed fine-tuning, and had the advantage that its dataset-wise accuracies were not as strongly influenced by the reduction of source data for training, thus allowing the user to use the sparser source data for increased efficiency. We plan to investigate this idea of extending a source network in different contexts as well, such as detection, regression and segmentation in the future. Since the information transfer happens mainly on the feature representation level, it should be independent of the predictive mechanism afterward.

Acknowledgments

The authors would like to thank the reviewers for their useful comments, and the Research and Technology Center of Robert Bosch GmbH in Palo Alto for their support. This work has been partially funded by the RobDREAM projet of the H2020 framework programme of the European Comission under the grant agreement number 645403.

References

[1] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[2] D. Held, S. Thrun, and S. Savarese. Robust single-view instance recognition. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2152–2159. IEEE, 2016.

[3] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[4] X. Song, L. Herranz, and S. Jiang. Depth CNNs for RGB-D scene recognition: Learning from scratch better than transferring from RGB-CNNs. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[5] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

[6] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

[7] A. Anderson, K. Shaffer, A. Yankov, C. D. Corley, and N. O. Hodas. Beyond fine tuning: A modular approach to learning on small data. *arXiv preprint arXiv:1611.01714*, 2016.

[8] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.

[9] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *Intern. Conf. on Machine Learning*, pages 97–105, 2015.

[10] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *Proc. of the IEEE Intern. Conf. on Computer Vision*, pages 4068–4076, 2015.

[11] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recog.*, pages 770–778, 2016.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[15] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *Robotics and Automation (ICRA), IEEE Intern. Conf. on*, pages 1817–1824. IEEE, 2011.

[16] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In *Advanced Robotics (ICAR), 2015 International Conference on*, pages 510–517. IEEE, 2015.

[17] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 509–516. IEEE, 2014.

[18] T. Tieleman and G. Hinton. Lecture 6.5 - rmpsprop. *COURSERA: Neural Networks for Machine Learning*, 2012.

[19] M. Hollander, D. Wolfe, and E. Chicken. *Nonparametric statistical methods*. Wiley Series in Probability and Statistics - Applied Probability and Statistics Section. Wiley, 2014. ISBN 978-0-470-38737-5.

[20] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML)*, volume 32 of *Proceedings of Machine Learning Research*, pages 647–655, Bejing, China, 22–24 Jun 2014. PMLR.

[21] M. Schwarz, H. Schulz, and S. Behnke. RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1329–1335, May 2015.

[22] M. Durner, S. Kriegel, S. Riedel, M. Brucker, Z.-C. Marton, F. Balint-Benczedi, and R. Triebel. Experience-based optimization of robotic perception. In *IEEE International Conference on Advanced Robotics (ICAR)*, Hongkong, China, July 2017. Best Paper Finalist.