ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

CONTINUAL LEARNING WITH SPARSE PROGRESSIVE NEURAL NETWORKS

M.Sc. THESIS

Esra ERGÜN

Department of Computer Science

Computer Science Programme

JULY 2020

ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

CONTINUAL LEARNING WITH SPARSE PROGRESSIVE NEURAL NETWORKS

M.Sc. THESIS

Esra ERGÜN
(704181008)

Department of Computer Science

Computer Science Programme

Thesis Advisor: Assoc. Prof. Dr. Behçet Uğur TÖREYİN

JULY 2020

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

SEYREK İLERLEMELİ SİNİR AĞLARI İLE SÜREKLİ ÖĞRENME

YÜKSEK LİSANS TEZİ

Esra ERGÜN
(704181008)

Bilgisayar Bilimleri Anabilim Dalı

Bilgisayar Bilimleri Programı

Tez Danışmanı: Assoc. Prof. Dr. Behçet Uğur TÖREYİN

TEMMUZ 2020

Esra ERGÜN, a M.Sc. student of ITU Informatics Institute student ID 704181008 successfully defended the thesis entitled "CONTINUAL LEARNING WITH SPARSE PROGRESSIVE NEURAL NETWORKS", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Behçet Uğur TÖREYİN** Istanbul Technical University	.....
Jury Members :	**Assoc. Prof. Dr. Nazım Kemal ÜRE** Istanbul Technical University	.....
	Asst. Prof. Dr. Mehmet TÜRKAN İzmir University of Economics	.....
		.....

Date of Submission : **15 June 2020**
Date of Defense : **21 July 2020**

To my family,

FOREWORD

I want to thank my supervisor Assoc. Prof. Dr. Behçet Uğur Töreyin for supporting me throughout my Continual Learning adventure. I also want to thank my family and friends for supporting me throughout my life. This thesis is supported in part by Istanbul Technical University (ITU) Vodafone Future Lab under Project No. ITUVF20180901P04.

July 2020 Esra ERGÜN

TABLE OF CONTENTS

Page

ABBREVIATIONS

AI	**:** Artificial Intelligence
CF	**:** Catastrophic Forgetting
CL	**:** Continual Learning
MLP	**:** Multi Layer Perceptron
ANN	**:** Artificial Neural Network
PNN	**:** Progressive Neural Networks
CNN	**:** Convolutional Neural Network
FC	**:** Fully Connected Layer
LASSO	**:** Least Absolute Shrinkage and Selection Operator

SYMBOLS

M	**:** An Artificial Neural Network Model
W	**:** Weight parameter of the model
b	**:** Bias parameter of the model
h	**:** Hidden layer output
t	**:** Number of tasks in a continual learning scenario
K	**:** Number of layers in a network
X	**:** Input Data
Y	**:** Target Data
o	**:** Classification layer
U	**:** Learnable parameter matrix of adapter layer
α	**:** Learnable weights
λ	**:** Learning Rate
s	**:** Sparsity Ratio

LIST OF TABLES

Page

LIST OF FIGURES

Page

CONTINUAL LEARNING WITH SPARSE PROGRESSIVE NEURAL NETWORKS

SUMMARY

Artificial Neural Networks address major problems of computer vision, natural language processing and data science in the last decade with the increasing computational power and amount of data. Despite its popularity and success, when presented with a sequence of tasks with only having access to current task's data, neural networks fail to preserve its performance on previously learned tasks. This problem is called catastrophic forgetting and one of the biggest obstacle on the way of artificial general intelligence. Continual Learning is defined as the research field of learning a collection of tasks sequentially without suffering from catastrophic forgetting while improving the forward and the backward transfer across tasks.

Human brain does not suffer from catastrophic forgetting because each task builds on top of the anterior tasks by effectively integrating skills and fine tuning. Inspired by human intelligence, this thesis builds on top of progressive neural networks. In this study, to continuously learn a visual classification task sequence, several progressive neural network models that exploits weight spaces with lateral connections are investigated. There are three main purposes of this thesis. First, sparsification of the progressive neural networks, second, decreasing the redundancy of weight spaces and investigating the effect of task order on performance.

Number of parameters of a Progressive Neural Network increases with the arrival of a new task. Therefore, in order to increase the efficiency of PNNs, the first part of this thesis focuses on sparsification. To achieve this, two methods are investigated: binary masks and sparse group Least Absolute Schrinkage and Selection Operator (LASSO) regularization. These methods are used for sparsifying the weights and activations of a PNN. Furthermore, in order to decrease the feature redundancy, projection of current tasks' weight matrix to parameters of previous tasks and mean squared error are employed as additional terms to the loss. The goal is to employ relevant features from previously trained tasks and create compact progressive neural networks. Lastly, the effect of various prior tasks on current task's performance is analyzed.

Experiments are carried out on KMNIST, FMNIST, MNIST, permutedMNIST and selected subtasks from CIFAR-100 dataset. In all scenarios a multi-head output settings are considered where a new classification layer is initialized within the arrival of a new task. We show that encouraging feature novelty on Progressive Neural Networks (PNN) prevents major performance decrease on sparsification. Results demonstrated that sparsification of a PNN produces fair results and decreases the number of learned task-specific parameters on novel tasks. Moreover, in progressive settings, type of the prior task affects the performance of current task. A similar task prior task boosts accuracy while a dissimilar task harms the performance.

SEYREK İLERLEMELİ SİNİR AĞLARI İLE SÜREKLİ ÖĞRENME

ÖZET

Biyolojik zeka öğrenilmiş becerileri yeni karşılaşılan görevlere transfer edebilmekte, bu girişleri, zamana bağlı, öncül bilgi cinsinden ifade ederek öğrenebilmektedir. Daha önce öğrenilen beceriler, yeni öğrenilen becerilerin performansını artırmakta, yeni öğrenilen beceriler de, geçmiş becerilerin pekişmesini sağlamaktadır. Yapay sinir ağları son yıllarda birçok görsel, işitsel ve sekans verilerinde başarım göstermiştir ancak bir dizi görev üzerinde performans sağlamak konusunda yetersizdir. Yapay Sinir Ağları optimizasyona dayalı doğaları nedeni ile birbiriyle ilişkili veya ilişkisiz bir dizi görev ile karşılaştıklarında "yıkıcı unutkanlık" problemi ile karşılaşmaktadırlar. Eğitim yapılmış bir model başka bir görev için eğitildiğinde daha önce öğrenilen görevlerde ciddi performans düşüşü yaşamaktadır. Bu problem yapay genel zeka yolundaki en büyük engellerden biridir.

Yıkıcı unutkanlık problemi ve görsel verilerde sınıflandırma yapay zekanın ve bilgisayarla görü alanının üzerinde yoğun derecede çalıştığı alanlardır.

Bir sürekli öğrenme senaryosunun amacı, art arda gelen görev dizisini yıkıcı unutkanlık problemiyle karşılaşmadan çözebilmek, ileri ve geri bilgi transferini arttırmaktır. Amaç, geçmişte öğrenilen becerilerin gelecekteki görev başarımını ve verimliliğini, yeni öğrenilen becerilerin geçmiş görevlerde verimliliği arttırmasıdır. Eldeki görev öğrenilirken, daha önce karşılaşmış görevlerin dağılımlarına erişim yoktur ve bu durum problemi zorlaştırmaktadır. Varolan yöntemler üç ana gruba ayrılmaktadır: model yapısının değişmesine dayalı yöntemler, tekrarlama mantığına dayalı yöntemler ve regülarizasyona dayalı yöntemler.

İlerlemeli Sinir Ağları, sürekli öğrenme için pekiştirmeli öğrenme problemleri üzerinde denenmiş model değişimine dayanan yöntemlerden biridir. Yeni görevler ile birlikte bu görevlere özel parametreler tanımlanır ve daha önce öğrenilen görevlere ait parametrelerin güncellenmesi durdurulur. Bu model yapısı gereği, yıkıcı unutkanlık problemine bağışıktır. Geleneksel İlerlemeli Sinir Ağları'nın pekiştirmeli öğrenme problemlerinde başarımı gösterilmiştir. Bir eksiği ise parametre sayısının görev sayısıyla beraber karesel artmasıdır. Yanal bağlantılar ile her görev için yeni bir parametre kümesi öğrenmek eğitim örnek setinin ezberlenme riskini arttırmakta, matris çarpımlarını pahalılaştırmakta, ihtiyaç duyulan güç ve hafızayı arttırmaktadır. Yapay sinir ağlarında ve sürekli öğrenme problemlerinde görevin zorluğuna bağlı olarak farklı boyutta modeller gerekebilir. Yapay sinir ağlarında seyreklik modellerin regülarizasyonu amacıyla kullanılmaktadır. Seyreklik sayesinde matris çarpımları daha hızlı gerçekleştirilebilmekte ve sinir ağı modelleri seyrek matrisler kullanılarak daha verimli depolanabilmektedirler. Seyreklik nöron seviyesinde ya da ağırlık seviyesinde sağlanabilmektedir. Regülarizasyon katkıları ve hız-depolama verimliliğiini arttırmaları, biyolojik sinir ağlarına yakınlıkları ve

gürültüye karşı dayanıklılıkları nedeni ile sinir ağlarında seyreklik çokça çalışılmış ve çalışılmakta olan bir alandır.

Özellikle gürültünün fazla olduğu çerçevelerde ezberlemenin önüne geçen seyreklik ağırlık ya da nöron seviyesinde farklı yöntemlerle elde edilebilmektedir. Bunlardan biri, bağlantının şiddetiyle ilişkili olarak ağırlık budama veya maskelemedir. Bunların yanı sıra regülarizasyon terimi ile seyreklik elde eden çalışmalar da vardır. En Küçük Mutlak Daralma ve Seçme Operatorü regülarizasyonu bunların bir örneğidir. Literatürdeki mevcut çalışmalar ağırlıkları sıfıra iterek ezberlemeyi azaltmaktadırlar.

Bu tezde, model yapısının değişmesine bağlı yöntemler grubuna ait, yanal bağlantıları kullanan, İlerlemeli Sinir Ağları modelinin birkaç çeşidi bir dizi görsel sınıflandırma problemleri üzerinde denenmiştir. Daha önce pekiştirmeli öğrenme probleminde kullanılan İlerlemeli Sinir Ağlarının görsel sınıflandırma görevlerindeki başarımı ve ileri transfer yeteneği FMNIST, KMNIST ve MNIST ve CIFAR-100 veritabanları kullanılarak ölçülmüştür. İlerlemeli Sinir Ağlarının verimini arttırmak amacıyla aktivasyon ve ağırlık seyrekliğine gidilmiştir, bunun için maskeleme ve hata fonksiyonunun regülarizasyonu incelenmiştir. Maskeleme yönteminde, aktivasyonlar veya ağırlıklar sıralanmış ve en yüksek değere sahip K nöron dışındaki aktivasyonlar veya ağırlıklar sıfıra eşitlenmiştir. Hata fonksiyonuna terim ekleyerek yapılan regülarizasyonda Grup Seyrek LASSO regülarizasyonu kullanılmıştır. Grup seyrek LASSO regülarizasyonu bir nöronun bağlantılarının grup olarak sıfıra gitmesini sağlar. Bu regülarizasyonun geleneksel ağırlık regülarizasyonu l_p normlarından farkı, grup bazında seyreklik ile bir nöronun modelden tamamen çıkarılmasını sağlayabilmektir. Her iki seyreklik yöntemi ile beraber, öznitelik uzaylarının birbirinden farklı olmasını teşvik edici izdüşüm regülarizasyon terimi eklenmiş ve etkisi incelenmiştir.

Maskeleme yönemi ile elde edilen seyrek ilerlemeli sinir ağları incelemesinde FMNIST, KMNIST VE MNIST veri setleri kullanılmıştır. Model farklı sıralarda üç verisetiyle eğitilmiş, görev sırasının, yanal bağlantıların ve regülarizasyon terimlerinin performansa etkisi raporlanmıştır. Bu deneylerde tam bağlantılı katmanlar ve konvolüsyonel katmanların kullanıldığı iki farklı ilerlemeli sinir ağı modeli kullanılmıştır.

Seyrek Grup LASSO regülarizasyonu ile elde edilen ilerlemeli sinir ağlarında da hem çok katmanlı tam bağlantılı sinir ağı hem de konvolüsyonel sinir ağlarında MNIST verisetinin piksellerinin rastgele yer değiştirilmesiyle elde edilen permutedMNIST veriseti kullanılmıştır. Maskeleme ile seyreklikte olduğu gibi burada da öznitelik artıklığını önlemek için projeksiyon regülarizasyonu uygulanmıştır. Bu sistemde kullanılan ilerlemeli sinir ağı modelinde ağırlıkları seyrekleştirerek de makul performanslara ulaşılabildiği; öznitelik artıklığı için kullanılan bu regülarizasyonun, seyrekleştirmenin getirdiği performans düşüşünü büyük ölçüde önlediği ve yeni göreve özel öğrenilen ağırlık sayısını azalttığı görülmüştür.

Art arda gelen görevleri içeren sürekli öğrenme sisteminde görev sırasının etkisini anlamak için CIFAR-100 verisetinin çeşili alt kümeleri (insan, ev mobilyaları, orta boyutlu memeliler, araçlar 1) seçilerek farklı sıralarda ilerlemeli sinir ağlarıyla eğitim yapılmıştır. Bu deneylerde tam bağlantılı ve konvolüsyonel ilerlemeli sinir ağları kullanılmıştır. Burada, konvolüsyonel katmanların yanal bağlantılardan gelen girişleri yinelemeli olarak yığınlanmış, tam bağlantılı katmanı, geleneksel ilerlemeli sinir ağlarından farklı bir ileri fonksiyon kullanılmıştır. Her görev için özel öğrenilen filtrelerle elde edilen aktivasyon haritaları yinelemi olarak yığınlanmıştır. Kullanılan

ilerleme fonksiyonunda gereken parametre sayısı literatürden daha azdır. Öncül görevin şu anki görevle olan ilişkisinin performans üzerindeki etkisi incelenmiştir.

Yapılan analizlerin permutedMNIST veriseti ile çok katmanlı sinir ağları kullanılan kısmında elde edilen sonuçlarda, İlerlemeli Sinir Ağları'nda seyrek grup regülarizasyonu kullanmak, beklenildiği gibi ortalama performansı düşürmektedir. Bu düşüş örtüşmeyen ağırlık matrisleri öğrenmeye zorlanılarak büyük ölçüde engellenebilmektedir. Her görev için örtüşmeyen ağırlık matrislerini öğrenmek ve yanal bağlantıları kullanmak, göreve özel öğrenilen aktif bağlantı sayısını azaltmakta, ileri transfer sağlanmaktadır. Görev sırası ile ilişkin CIFAR-100 alt kümeleri ile yapılan deneylerde elde edilen sonuçlar ilerlemeli sinir ağlarının görsel sınıflandırma dizisinde yıkıcı unutkanlık sorunuyla karşılaşmadan ileri bilgi transferini yakaladığını ve bunun görev sırasıyla ilişkili olduğunu göstermiştir. Kullanılan seyreklik yöntemleri ile yüksek seyreklik oranları elde edilmiş, benzer zorluktaki görevlerde görev sırasıyla doğru orantılı olarak ihtiyaç olan nöron sayısı azalmıştır. Konvolüsyonel İlerlemeli Sinir Ağları ile yapılan deneylerde hem seyrek hem de seyrek olmayan durumlarda yanal bağlantıların ileri transferi sağladığı görülmüştür.

1. INTRODUCTION

Artificial Neural Networks are addressing major problems of computer vision, natural language processing and data science in the last decade with increasing computational power and amount of data. Despite its popularity and success, when presented with a sequence of tasks with only having access to current task' data, neural networks fail to preserve its success on anterior tasks. This problem is called catastrophic forgetting and one of the biggest obstacle on the way of artificial general intelligence.

Learning tasks sequentially is challenging for artificial neural networks because of their optimization based nature. The goal is rewritten by the new objective function for the incoming task. Therefore, performance on former tasks degrades over time. On the other hand, human brain builds knowledge on the top of experience by correlating different skills and factors. The goal of Continual Learning (CL) is to learn multiple tasks in a sequential fashion without suffering from catastrophic forgetting.

The goal of this thesis are investigating a special type of Neural Networks that exploits lateral connections in order to address Continual Learning problem on visual classification domain, employing sparse representations to increase efficiency of this dynamic approach and to investigate the effect of task priors. In this context, Multi Layer Perceptrons (MLP) and convolutional neural networks (CNN) are employed with various propagation rules. The models are sparsified using binary masking and sparse group LASSO regularization. In order to decrease the interference of weight spaces, projection loss and mean squared errors are employed. Lastly, on selected subtasks of CIFAR100, the effect of task orders on sequential settings are investigated.

In order to improve forward transfer, recursion and stacking feature maps are investigated. Results show that, exploiting PNNs on recursive fashion improved performance. Stacking feature maps on convolutional layers did not help with improving forward transfer. Results suggested that sparsification of weight matrices with projection loss leads to learn decreased number of task specific parameters on upcoming tasks for permutedMNIST dataset. Also, it is observed that training a

task with related priors outperforms training the same task with dissimilar priors. Performance of the new task depends on priors.

Masking experiments are carried out with FMNIST, KMNIST and MNIST datasets on a 3-task continual learning scenario. Task priors are investigated with using various subtasks of CIFAR-100.

The rest of the section explains problem formulation and related work.

1.1 Problem Formulation

In a conventional Continual Learning (CL) setting, a model learns to perform a collection of tasks $T = \{T_1, T_2, \ldots, T_t\}$ sequentially with corresponding datasets $D = \{(X_1, Y_1), (X_2, Y_2), ...(X_t, Y_t)\}$ where X_i and Y_i represent data and target, respectively. Here T_j represents j^{th} task and $j = 1, 2, 3, ..., t$ indicate training order. The model has no access to formerly seen datasets while training on the current task T_j. A typical CL scenario is illustrated in Figure 1.1. Here, each x_j is drawn from a different distribution P_j, belonging to task at order j, T_j. The purpose is continually learning these sequence of tasks without accessing (X, Y) where $i < j$. This setting is a supervised classification one with multi-head output, that is, for each task, model learns a different classification layer o_j. The purpose of continual learning is to be able to learn a sequence of tasks without suffering from catastrophic forgetting while improving forward and backward transfer across tasks.

1.2 Related Work

There has been numerous studies to address catastrophic forgetting in recent years [5–8]. Existing studies can be grouped into three main categories: Dynamic architectures, regularization based methods and replay based methods. Dynamic architectures aim to preserve performance by expanding the model, i.e. adding layers/neurons. For example in [9], authors proposed Dynamically Expandable Networks and prunes additional weights by group Lasso regularization. This is a complex method that requires additional hyperparameters and sub-algorithms. In Reinforced Continual Learning [10], model structure is updated with reinforcement learning and complexity-performance is balanced with the help of a reward function.

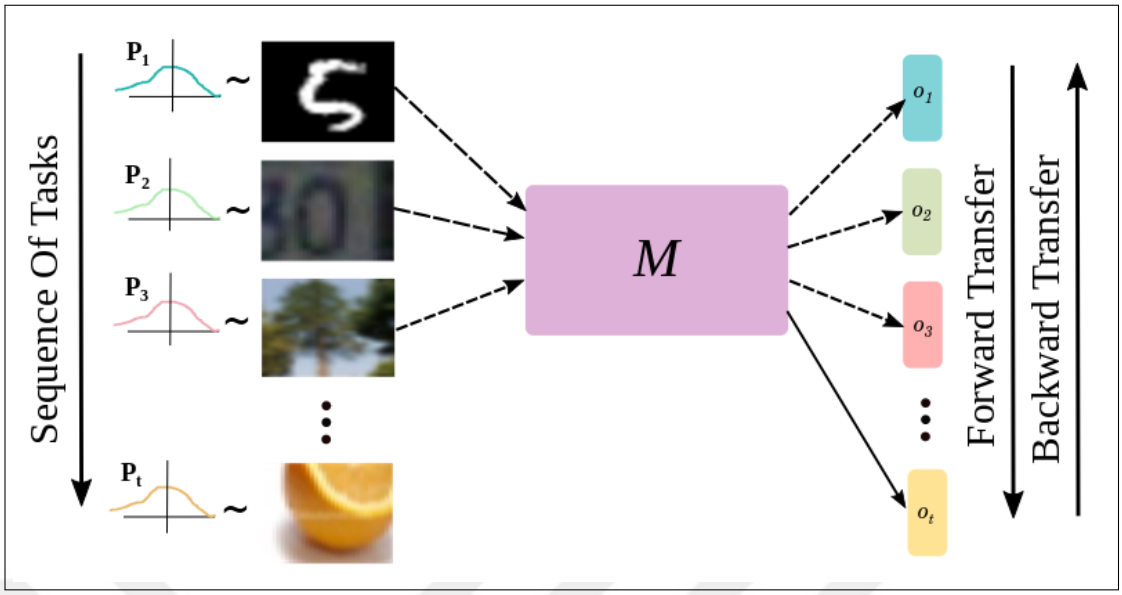


Figure 1.1 : An example of multi head continual learning scenario. For each input distribution P, model learns a different classifier o. Number of tasks is denoted with t. Purpose is to address "catastrophic forgetting" by improving forward and backward transfer.

A method that separates parameter estimation and finding optimal model structure proposed in [11]. This method employs various architecture search techniques to optimize the model structure. Another example of dynamic architectures is Progressive Neural Networks(PNN) [1]. PNNs initialize new set of learnable weights for each task and passes new input through all previously defined subnetworks. Purpose of PNNs is to relate new input with previuosly seen ones. One major disadvantage of PNNs is that the number of parameters requires grow quadratically with the number of tasks. [12] proposes a method that optimizes the neural structure and parameter learning/fine tuning component. This method searches for optimal architecture by separating architecture search and parameter estimation. Another study Bayesian Optimized Continual Learning with Attention Mechanism that dynamically expands the model capacity by Bayesian optimization and selects previuos knowledge by attention mechanism [13]. Further, Strannegård et al., proposes a model with expanding, generalizing, forgetting and back propagation rules insipred by biologic neuroplasticity [14]. These studies do not require additional memory however they tend to have complex implementations.

Replay-based methods evaluates model performance and address catastrophic forgetting by keeping a subset of previous datasets. There are two types of replay-based

methods: generative replay and episodic replay. Episodic replay methods rely on a sampling strategy and samples and stores a subset from each task. Gradient Episodic Memory (GEM) picks a subset from each task and uses this set to formalize a constrained optimization problem. Modifying cost function in this way prevents overfitting to pre-determined subset. Another variant of GEM is proposed in [15] modifies the cost function and achieves better performance with a new evaluation fashion. d'Autume et al. proposes an episodic memory method with local adaptation and sparse distributed experience replay to address catastrophic forgetting [16]. Reimer et al. fuses experience replay and maximization based meta learning to align gradients from different samples [17]. This approach decreases the possibility of interference of upcoming and previous gradients. Generative replay based methods captures underlying data distribution instead of sampling and generates samples. One example is Deep Generative Replay. This method captures distribution of current and previous tasks with a deep generative model and resulting model is used to preserve performance of all tasks.

Regularization based methods address catastrophic forgetting by constraining the change of model parameters. Elastic Weight Consolidation (EWC) calculates fisher information matrix after each task's training and punishes the change of important parameters proportional to Fisher Information Matrix (FIM) [5]. One disadvantage of this method is cost of calculating FIM and storing model parameters for each task on offline version. Synaptic Intelligence stores role of parameters on tasks and constrains the problem based on those stored values [6]. SI calculates this importance on each iteration while EWC makes a point estimate about it. There are also methods that employ Bayesian inference. Variational Continual Learning merges Monte Carlo approaches and variational inference [18]. Another study proposed a task based special attention mechanism to learn the current task without harming the performance of previous tasks [19].

One of the most common ways to regularize parameters of a complex neural network is to employ sparsity. Sparse representations and weight matrices are commonly used in compression such as prunning. Sparsity can be obtained through weight sparsification or representations sparsification. Employing sparse representations decreases the possibility of representation interference, therefore this concept is also employed in

the course of continual learning [20, 21]. Sparsity is also employed in biologically inspired neural networks since biological neurons spikes in a sparse fashion.

Organization of this thesis is as follows. Next section explains the methods utilized. Section 3 is dedicated to experimental results and section 4 is the conclusion.

2. SPARSE PROGRESSIVE NEURAL NETWORK

In this thesis, several types of Progressive Neural Networks(PNN) are used investigated for a sequence of visual classification tasks. This section is dedicated to Progressive Neural Networks and sparsification with binary masking and sparse group LASSO regularization.

2.1 Progressive Neural Networks

The long-standing goal of continual learning is to be able to transfer knowledge obtained through previous experiences. Human brain does not suffer from catastrophic forgetting because each task builds on top of previously learned ones by effectively integrating skills, learning new ones and fine-tuning. Inspired by this, this thesis investigates various types of progressive neural networks for Continual Learning, which adapt lateral connections to integrate prior knowledge. This study employs dynamic architectures similar to [1]. Purpose of such progressive models is to transfer knowledge, achieve faster convergence and improve performance of arriving task.

Initially, a PNN starts with a conventional neural network with desired number of layers and neurons that performs on a single task. Within the arrival of a new task new columns of task specific layers and weights are initialized. Previous weights of previous tasks are frozen in order to preserve knowledge. Therefore, this setting is immune to catastrophic forgetting. The purpose of lateral connections are to enable the transfer learning when possible.

In [1], the focus is reinforcement learning applications. In this study we investigated the performance of several type of PNNs on several visual classification scenarios. Later, PNNs are sparsified using binary masking and employing Sparse Group LASSO regularization. Lastly, the effect of task order on forward transfer is investigated.

Figure 2.1 illustrates a conventional PNN. Dashed arrows denote inactive connections while others denote active connections. Each column belongs to a task. Here, h refers

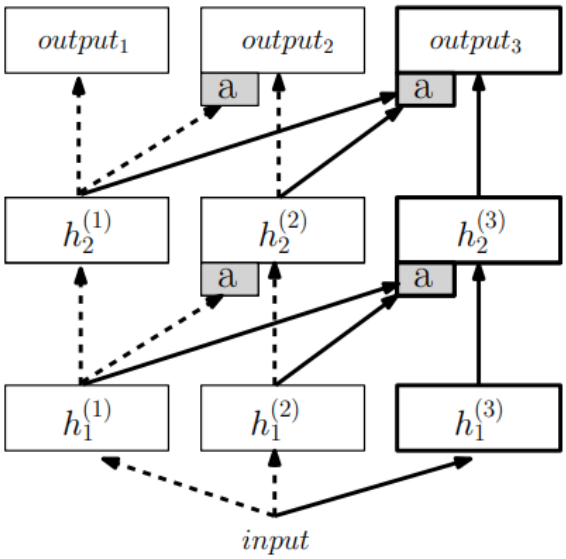


Figure 2.1 : Progressive Neural Network introduced in [1] with t = 3. For each task, different classification layer is used. Extracted from [1].

to output of a layer, hidden activations. Activation of layer i in a K task setting is calculated as follows:

$$h_k^t = f_{act}(W_k^t h_{k-1}^t + \sum_{j<t} U_k^{t:j} h_{k-1}^j) \tag{2.1}$$

In this equation, f_{act} refers to element-wise non linearity, U is weight matrix transforms lateral information from previous tasks, and W is parameters of current layer. Layer h_k^t has multiple inputs. U and W matrices represent $n_i \times n_j$ dimensional spaces.

In practive, every lateral previous connection h_{k-1}^j is concatenated to a single vector as $h_{i-1}^{<k}$. Then, a multi layer perceptron, called *adapters* is employed. In this case, output of the progressive network is calculated as:

$$h_k^t = \sigma(W_k^t h_{k-1}^t + U_k^{t:j} \sigma(V_k^{t:j} \alpha_k h_{k-1}^{<t})) \tag{2.2}$$

Where V is projection vector inside the MLP. Next section will explain propagation rules and structures of Progressive Neural Networks employed in this study.

2.2 Our Approach

In this section, the details on the progressive models utilized in this thesis are given.

2.2.1 MLP Progressive Neural Networks

This section explains the propagation rule and structure of PNNs employed in this study that consists of dense layers.

Figure 2.2 shows a PNN that consists of 2 dense layers. This structure is utilized on binary masking experiments. In this figure, input sampled from T_j is represented as x_j. Distributions of P_j's, corresponding to x_j's may vary for different values of $j = 1,...,t$ can be similar or dissimilar. This is a model trained for t tasks that includes t subnetworks, each having k layers. For each new task T_t a new weight set $\{W_1^t, W_2^t, ..., W_k^t\}$ is initialized and all W_i^j's where $j < t$, and are kept frozen making model immune to catastrophic forgetting naturally. Each task has its own task specific classification layer o^t and layer specific weighting parameters α_{jk}^t's for corresponding task t. After initialization of new weight set for current task, input is passed through all previously trained subnetworks and summed with α's. This is different from original PNN approach [1], where an additional adaptive layer is employed for the current layer. Moreover, another difference from the approach in [1] is, instead of directly taking nonlinear output of a layer, this model exploits weight spaces directly. Input of current layer is linearly combined with task specific α's. Omitting biases, output of k-th layer of the t-th task, h_k^t, is calculated as follows:

$$h_k^t = f_{act}(\sum_{j=0}^{t-1} \alpha_{jk}^t W_k^j x_{k-1}^j + W_k^j x_k^t) \tag{2.3}$$

where $t-1$ shows the number of utilized lateral connections and x_k^t shows the input of layer k of task t coming from the same subnetwork. $W_k^j \in R^{n_i \times n_j}$ is weight matrix of layer-k of task-j and the activation function, $f_{act}(.)$, is decided to be ReLU.

2.2.2 Recursive Progressive Neural Networks

In this section, Recursive Progressive Neural Networks (R-PNN), a special type of PNN is introduced. An R-PNN recursively computes the outputs of prior subnetworks. The last subnetwork of the current task does not take input from any previous subtasks except the last one. Processing input recursively increases non-linearity.

Figure 2.3 illustrates an R-PNN trained for three sequential tasks where number of layers of each initialized subnetwork is 2. Each blue box represent a hidden layer h,

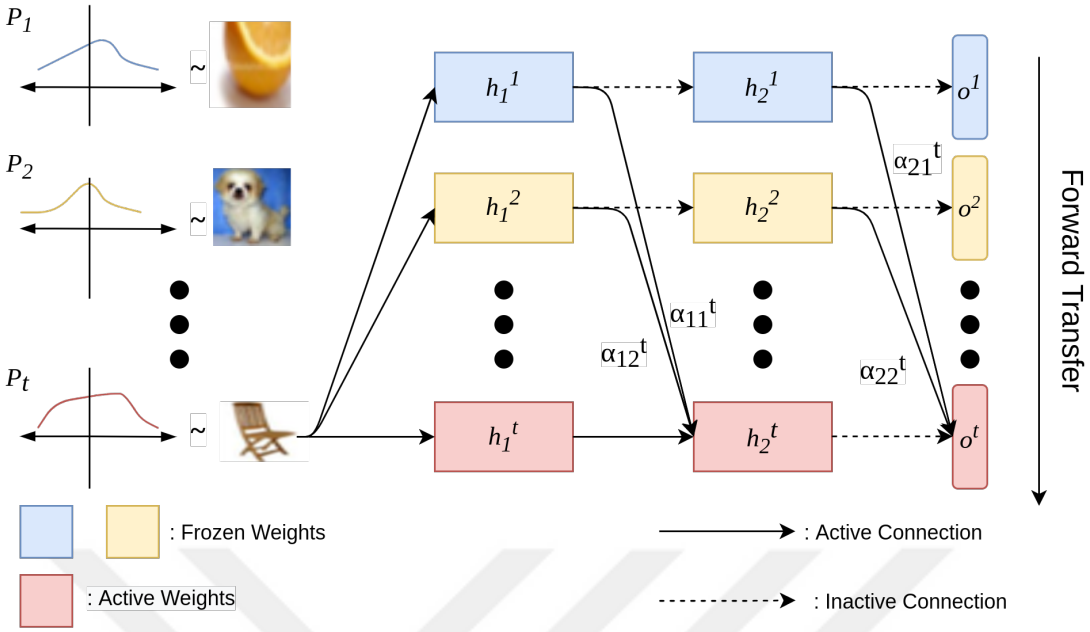


Figure 2.2 : A progressive Neural Network trained for t sequential tasks. For each task, different classification layer o is learned.

where yellow box refers to task specific classification layer, o. Recursive connections are shown with orange lines. Lernable weights for lateral connections are referred with α's. In this model, input of a hidden layer h consists of two parts. First branch comes from the previous layer of the same subnetwork, the second branch comes from the last prior task. When the input is passed to h_1^1, output of this layer is given to the next subnetwork only.

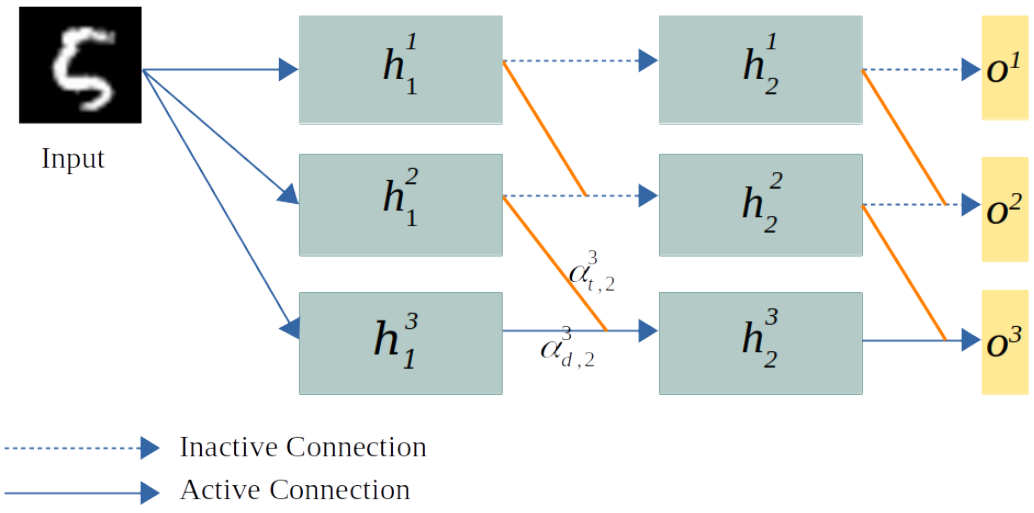


Figure 2.3 : Recursive Progressive Neural Networks where t = 3. Recursive connections are shown with orange lines.

In an R-PNN, output of a layer is calculated as:

$$h_k^t = f_{act}(\alpha_{d,k}^t W_k{}^t x_k^t + \alpha_{t,k}^t W_k^{t-1} x_k^{t-1}) \tag{2.4}$$

where t is current task and $\alpha_{d,k}^t$ and $\alpha_{t,k}^t$ are learnable parameters that weigh inputs from previous layer and previous task. f_{act} is point-wise non-linearity. Figure 2.4 illustrates the forward propagation rule of a layer.

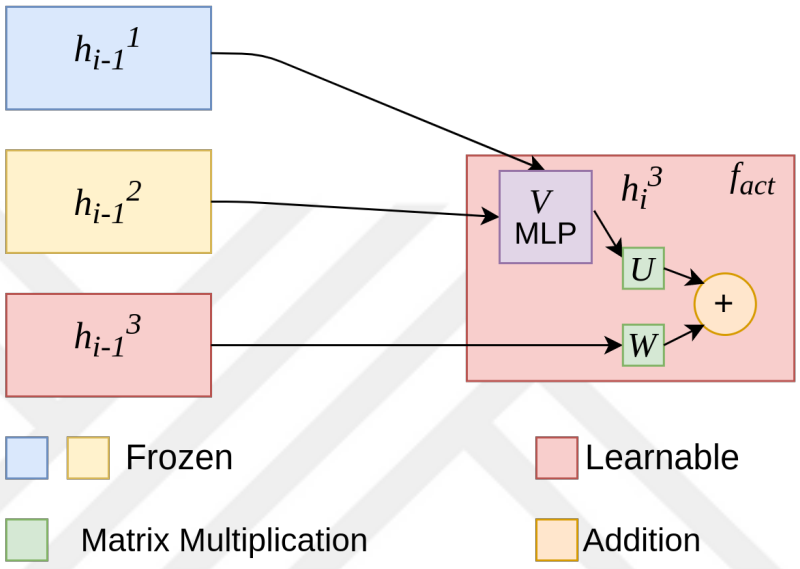


Figure 2.4 : Propagation rule of dense layers.

2.2.3 Stacking Feature Maps Progressively

Equation 2.3 exploits learnt weight spaces. On a convolutional layer, another approach, treating feature maps as a channel of the input that comes from previous layers in the current subnetwork is investigated. This is achieved by concatenating feature maps that comes from all the subnetworks. Figure 2.5 shows how feature maps are stacked progressively for task number $t = 3$. In contrast to prior work, before feeding to a non-linear transformation as shown in 2.5, lateral activations are summed after weighing, instead of concatenation. Therefore, number of parameters are decreased compared to the [1]. The propagation rule of fully connected networks becomes:

$$h_k^t = \sigma(W_k^t h_{k-1}^t + U_k^t \sigma(V_k^t \sum_{j=0}^{t-1} W_k^j x_{k-1}^j)) \tag{2.5}$$

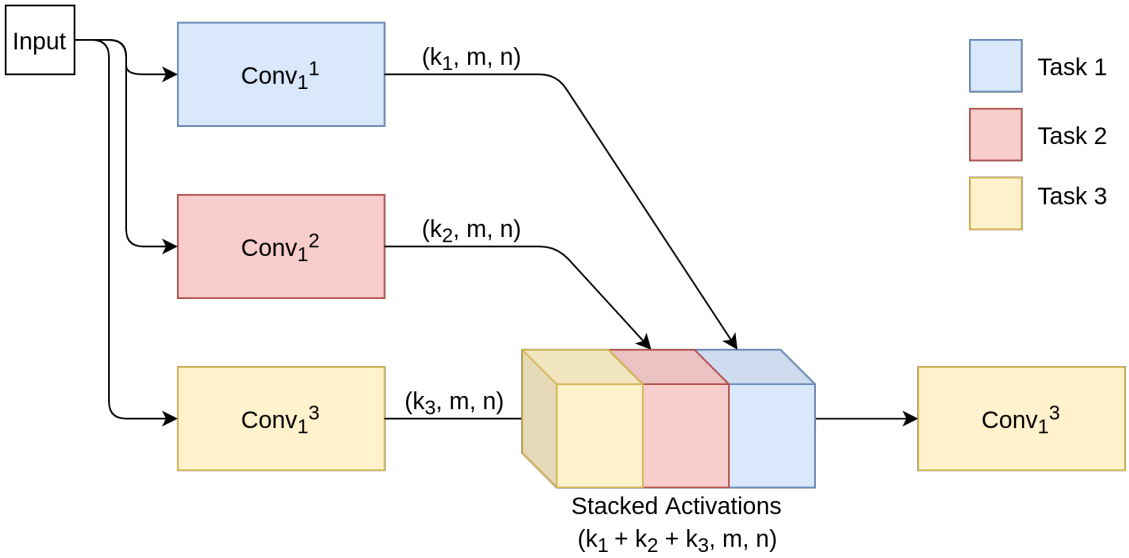


Figure 2.5 : Stacking feature maps of anterior subnetworks.

where i and k indicate layer and task indices, respectively. With this rule, previous layers and previous tasks are transformed onto different spaces. In this case, number of parameters decreases when compared to [1].

2.2.4 Feature Space Diversification

Visual tasks share low-level features and tasks interfere. Since task-specific weight matrices are initialized, features may overlap without regularization. However, the goal is to build new features on previously extracted ones. In order to decrease feature redundancy and to obtain non-overlapping activation maps on same column of the model, we utilized an additional loss term. Besides lateral connections and sparse activations, following term is added to loss function:

$$L_2 = -\sum_{l}^{L}\sum_{j}^{k-1} \|W_k^l - W_j^l\|^2 \tag{2.6}$$

Another way of obtaining sparsity is projecting current tasks' weight matrice onto previously learnt ones. This is shown in Equation 2.7.

$$L_3 = \sum_{k=0}^{K}\sum_{j=0}^{t-1} W_k^t W_k^j \tag{2.7}$$

2.3 Sparse Representations

Representations are activities of units in a network and define quality of a neural network. Useful representations are expressive representations and their quality can be

measured by comparing number of parameters that are learned and number of regions that representations can distinguish [22].

Figure 2.6 shows distributed and sparse representations with colormaps. Sparse representation allows only limited number/ratio of neurons/connections to be active at a time. The activation limitation on neurons/weights can be set manually or a sparsity ratio can be defined for optimization. Sparse representations have been used widely in biologically inspired neural networks to imitate encoded information representation of human brain and in continual learning problems to mitigate catastrophic interference [12, 23, 24].

There are several ways to obtain sparsity in a neural networks. In this thesis, we consider two approaches. First, we apply binary masks to activations and weights, then we apply Group LASSO regularization to the loss term.

2.3.1 Applying binary masks

Binary masks have a wide variety of applications within machine learning [25]. In this thesis, binary masks are used for obtaining sparse activations or sparse weight matrices. Shape of a binary mask is same as the weight matrix it is responsible for. During forward pass, weights or activations are ranked with the help of Softmax function. With normalized tensors, pre-determined k most strong connections or active neurons are picked. With indices of selected connections or neurons, binary masks m's are created. Activation maps and weights are masked to obtain sparse representations. In this way, only top-k connections are allowed to be active at a time. Similarly, in the case of sparse activations, only top-k neurons are allowed to propagate information. This procedure is applied both in training and testing time. Using binary masks are similar to employing Dropout [26]. However, dropout sets neurons to zero in a stochastic way while this procedure employs ranking.

Figure 2.7 depicts the employed architecture for three tasks. For each activation tensor, corresponding binary masks are created using softmax ranking and activations are sparsified. After masking each activation tensor, maps are weighted with learnable parameter α's. Each element of lateral connections are weighed with same α value. After that, maps are summed.

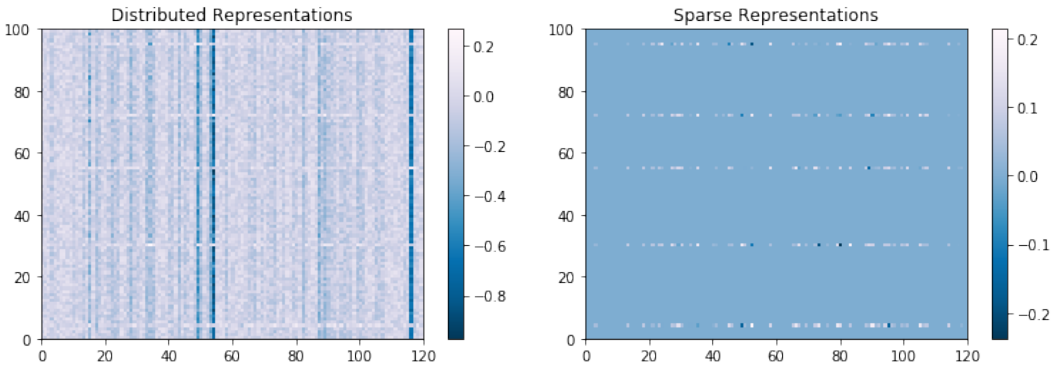


Figure 2.6 : Distributed (left) and Sparse (right) Representations.

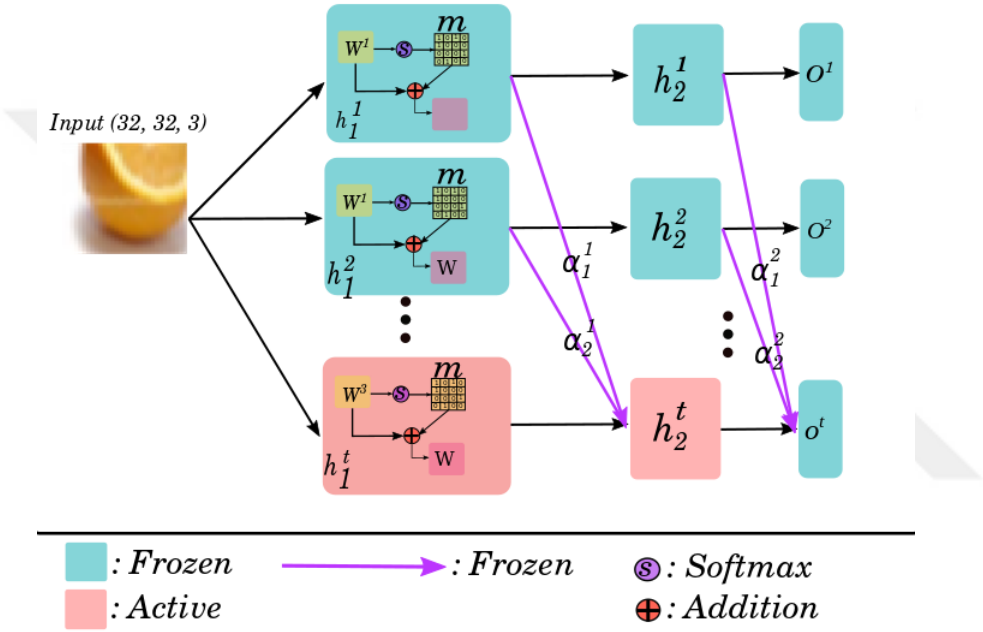


Figure 2.7 : Employed Continual Learning Architecture for three tasks with sparse activations. After calculating logit, learnable α parameters, activations are sparsified by using binary masks. Binary masks obtained by ranking weights are denotes as m.

In this case, output of layer k of task t is calculated as in 2.8. $W_j k^l$ refers to weight matrix of layer l of previous task j.

$$h_k^t = f_{act}(\sum_j^t \alpha_{jk}^t W_{jk}^t x_j k^t) \tag{2.8}$$

Here, t refers to the total number of tasks, which means architecture has K sub-networks. Each subnetwork has L pre-defined number of layers. W_i^l represent weight matrix of layer l of task i. For each new task i, new set of parameters W_i and $W_d i$ are initialized where W_i^l represent weight matrix of layer l of task i and $W_d i$ is decision weight. We calculate logits as in (2.3), here f_{act} was ReLu. If we are employing sparse activations, we mask z_i^l and obtain sparse matrices. If we sparsify

weights, then this step is done before calculating logits. After that we calculate output y and loss L depending on which term we use. When training for current task is done, we freeze and store all parameters W_i and W_di, hence they are not affected by the newly learned features in the forward pass. Hence, on each training procedure, only related parameters get update.

2.3.2 Employing Sparse Group LASSO regularization

There are two traditional weight regularization methods for preventing overfitting and obtaining better generalization in parametric machine learning models. One of them is using l_2 norm and its applied by adding the following term to loss function:

$$R_{l_2}(w) \triangleq (\|w\|_2^2) \tag{2.9}$$

l_2 regularization is also called "weight decay" since parameter magnituted are constrained and decreased through iterations. l_2 norm constitutes the rigde regression algoritm.

The second most popular approach to regularize neural network parameters is using l_1 norm. This approach stems from lasso algorithm and defined as:

$$R_{l_1}(w) \triangleq (|w|_1) \tag{2.10}$$

Both l_1 and l_2, *weight decay* regularizations are used to achieve generalization and can lead to sparsified neural network connections. However, employing l_1 and l_2 do not result in compact networks, since removing a neuron completely from a network requires all connections going out from that neuron to be zeroed out. A way for obtaining compact sparse networks is applying group LASSO which enables to safely remove a neuron. Therefore, *Group-level* sparsity is employed in this study to sparsify PNNs as applied in [2]. Purpose of this type of regularization is to eliminate all outgoing connections *group* from a neuron. There are three defined groups on group level sparsity:

- Input group G_{in}: An element of this group $g_i \in G_{in}$ represents set of outgoing connections of neuron i.
- Hidden group G_h: An element of this group represents all outgoing connections from a hidden neuron.

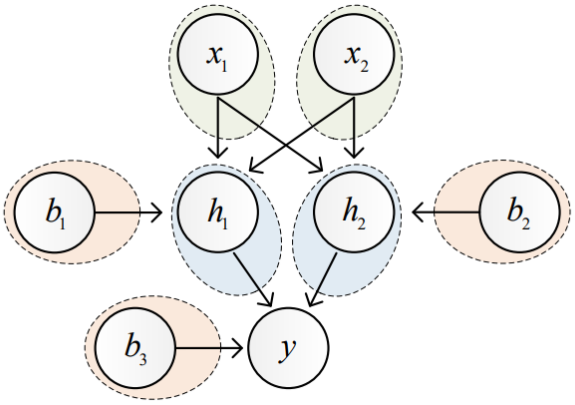


Figure 2.8 : Group representations in a network for group LASSO regularization. Grey, blue and red areas represent input, hidden and bias neuron groups, respectively. Here, the last node, y is output. Extracted from [2].

- Bias group G_b: An element of bias group is a scalar value being the bias of the corresponding neuron.

Figure 2.8 represents three different group types with different color coded nodes. Grey group represents the input group, red group is bias group and blue group is bias.

Defining all groups in a network as $G = G_{in} \cup G_h \cup G_b$, sparse group regularization, $L_{l_{2,1}}$, is shown in (2.11) as follows.

$$L_{l_{2,1}} \triangleq (\sum_{g \in G} \sqrt{|g|} \|g\|_2) \tag{2.11}$$

where $|g|$ is norm of a group and it ensures all groups gets weighted uniformly. Sparse group LASSO regularization, L_{SGL}, is defined as:

$$L_{SGL} \triangleq L_{l_{2,1}} + L_{p1} \tag{2.12}$$

Effect of this term is shown in Figure 2.9. In this figure, active and inactive connections of a small weight matrix is shown for three different losses. Grey boxes refers to inactive connections. The first row demonstrates the effect of LASSO regularization. Second row shows Group Lasso, while the last row shows the Sparse Group-Lasso. While all cases obtain sparsity, Group wise removal of neurons are not achieved by Lasso and the third row reaches the highest sparsity.

In addition to (2.12), we also put another constraint to loss function to motivate diversification of task specific features and to decrease feature redundancy. This

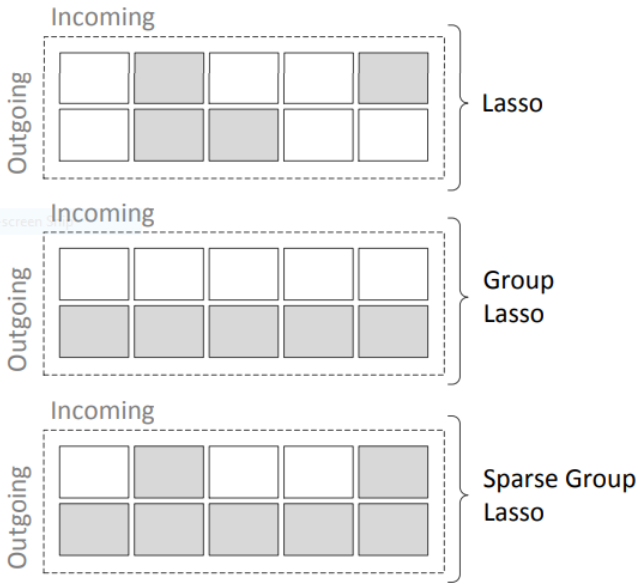


Figure 2.9 : Connection activities of a single weight matrix in three different regularizations. Gray represents the dead connections. White boxes refers to active connections. Extracted from [2].

projection term is defined as:

$$L_3 = \sum_{k=0}^{K} \sum_{j=0}^{t-1} W_k^t W_k^j \tag{2.13}$$

The purpose of this term is to improve feature efficiency and to decrease interference between tasks on a lateral setting. The goal is to learn non-overlapping weight matrices with the help of (2.13). Classification loss is cross entropy loss function, defined in (2.14):

$$E_1 = \sum_i y_i \log(p_i) \tag{2.14}$$

Where y_i and p_i are being target and prediction vectors for an input sample x_i. In this case, we try to minimize the compound loss term, E, as follows:

$$E = E_1 + \lambda_1 L_{SGL} + \lambda_2 L_3 \tag{2.15}$$

3. EXPERIMENTAL RESULTS

All experiments are implemented using PyTorch [27], an open source deep learning library, on Nvidia GeForce 2080Ti graphics processing unit. For all experiments accuracy metric is calculated as follows:

$$Accuracy = \frac{s_{true}}{s_{false} + s_{true}} \times 100 \tag{3.1}$$

Where s_{true} is number of true labeled samples where s_{false} is number of samples that have false labels.

On all reported results, sparsity ratios, S, on tables are calculated as in (3.2) where C_{act} and C_t are number of active connections and total connections, respectively.

$$S(\%) = (1 - \frac{C_{act}}{C_t}) \times 100\% \tag{3.2}$$

All experiments are carried out with rectified linear units. To assess effect of sparsity we did not employ dropout regularization. We employed ADAM optimizer which takes a different step for each term engaged in loss function. Details of experiments are given on Appendix A.

3.1 Sparse Progressive Neural Networks with Binary Masks

In this section, we consider MNIST [28], Fashion-MNIST (FMNIST) [3] and KMNIST [4] datasets in a 3-task continual learning setting. Fashion MNIST is an MNIST-like dataset that contains grayscaled images from 10 classes. KMNIST dataset contains japanese letters of 10 different types. Both FMNIST and KMNIST are drop-in replacements for MNIST dataset and share the same image size (28×28) and structure of training and testing splits. Details on datasets are given on Appendix A.

In this section, performance of several continual learning scenario that consists MNIST, FMNIST and KMNIST are evauated on various types of sparse PNNs. Sparsifications are achieved using binary masks as explained in section 2. For all sparsity scenarios we picked s to be 0.8 meaning that 20% of all neurons/connections

are active at a time. We averaged results of various priors for an order of task. To calculate MNIST dataset's second task performance, we averaged results of prior KMNIST and prior FMNIST.

In the first setting, each task-specific subnetwork is a multi layer perceptron with two fully connected layers. This architecture is shown in Figure 2.7. First, the setting is implemented with all fully active layers without employing sparsity and each lateral connection from a prior task weighted equally. The results are illustrated on table 3.1. Sub-indices of the first column indicates the order of the task and the reported results are averaged for tasks other than aforementioned one. Rows 1 and 2 show accuracies on datasets with corresponding task orders. The last two rows report results on sparsified PNN with sparsity ratio $s = 0.8$. In experiments, sparsification is achieved with activations, the top 0.2 of neurons are allowed to be active at a time. It is observed that, averaging auxiliary inputs harms the performance and model fails on transferring knowledge.

The second setting is similar to the first one, where coefficient α's are learnable parameters. All neurons and weights are fully activated. Each auxiliary input is weighed with corresponding α_j. Results are shown in Table 3.2. the last two rows reports the results where MSE regularization term applied to weights in order to achieve feature diversity with various learning rates, λ's. The last row results in 2.6 % of improvement in total.

Table 3.1 : Continual Learning Architecture performance without learnable α's. First and second rows are results with fully active neurons. Last two rows reports results with sparsity ratio s is 0.8, where top 20% of activations are fired. Lateral connections still fail forward transfer.

MLP	MNIST	FMNIST	KMNIST
T_1	95.5	84.2	82.1
T_2	94.5	84.2	83
T_1	95	84.2	84
T_2	94.5	82.5	81.5

In Table 3.2, performances of sparsified PNNs are reported with learnable lateral weighing parameters α's. When second rows of Table 3.1 and 3.2 are compared, there are minor improvements for each task. Overall, this table illustrates that, using learnable α's yields improvements on accuracies and and employing diversification

Table 3.2 : Continual Learning Architecture performance with learnable α's. First and second rows reports results with fully active neurons and connections. In last two rows, mean squared error is used to regularize weights to utilize diverse features. It is shown that using learnable α's improve accuracies and combining sparsity with regularization improves forward transfer.

MLP	MNIST	FMNIST	KMNIST
T_1	95.5	84.2	82.1
T_2	96.5	86	87.1
T_2, MSE, $lr = 0.6$	96.5	85	86
T_2, MSE, $lr = 0.1$	97.5	87.5	87.2

Table 3.3 : Performance of sparsified network with sparse activation where s is 0.8 with learnable parameters α's. Performance of this architecture on MNIST, FMNIST and KMNIST datasets with various regularization terms and learning rates are reported. Employing projection loss results in %1.4 accuracy improvement.

MLP	MNIST	FMNIST	KMNIST
T_1	95	85.2	81.9
T_2	96.5	87	88
T_2 , $E_1 + \lambda L_2$, $\lambda = 0.001$	97.5	85.9	88
T_2, $E_1 + \lambda L_2$, $\lambda = 0.1$	97.5	86.5	87.5
T_2, $E_1 + \lambda L_3$, $\lambda = 0.1$	97.5	87	88.4

regularization improves forward transfer. This shows that some prior tasks contributes more on forward transfer than others.

In order to understand the effect of sparsity on this architecture, various s values are investigated and other regularizations are discarded. Figure 3.1 and 3.2 show performance of PNN model without employing any additional regularizations on sparse PNN for task order 2 and 3, respectively. Sparsifying connections still produces fair results and performance gap between $s = 0.1$ and $s = 0.6$ increases when the order is three.

Accuracies of activation sparsification of PNNs are shown in Table 3.3. The first row shows the plain task accuracies of MNIST, FMNIST, and KMNIST with activation sparsification. Second row illustrates that employing lateral connections achieves forward transfer and prior tasks improve accuracies by %9.4 on three employed tasks. The fourth row shows that employing mean squared error regularization on weights decreases the performance of FMNIST dataset. Using projection loss neither decreases

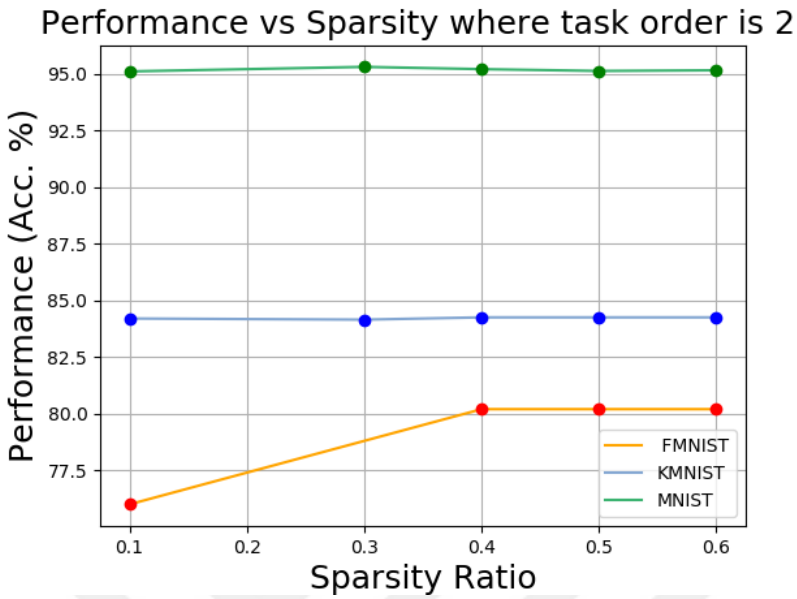


Figure 3.1 : Sparsity ratio versus accuracy curves for architecture in Figure 2.7 with sparse weights for the task orders 2.

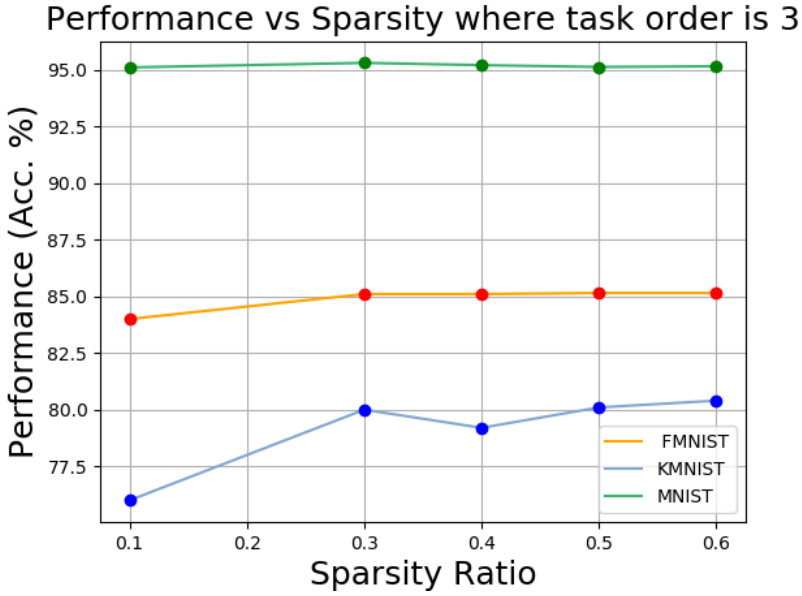


Figure 3.2 : Sparsity ratio versus accuracy curves for architecture in Figure 2.7 with sparse weights for the task orders 2. Loss is not combined with additional terms. Sparse weights produce fair results. Performance gap between 0.1 and 0.6 is increased when task order is 3.

or increases accuracy on FMNIST or KMNIST datasets and performance increase is only observed on MNIST as shown on the last row. However, on average, projection loss is superior to mean squared loss by %0.47 accuracy. On average, employing mean squared error do not improve T_2 accuracy. On the other hand, projection regularization results in 1.4 % accuracy improvement.

Table 3.4 : Sparse weights (where s is 0.8), with activation normalization and learnable parameters α's. Order of the task denoted as the subscript of T. Accuracies of this architecture trained with additional loss function, mean squared error and dot product are reported on this table with various learning rates.

MLP	MNIST	FMNIST	KMNIST
T_1	94	83.1	74
T_2	95	85	79
T_3	96	84.9	79
T_2, L2 = W1.W2	95.5	84.9	80.5
T_3, L2 = W1.W2	95.5	85	80.5
T_2, L2 = MSE	95.5	85.3	79

Another scenario is sparsifying the connections. In Table 3.4, connections are sparsified, in which only top-k percent of connections are active at a time. Sparsifying weights harms the performance more than sparsifying activations, hence it is more challenging to transfer knowledge forward. Best performance is obtained with Eq. 2.7 as shown in Line 5. As illustrated on the last row, employing mean squared error improves only the performance of MNIST.

Table 3.5 : Sparse Convolutional PNN where s is 0.8 . Both FMNIST and KMNIST task shows improvements for order three.

CNN	MNIST	FMNIST	KMNIST
T_1	95.9	78.8	79.3
T_2	97.9	83.7	87.6
T_3	98.1	86.1	92.5

Later, Convolutional PNN is investigated on the same sequential scenario. Employing Convolutional layers with addition operation did not contribute the results, therefore results are not reported. Instead, prior feature maps are stacked and recursive layers are employed as explained in section 2. Table 3.5, 3.6 and 3.7 reports the results for sparse convolutional PNN with s = 0.8. Employing lateral connections improves accuracy

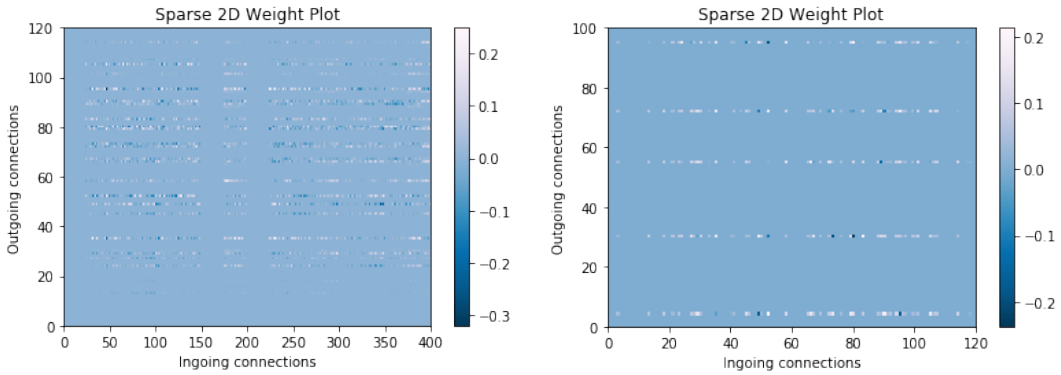


Figure 3.3 : Weight plots of two layers of a PNN sparsified with Sparse Group LASSO regularization.

by 15.2 and 25.7 % for task order 2 and 3, respectively. The biggest improvement is observed on KMNIST with 13.2 %.

Table 3.6 : Recursive PGN with 0.8 sparsity ratio. Performance on recursive case is slightly better when compared to conventional PGN, KMNIST's accuracy is improved by 1% .

CNN	MNIST	FMNIST	KMNIST
T_1	95.9	78.8	79.3
T_2	97.8	83.9	87.7
T_3	**98.1**	**87.2**	**92.4**

Next, recursive connections are evaluated. Table 3.6 reports the results of a sparse recursive PNN with convolutions. Results show that recursion improves the accuracy for FMNIST by 0.5 and 1.1 % compared to Table 3.5 on task order 2 and 3.

Lastly, stacking activations and recursion are combined and results are reported on Table 3.7. The third row show that stacking activations decreases the accuracy of the third task by 11.3 %. Forward transfer is 12.4 % on the order of 3.

Next section is dedicated to Sparse Group LASSO regularization in PNNs.

3.2 Sparse PNNs with Sparse Group Lasso

Table 3.7 : Recursive PGN with stacked activations .

CNN	MNIST	FMNIST	KMNIST
T_1	95.9	78.8	79.3
T_2	96.5	83	83
T_3	96.5	83.5	86.4

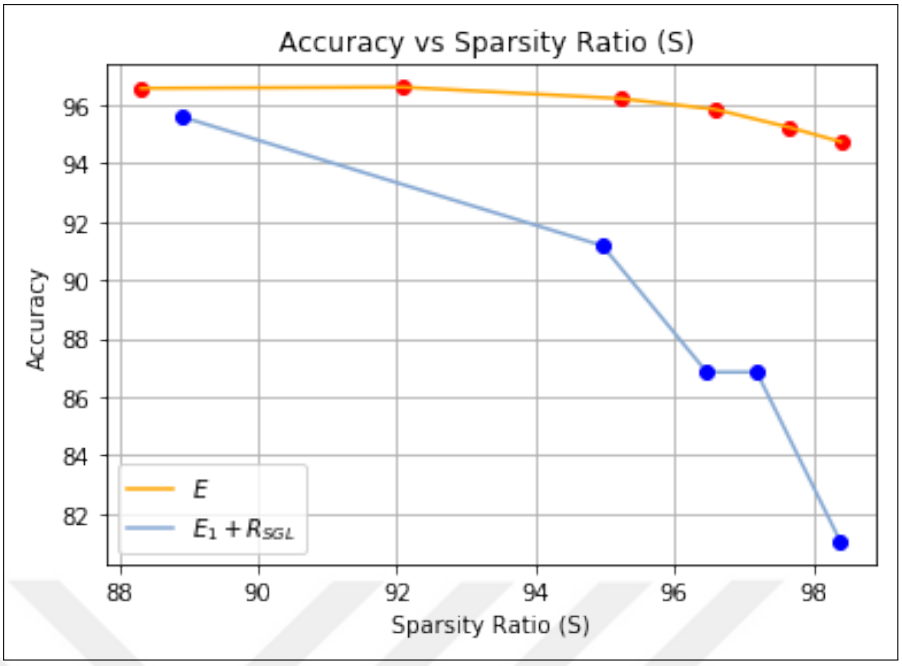


Figure 3.4 : Average accuracy versus sparsity ratios for $E_1 + R_{SGL}$(blue) and E(red). Projection loss is employed in addition to Sparse group LASSO and cross-entropy loss in red plot. Each dot represents averaged performance through 5 permutedMNIST tasks. Increasing level of sparsity causes performance drop (red curve). Employing projection terms prevents this decrease up to 12% (blue curve).

In this section the performance and efficiency of PNNs on two continual learning settings are evaluated. In the first setting, proposed approach is evaluated on 5 consecutive permuted MNIST tasks. PermutedMNIST dataset is obtained with permuting the pixels of MNIST dataset. In the second setting, lateral connections are evaluated on selected subtasks from CIFAR-100 with different task orders. Each subtask of CIFAR-100 includes one superclass, i.e., 5 fine labels. The purpose of CIFAR100 subtasks setting is to measure the effect of various task orders on forward transfer. Figure 3.3 illustrates two layers of a PNN trained with sparse group LASSO regularization.

3.2.1 Learning Non-overlapping Features

In the first scenario, the proposed approach is evaluated on permutedMNIST tasks where we set $t = 5$ and $k = 1$. Table 1 shows accuracy for each tasks trained sequentially. Here the subscript j of T_j indicates the order of the task. Average sparsity ratio for second and third columns is 95%. First row reports accuracy of fully active model only trained with cross-entropy loss. Second row is accuracy of sparse network

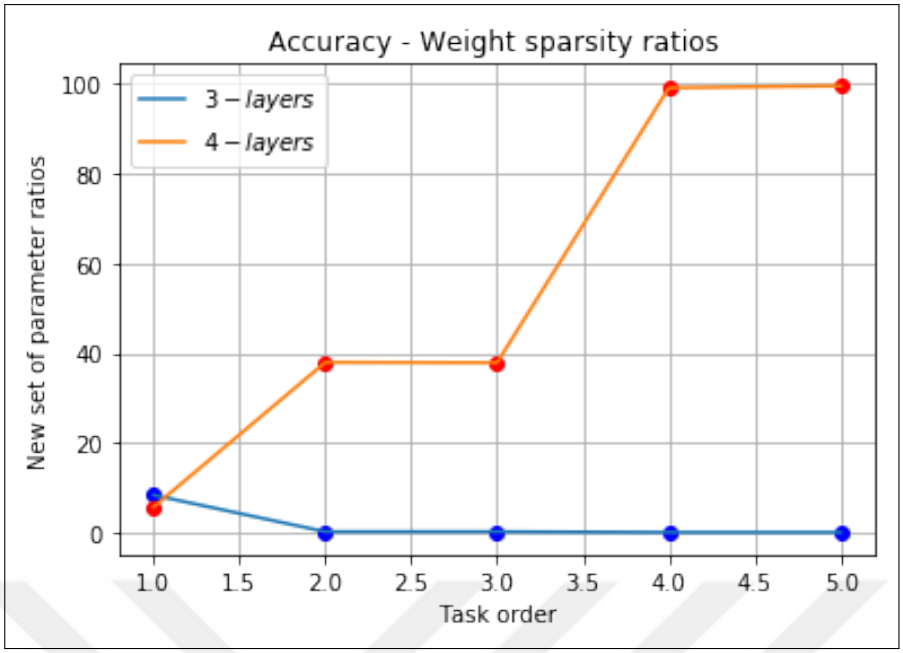


Figure 3.5 : Variation of sparsity ratio with changing task order for two different fully connected neural networks, namely, with three (blue) and four (red) layers.

Table 3.8 : Performances of Various Loss Functions for 5 tasks (%).

Task Name	T_1	T_2	T_3	T_4	T_5
E_1	97.74	97.71	97.76	97.64	97.88
$E_1 + R_{SGL}$	96.07	79.27	96.21	82.89	82.94
$E_1 + R_{L3}$	97.41	97.43	97.19	97.05	97.11
E	95.71	96.53	96.65	96.18	96.06

without projection loss. Performance of model trained with both sparse group LASSO and projection term is reported in the third row. Second row shows that employing only sparse group LASSO regularization leads to performance decrease on the model, especially on T_2 and T_4. Third row reports the results with both sparse group LASSO and projection term, combining (2.15) with sparse group LASSO prevents performance decrease and produces results close to original cross entropy. Results are close to non-sparse case, in the third row, for all tasks. When combined with cross entropy, projection loss do not positively contribute on task performances. The last row shows the effect of using projection loss with cross-entropy without employing sparsity.

Change of task performance with respect to sparsity ratio is plotted in figure 3.4 for models trained with (2.15) and $E + R_{SGL}$. Increasing sparsity decreases performance on both cases. However, for similar sparsity ratios, S, equation 2.15 outperforms equation

Table 3.9 : Sparsity Ratio of task specific parameters in proposed approach for different λ values. Required number of task specific parameters decreases in total. Non-overlapping features result in forward information transfer.

λ_2	T_1	T_2	T_3	T_4	T_5
5e-4	97.1	97	98	98.1	98.3
6e-4	98.3	97.9	98.5	98.7	98.7
3.5e-4	94.2	94.3	95.2	96	97

$E + R_{SGL}$. Equation 2.15 prevents major performance decrease caused by sparsification and results in similar performances with non-sparse case.

Lastly, Table 3.9 reports active connection ratio of model trained with equation 2.15. Number of task specific learned parameters decreases with increasing order of task. Employing projection regularization decreases task specific learned parameters. Each task requires less numbers of parameters than the previous one. Considering each task has similar complexity, required number of connections is decreased in total. Non-overlapping features result in forward information transfer for this scenario. Last row of Table 2 shows performance of projection term and cross entropy. One can observe that, employing projection term on non-sparse case do not improve forward transfer.

Figure 4 shows sparsity ratios, S, of two architectures with different layer numbers (3,4). Each architecture has similar number of neurons in total (See Appendix). Average accuracies through 5 tasks is 92% for 4 layers and 94% for 3 layers. Weighing each element of incoming activations by learnable vector α's instead of a scalar did not change sparsity ratios, S, and accuracies noticeably. In this scenario, we found that with increasing number of tasks, t optimization and hyperparameter tuning becomes difficult due to increasing number of lateral connections.

3.2.2 Measuring Effect of Task Order

Second scenario is evaluation of consecutive CIFAR-100 subtasks. In this case weighting parameters of lateral connections are vectorized, meaning that, each neuron of a lateral connection has a weighting parameter. The purpose in this section is to investigate the effect of prior task's similarity to current task on the magnitude of forward transfer. For this reason we picked several superclasses from CIFAR-100

Table 3.10 : Selected superclasses and corresponding classes from CIFAR-100 dataset. Task reference is shown in paranthesis.

Superclass	Classes
medium-sized mammals (1)	fox, porcupine, possum raccoon, skunk
small mammals (2)	hamster, mouse, rabbit shrew, squirrel
people (3)	baby, boy, girl man, woman
household furniture (4)	bed, chair, couch, table, wardrobe
vehicles 1 (5)	bicycle, bus, motorcycle pickup truck, train
vehicles 2 (6)	lawn-mower, rocket streetcar, tank, tractor
Household Electricity (7)	clock, computer keyboard lamp, telephone, television
Trees (8)	maple, oak, palm pine, willow
Fruit and Vegetables (9)	bicycle, bus, motorcycle pickup truck, train
Flowers (10)	orchids, poppies roses, sunflowers, tulips
Fish (11)	aquarium fish, flatfish, ray shark, trout
Aquatic Mammals (12)	beaver, dolphin otter, seal, whale

dataset and trained the model with different task orders. Table 3.10 shows selected superclasses and corresponding fine labels. Classes are referred as scalars as shown in parenthesis in Table 3.10. There are two groups of training set for CIFAR-100 on experiments. First one is employing directly E_1 without any regularization. The aim is to assess the performance of current task with various prior tasks on implemented PNN. In Table 3.11, comparison of Training 1 and 2 demonstrates that learning "Vehicle 1" (Task 5) after "Vehicle 2" (task 6) do not result in any performance increase. However, comparison of Training 1 and 3 shows, learning "Vehicle 1" after "People" (Task 3) result in performance decrease. This shows, task order does matter when employing lateral connections for continual learning. Training with a dissimilar prior harms the performance. Likewise, both performance of "Medium-sized Mammals" (Task 1) and "People" varies based on prior tasks.

Table 3.11 : Accuracy (%) Results for Various Task Orders for Subtasks of CIFAR-100 for Progressive MLP

Order	T_1	T_2	T_3	T_4
Class (1)	6	5	3	1
Acc. (%) (1)	66.4	50	29.4	47
Class (2)	5	6	3	1
Acc. (%) (2)	50	66	32.4	45.6
Class (3)	3	5	6	1
Acc. (%) (3)	29.2	47	62.8	47.4

On second case the goal is to see effect of task order in sparsified PNNs. We did not report results of employing projection term here because any performance improvement was not observed. Performance and sparsity ratios, S, of 3 trainings with various task orders are shown on Table 3.14. It can be seen that Task 2 i.e., "Small mammals" has 10% performance boost when trained after "Medium-sized Mammals" comparing to its first task performance in Training 2. However, when trained after "People", performance increase is 2%. Training with a similar prior boosts performance more. For all trainings "Household Furniture" (Task 4) is left to the last task and performance of all trainings are similar.

Table 3.12 : Accuracy (%) and Sparsity Results for Various Task Orders for Subtasks of CIFAR-100 for Progressive MLP

Order	T_1	T_2	T_3	T_4
Class (Training 1)	1	2	3	4
Acc. (%) (Training 1)	38.4	40.6	26	33
S (%) (Training 1)	99.11	99.31	98.86	98.87
Class (Training 2)	2	1	3	4
Acc. (%) (Training 2)	30.8	32.8	25.4	34.8
S (%) (Training 2)	99.09	99.31	98.86	98.82
Class (Training 3)	3	1	2	4
Acc. (%) (Training 3)	28	33.6	25	33
S (%) (Training 3)	99.43	99.32	98.92	98.91

After these experiment settings, task relation on stacked convolutional progressive neural network is investigated. Table 3.13 reports the results for first and second task performances of 4 subtasks of CIFAR100 and corresponding sparsity ratios. First set of results of the table reports the performance of non-sparse model. Sparsification results in performance decrease on all cases except Medium mammals subtask. Sparse regularization improves performance by 4%. Lateral connections of convolutional

Table 3.13 : Accuracy (%) and Sparsity Results for Various Task Orders for Subtasks of CIFAR-100 for Convolutional Progressive Neural Network

	People	Medium Mammals	Household F.	Vehicles 1
1	36	63.6	60	67
2	40	64	67	65
1	34.2	67.6	54	64
2	34.6	64	61.4	62
S (%)	81.79	75.85	74.37	71.03

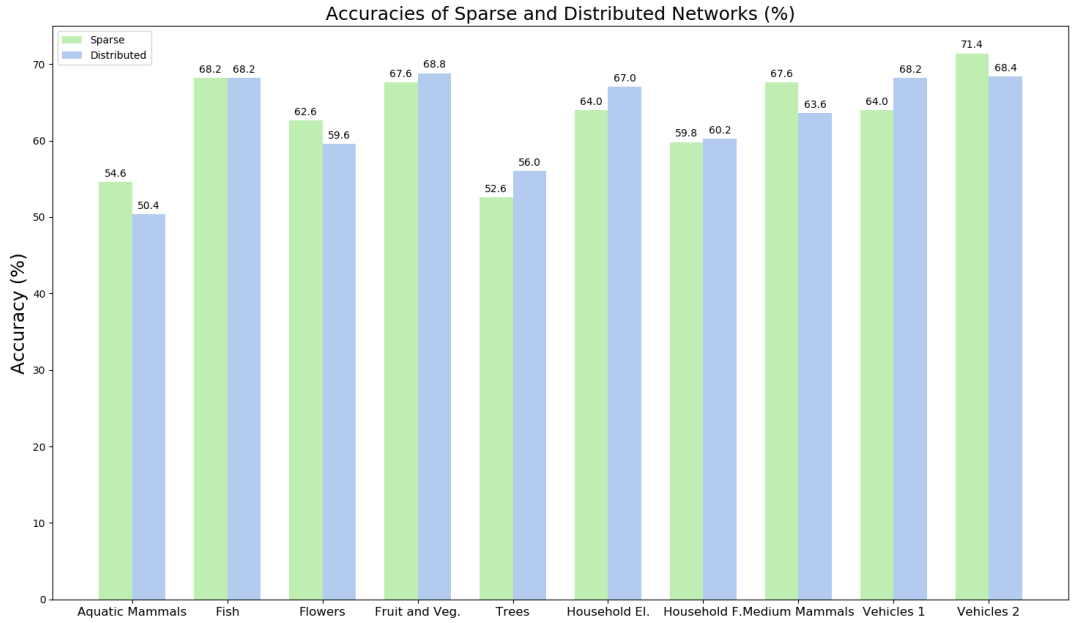


Figure 3.6 : Performances of Sparse and Distributed Networks for 10 different tasks.

PNN outperforms first task results. Lateral connection in Convolutional PNN with group level sparsity improves the performance with 2.4% on average. For all experiments, second tasks' convergence were faster when compared to task order 1.

Lastly, PNNs with stacked feature maps are used to perform on 10 sequential CIFAR100 subtasks. Figure 3.6 illustrates baseline performances of tasks. Average accuracy for 10 tasks is 63.04%. When sparsity is involved average accuracy is 63.24%. Involving sparsity had a minor improvement on performance.

Lastly, Table 3.14 shows performance and sparsity ratios of two training procedure, Training 1 and Training 2. Corresponding subtasks are given in table. Two training procedures includes same subtasks in a reversed order. The average accuracy of the first and second training sets are 65.06% and 63.56% ,respectively. The number of total parameters for the subnetworks are given on Appendix. Results show that, number of

required parameters did not constantly decreased with the increasing number of tasks. Still, lateral connections improved the performance by 1.92% and 0.42%, respectively.

Table 3.14 : Accuracy and Sparsity Results for Various Task Orders for Subtasks of CIFAR-100 Sparse PNN with Stacked feature maps. Training 2 follows reversed order of Training 1 tasks.

Order	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
Subtask	12	11	10	9	8	7	4	1	5	6
Acc. (%)	55.2	69.6	62	69	53.	65.6	63.2	69.2	66.6	77.2
S (%)	80.29	95.29	96.6	95.05	96.74	94.704	94.09	95.87	97.18	97.05

8pt

Table 3.15 : Accuracy and Sparsity Results for Various Task Orders for Subtasks of CIFAR-100 Sparse PNN with Stacked feature maps. Training 2 follows reversed order of Training 1 tasks.

Order	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
Subtask	6	5	1	4	7	8	9	10	11	12
Acc. (%)	71.4	65.6	65.6	58.6	63.8	53	70.6	62.4	68.6	56
S (%)	75.75	82.17	92.43	92.37	93.13	94.59	94.22	92.53	91.8	92.76

4. CONCLUSION

In this study, several types of Progressive Neural Networks are sparsified and investigated to perform on a series of visual classification tasks. In the analysis, MNIST, FMNIST, KMNIST, PermutedMNIST and CIFAR-100 datasets are used with various types of progressive neural networks. The effect of sparse representations are investigated to address the quadratically growing number of parameters of conventional progressive networks. Weights and activation maps are sparsified with binary masking and sparse group LASSO regularization. Projection and mean squared regularizations are employed to encourage the feature diversification.

On the experiments, the contributions of lateral connections on visual classification domain is verified. Recursive PNNs outperforms conventional lateral connections. Treating the prior information as a channel of the input did not yield better results.

It is verified that employing group level sparsity on Progressive Neural Networks causes performance decrease as expected. With the experiments on permutedMNIST, it is seen that, the amount of performance decrease can be largely addressed by learning non-overlapping weights of tasks. Minimizing projection between weight matrices and employing lateral connections decreased the number of task specific parameters.

Lastly, the effect of various task priors are analyzed using subtasks of CIFAR100 dataset. Results show that, correlation of current task with priors affects the performance of current task. Similar priors increase the forward transfer of lateral connections. It is seen that, over parameterization of the prior work [1] did not enhance the results on convolutional progressive neural network visual classification setting with CIFAR-100 subtasks. Also, overfitting of the prior task causes high amount of performance decrease on subsequent tasks. Therefore prior tasks needs to be carefully trained. Again, employing lateral stacked activation maps and treating lateral features as input channels did not yield satisfying results.

Progressive Neural Networks are successful on forward transfer in visual classification domains. However, the number of parameters increases with growing number of tasks. The purpose of this thesis was to analyze Progressive Networks in visual domains, to see whether its possible to decrease number of required task specific parameters as the number of tasks increases with engaging sparsity and to analyze the effect of task order. It is observed that, depending on the task priors, sparsity ratios can increase or decrease through tasks. Task order affects the forward transfer.

Detailed analysis of recursion, task orders and scalability are left for future work.

REFERENCES

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

[2] **Scardapane, S.**, **Comminiello, D.**, **Hussain, A. and Uncini, A.** (2017). Group sparse regularization for deep neural networks, *Neurocomputing*, *241*, 81–89.

[3] **Xiao, H.**, **Rasul, K. and Vollgraf, R.** (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, *arXiv preprint arXiv:1708.07747*.

[4] **Clanuwat, T.**, **Bober-Irizar, M.**, **Kitamoto, A.**, **Lamb, A.**, **Yamamoto, K. and Ha, D.** (2018). Deep Learning for Classical Japanese Literature, *arXiv preprint arXiv:1812.01718*.

[5] **Kirkpatrick, J.**, **Pascanu, R.**, **Rabinowitz, N.**, **Veness, J.**, **Desjardins, G.**, **Rusu, A.A.**, **Milan, K.**, **Quan, J.**, **Ramalho, T.**, **Grabska-Barwinska, A.** *et al.* (2017). Overcoming catastrophic forgetting in neural networks, *Proceedings of the national academy of sciences*, *114*(13), 3521–3526.

[6] **Zenke, F.**, **Poole, B. and Ganguli, S.** (2017). Continual learning through synaptic intelligence, *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3987–3995.

[7] **Ritter, H.**, **Botev, A. and Barber, D.** (2018). Online structured laplace approximations for overcoming catastrophic forgetting, *Advances in Neural Information Processing Systems*, 3738–3748.

[8] **Lopez-Paz, D. and Ranzato, M.** (2017). Gradient episodic memory for continual learning, *Advances in Neural Information Processing Systems*, 6467–6476.

[9] **Yoon, J.**, **Yang, E.**, **Lee, J. and Hwang, S.J.** (2017). Lifelong learning with dynamically expandable networks, *arXiv preprint arXiv:1708.01547*.

[10] **Xu, J. and Zhu, Z.** (2018). Reinforced continual learning, *Advances in Neural Information Processing Systems*, 899–908.

[11] **Li, X.**, **Zhou, Y.**, **Wu, T.**, **Socher, R. and Xiong, C.** (2019). Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting, *arXiv preprint arXiv:1904.00310*.

[12] **Han, S.**, **Pool, J.**, **Tran, J. and Dally, W.** (2015). Learning both weights and connections for efficient neural network, *Advances in neural information processing systems*, 1135–1143.

[13] **Xu, J.**, **Ma, J. and Zhu, Z.** (2019). Bayesian Optimized Continual Learning with Attention Mechanism, *arXiv preprint arXiv:1905.03980*.

[14] **Strannegård, C.**, **Carlström, H.**, **Engsner, N.**, **Mäkeläinen, F.**, **Seholm, F.S. and Chehreghani, M.H.** (2019). Lifelong Learning Starting From Zero, *arXiv preprint arXiv:1906.09852*.

[15] **Chaudhry, A.**, **Ranzato, M.**, **Rohrbach, M. and Elhoseiny, M.** (2018). Efficient lifelong learning with a-gem, *arXiv preprint arXiv:1812.00420*.

[16] **d'Autume, C.d.M.**, **Ruder, S.**, **Kong, L. and Yogatama, D.** (2019). Episodic Memory in Lifelong Language Learning, *arXiv preprint arXiv:1906.01076*.

[17] **Riemer, M.**, **Cases, I.**, **Ajemian, R.**, **Liu, M.**, **Rish, I.**, **Tu, Y. and Tesauro, G.** (2018). Learning to learn without forgetting by maximizing transfer and minimizing interference, *arXiv preprint arXiv:1810.11910*.

[18] **Nguyen, C.V.**, **Li, Y.**, **Bui, T.D. and Turner, R.E.** (2017). Variational continual learning, *arXiv preprint arXiv:1710.10628*.

[19] **Serrà, J.**, **Suris, D.**, **Miron, M. and Karatzoglou, A.** (2018). Overcoming catastrophic forgetting with hard attention to the task, *arXiv preprint arXiv:1801.01423*.

[20] **Sangwon Jung1, Hongjoon Ahn2, S.C. and Moon1, T.** (2020). Adaptive Group Sparse Regularization for Continual Learning, *arXiv preprint arXiv:2003.13726*.

[21] **Aljundi, R.**, **Rohrbach, M. and Tuytelaars, T.** (2018). Selfless sequential learning, *arXiv preprint arXiv:1806.05421*.

[22] **Bengio, Y.**, **Courville, A. and Vincent, P.** (2013). Representation learning: A review and new perspectives, *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

[23] **Narang, S.**, **Elsen, E.**, **Diamos, G. and Sengupta, S.** (2017). Exploring sparsity in recurrent neural networks, *arXiv preprint arXiv:1704.05119*.

[24] **Javed, K. and White, M.** (2019). Meta-learning representations for continual learning, *Advances in Neural Information Processing Systems*, 1818–1828.

[25] **Medhat, F.**, **Chesmore, D. and Robinson, J.** (2017). Masked Conditional Neural Networks for Environmental Sound Classification, *Artificial Intelligence XXXIV*, Springer International Publishing, Cham, pp.21–33.

[26] **Goodfellow, I.**, **Bengio, Y. and Courville, A.** (2016). *Deep learning*, MIT press.

[27] **Paszke, A.**, **Gross, S.**, **Chintala, S.**, **Chanan, G.**, **Yang, E.**, **DeVito, Z.**, **Lin, Z.**, **Desmaison, A.**, **Antiga, L. and Lerer, A.** (2017). Automatic differentiation in PyTorch.

[28] **LeCun, Y.** (1998). *The MNIST database of handwritten digits.*, http://yann. lecun. com/exdb/mnist/.

APPENDICES

APPENDIX A.1 : Datasets

APPENDIX A.2 : Experimental Details

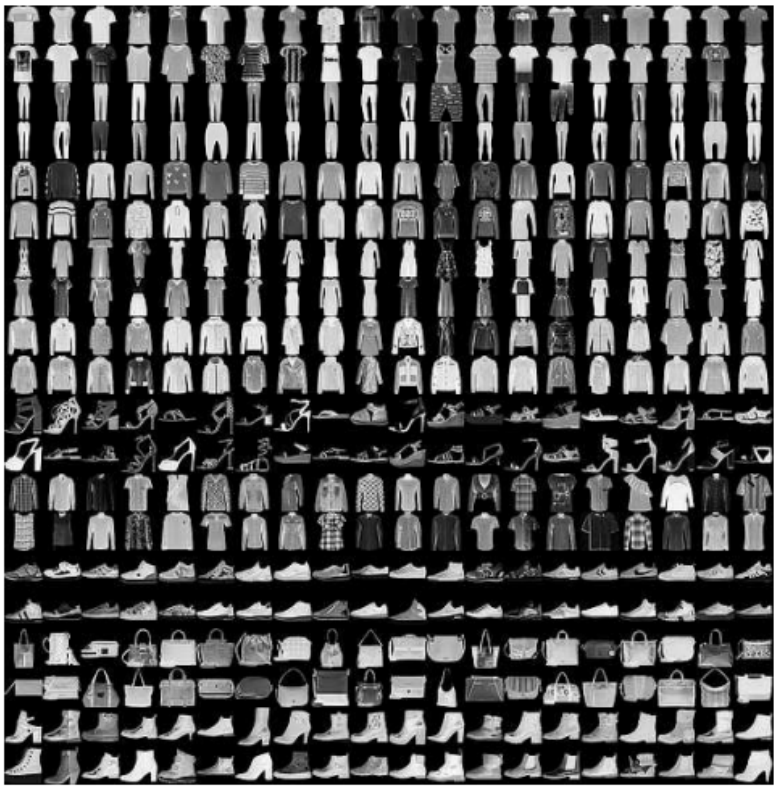
Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Figure A.1 : Details on Fashion MNIST Dataset. Extracted from [3].

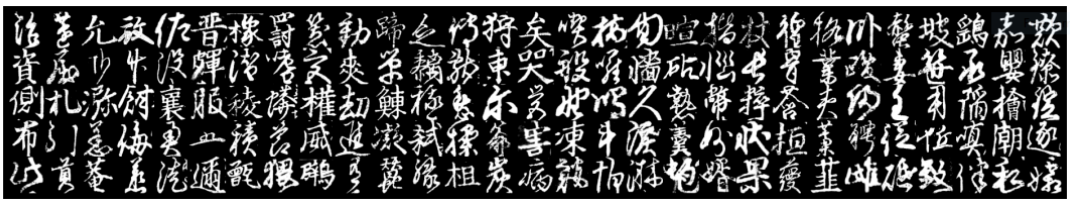


Figure A.2 : Details on Fashion MNIST Dataset. Extracted from [4].

APPENDIX A.1

Figure A.1 shows labels, class description and corresponding examples of Fashion-MNIST Dataset. FMNIST dataset consists of 60,000 training and 10,000 test images similar to MNIST.

Figure A.2 shows examples from KMNIST dataset. This dataset contains Japanese letters and considered to be more challenging than MNIST Dataset.

APPENDIX A.2

Experiments for Sparse Group LASSO regularizations and CIFAR100 subtasks are repeated for 3 seeds.

Multi Layer Perceptrons that employed in binary masking consists of 2 FC layers with 128 neurons on each layer.

CNN subnetworks for all experiments consists of 2 convolutional layers and 2 fully connected layers. Each convolutional layer has 16 filters with 3×3 kernel size. Stride size is set to 1. The first and the second FC layers consists of 150 and 10 neurons, respectively.

Multi Layer Perceptron that used for CIFAR100 subtasks experiments consists of 4 FC layers. Number of neurons are 350, 250, 150 and 100.

Hyperparameters for projection loss and Sparse Group LASSO are sampled from categorical disribution.

Parameters are truncated in order to achieve sparsity and convergence with Sparse Group LASSO regularization. Without applying truncated gradients, magnitudes oscillate between small positive and negative values.

On binary masking experiments, weight/activation magnitudes are ranked with softmax funtion.

CURRICULUM VITAE

Name Surname: Esra Ergün

Place and Date of Birth: Istanbul/Turkey 1994

E-Mail: ergunesr@itu.edu.tr

EDUCATION:

- **B.Sc.:** 2017, Istanbul Technical University, Electrical and Electronics Faculty, Electronics and Communication Engineering Department

PROFESSIONAL EXPERIENCE AND REWARDS:

- Research Assistant at Istanbul Technical University (2019-...)

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- E. Ergün, B.U Töreyin, " Continual Learning with Sparse Progressive Neural Networks", SIU, Gaziantep, Turkey, 2020.