A Compilation of Annotated Bibliographies

September 29, 2016

Sazia Mahfuz


**Abstract:** This paper extends prior work on knowledge consolidation and the stability-plasticity problem within the context of a Lifelong Machine Learning (LML) system. A context-sensitive multiple task learning (csMTL) neural network is used as a consolidated domain knowledge store. Prior work has demonstrated that a csMTL network, in combination with task rehearsal, can retain previous task knowledge when consolidating a sequence of up to ten tasks from a domain. However subsequent experimentation has shown that the method suffers from scaling problems as the learning sequence increases resulting in the loss of prior task accuracy and a growing computational cost for rehearsing prior tasks using larger training sets. A solution to these two problems is presented that uses a sweep method of rehearsal that requires only a small number of rehearsal examples (as few as one) for each prior task per training iteration in order to maintain prior task accuracy.

**Annotation:**

**Lifelong Machine Learning**  Lifelong Machine Learning (LML) considers "persistent and cumulative nature of learning" (Thrun 1996). LML systems should have the mechanism to retain old knowledge as well as acquire new knowledge. Integrating new knowledge into the repository of old knowledge is known as consolidation.

**Stability-plasticity Problem**  Stability refers to the issue that the learning mechanism can retain prior knowledge efficiently.
Plasticity refers to the issue that the learning mechanism can accommodate new knowledge effectively. A stable and plastic learning mechanism can rehearse examples of prior knowledge to maintain stability while slowly changing to accommodate new knowledge.

**Conditions to ensure stability-plasticity in MTL**  
Four conditions have been identified to ensure a stable and plastic learning mechanism.

(a) Task Rehearsal: Rehearsing synthesized virtual examples of prior tasks while simultaneously learning examples of new tasks.

(b) Internal Representation: Sufficient number of hidden nodes to accommodate the learning of all the tasks in the domain.

(c) Learning Rate: a small learning rate must be used to ensure slow integration of the new task into existing representation while maintaining the functional accuracy of prior tasks.

(d) Handling overfitting: Creating a validation set to check for minimum error can avoid the problem of overfitting.

**Limitation to MTL**

(a) Multiple inputs for each task: Having multiple inputs per task becomes challenging to maintain accuracy for related tasks.

(b) A separate output for each task: A separate output for each task increases complexity like determining the association between the task output and the set of examples.

**Approach used in the paper for task rehearsal**  
For task rehearsal in a csMTL network, the following steps have been taken to achieve better performance and decrease computational cost.

(a) Increase of hidden layers: The number of hidden layers is increased to represent the hierarchy of features in the multiple tasks.

(b) Weighting of training and validation set: The training and validation set are equally weighted for each task.

(c) Generating virtual examples: More effective virtual examples are generated from the knowledge of probability distribution over the input.

(d) Sweep Task Rehearsal: A sweep method of task rehearsal is used where randomly selected virtual examples are used
for rehearsing the prior task rather than using a large set of virtual examples for rehearsing the prior tasks.

**Advantages of using Sweep Task Rehearsal**

(a) Reduced Computational cost: Small set of virtual examples used in each iteration

(b) Avoiding local minima: Small changing set of virtual examples in each iteration cause a stochastic noise which benefits escaping from local minima.

**Summary of Experiments**

**Approach for Experiment 1** For experiment 1, a csMTL model was developed for the three domains of Logic Task, Cover-Type and Dermatology. The model used actual examples for training and validation as well as for consolidating the new task. The accuracy derived in this experiment serves as baseline for the subsequent experiments.

**Approach for Experiment 2 and Experiment 3** Three approaches are adopted to conduct the experiments over three domains. The first approach is to use actual examples for training and validation of the prior tasks as well as for the new task. The second approach is to generate a number of virtual examples for training and validation of the prior tasks and using actual examples for the new task. The third approach is to use randomly chosen examples from the set of virtual examples for training and validation of the prior tasks and using actual examples for the new task.

(a) Logic Task Domain: Sweep task rehearsal performs best in this domain. It maintains stability by retaining accurate prior knowledge as well as adheres to plasticity in consolidating new knowledge. Sweep task rehearsal performs 10 times faster compared to standard rehearsal.

(b) CoverType Domain: The sweep task rehearsal performs as well as standard rehearsal but not as well as the method using the actual examples.

(c) Dermatology Domain: The sweep task rehearsal performs as well as standard rehearsal but not as well as the method using the actual examples.
Future Work  Resolving the loss of prior task knowledge by considering the conditional probability between the input variables to generate the virtual examples. Using deep learning methods in combination with csMTL approach may solve this issue.


Abstract: In this paper we explore the topic of the consolidation of information in neural network learning. One problem in particular has limited the ability of a broad range of neural networks to perform ongoing learning and consolidation. This is "catastrophic forgetting", the tendency for new information, when it is learned, to disrupt old information. We will review and slightly extend the rehearsal and pseudorehearsal solutions to the catastrophic forgetting problem presented in Robins (1995). The main focus of this paper is to then relate these mechanisms to the consolidation processes which have been proposed in the psychological literature regarding sleep. We suggest that the catastrophic forgetting problem in ANNs is a problem that has actually occurred in the evolution of the mammalian brain, and that the pseudorehearsal solution to the problem in ANNs is functionally equivalent to the sleep consolidation solution adopted by the brain. Finally, we review related work by McClelland, McNaughton & O’Reilly(1995) and propose a tentative model of learning and sleep that emphasises consolidation mechanisms and the role of the hippocampus.

Annotation:

Consolidation  Consolidation of information is a principle requirement to ensure lifelong learning in artificial neural network(ANN). Consolidation enables integration of new information into the existing learning system. Transfer refers to using the existing information of the learning system to facilitate the learning or solving new information.

Catastrophic Forgetting  Catastrophic forgetting can be briefed as following: "If after its original training is finished a network is exposed to the learning of new information, then the originally learned information will typically be greatly disrupted or lost." (Robins 1995)
The extent to which catastrophic forgetting affects the disruption of the existing knowledge in the learning system or completely deletes the old information depends on the overlap of distribution in the new information. If the new information to be learned has a similar input pattern as the old information with a completely different output pattern, then the extent of catastrophic forgetting will be worst. But if the new information to be learned has the same input distribution with the same output pattern, then there may not be any affect of the catastrophic forgetting.

**Rehearsal** To reduce the affect of catastrophic forgetting in the learning system, rehearsal of a subset of the old information forces the learning system to retain the structure of the old information. (Robins 1995) In this paper, 15% of the base population is taken as the number of examples to be used for rehearsal. Two approaches are investigated for rehearsal. They are briefed as following:

(a) Random Rehearsal: A fixed random subset of old information is incorporated into the training process when any new information has to be learned by the system.

(b) Sweep Rehearsal: A dynamic random subset of old information is chosen into the training session each time any new information has to be learned by the system. The random subset may be different each time.

**Pseudorehearsal** Pseudorehearsal is the concept of using pseudoitems in place of real examples at the time of rehearsal in the training session of the new information to retain the structure of the old information in the learning system. Pseudoitems are created by passing randomly created input vector through the learning system and then recording the output generated for that particular input vector. In this way, a pseudopopulation is created to retain a mapping of the output function of the old information learned by the system. Again, two approaches are used for pseudorehearsal. They are briefed as following:

(a) Random Pseudorehearsal: Randomly chosen pseudoitems are used in the rehearsal buffer with the new examples to be learned in the training session.

(b) Sweep Pseudorehearsal: Each time new information is to be learned by the system, a new pseudopopulation is created and pseudoitems are randomly chosen from that pseudopopulation in the training session.
Experimentation Summary  For the random and sweep rehearsal regimes the average error of the generalisation population typically exceeded the average error of the base population by no more than 0.003 to 0.005. For the random pseudorehearsal the average error of the test population typically exceeded the average error of the base population by no more than 0.002, and for sweep pseudorehearsal the errors were indistinguishable.

Sleep Consolidation Hypothesis  "Dreams may reflect a memory processing mechanism inherited from lower species, in which information important for survival is reprocessed during REM sleep" (Winson 1990)

Comparison of Sleep Consolidation and Pseudorehearsal

(a) Preserving the structure of old information
(b) Rehearsing approximations of old information
(c) Not necessary for "compatible" information
(d) An "offline" process


Abstract:  Deep learning architectures have advanced the state-of-the-art in many machine learning applications such as computer vision, speech recognition, and natural language processing. However, deep learning architectures, like other machine learning methods, cannot work well if there is a limited amount of training data. Transfer learning aims to use existing knowledge of previously learned tasks to help the learning of a new task; this can speed up learning and increase accuracy. Transfer learning fits well in deep learning architectures because of unsupervised feature learning, development of a rich hierarchy of features, and greater plasticity created by unsupervised learning. Pretraining is a form of transfer learning even for single task learning. In this research, we compare several methods for transfer learning using deep learning architectures, especially Deep Belief Networks. These include representational, functional and combined transfer. The domain of handwritten digits is used to train and evaluate these methods using reconstruction cross entropy, classification accuracy, and speed of learning. Empirical studies show that DBNs can develop better hidden node features, have better reconstruction cross entropy, and better classification accuracy than backpropagation networks. The most efficient (requires less time to train a model) DBN transfer learning method is representational transfer, and
the most effective (best classification accuracy and reconstruction cross entropy) DBN transfer learning method is functional transfer. Context sensitive multitask learning in DBNs produce better models compared with alternative transfer learning approaches. Combined transfer in DBNs can produce more accurate models than representational transfer and can learn faster than functional transfer approaches. However combined transfer produces less accurate models than functional transfer approaches on their own.

**Annotation:**

**Introduction**  The thesis seeks to find out the fundamental differences and performance difference between deep learning architectures (DLA) and backpropagation network in terms of transfer learning. The question as to which kind of transfer is more efficient and effective in DLA is also investigated. The thesis investigates unsupervised representational, functional and combined transfer in deep belief networks and backpropagation networks and then compares them.

**Functional Transfer**  Most commonly with multiple task learning (MTL) (Caruana 1997), functional transfer uses examples from related tasks to create an inductive bias for the primary task, with no requirement for a learned model of the previous task. This method requires two or more tasks to be learned simultaneously to form a good mutual internal representation. This form of transfer has its greatest value in terms of increased generalization performance from the resulting hypotheses. Unfortunately, it often requires added computational space and time to develop a model. As an example, for the problem of self-driving cars, we can train a steering control model based on visual cues to control the vehicle. However, we can also provide auxiliary tasks, such as vehicle detection and pedestrian detection, which will help build internal representations that can be shared with the steering control model.

**Representational Transfer**  Representational transfer involves the direct or indirect assignment of known task representation to the model of a new target task (Silver and Mercer 1996). In this way the learning system is initialized in favor of a particular region of hypothesis space of the modeling system (Ring 1993; Shavlik and Deitterich 1991; Singh 1992). We consider this to be an explicit form of knowledge transfer from a source task to the target task. Representational transfer often results in
substantially reduced training time with no loss in the generalization performance of the resulting hypotheses. Unfortunately, it rarely develops more accurate models.

**Combined Transfer** Combined transfer in neural networks uses both representational transfer and functional approaches. Most commonly, using an MTL or csMTL approach, the representation of a model for a source task is used as the starting point for learning both the source task and the target task. Training examples for the source task maintain the functional accuracy of the neural network for the prior task, while the examples for the target task are used to transform the representation to benefit the new task. The intention is to benefit from the rapid transfer of prior representation as well as increased accuracy of functional transfer from related task examples.

**Discussion** Transfer learning is much more effective using DLA compared to backpropagation networks. The most efficient transfer learning in terms of time is the representational transfer in deep belief network (DBN). The most effective transfer learning in terms of classification accuracy is the functional transfer in DBNs, specifically using csMTL (Context sensitive multiple task learning) DBNs. Combined transfer learning produces more accurate models compared to representational transfer in DBNs and learns faster compared to functional transfer approaches in DBNs. However, the combined transfer approach produces slightly worse models compared with functional transfer approaches in DBNs. This is because of the stability-plasticity problem: the combined approach finetunes the weights of the network for the source task using backpropagation. This creates an accurate and stable representation for the source task, but unfortunately it is not plastic enough to consolidate the knowledge of the target task.


**Abstract**: Machine lifelong learning (ML3) is concerned with machines capable of learning and retaining knowledge over time, and exploiting previous knowledge to assist new learning. An ML3 system requires effective task retention, and effective consolidation of new tasks. This thesis presents an ML3 system using a context-sensitive multiple task learning (csMTL) neural network that functions as a consolidated domain knowledge
store. csMTL was developed in response to structural limitations of multiple task learning (MTL) for ML3. Instead of additional outputs for each task csMTL uses additional context inputs that indicate the associated task. The csMTL-based system is analyzed empirically using synthetic and real domains. The experiments focus on the effective retention of knowledge and the effective consolidation of new knowledge, using independent test set accuracy as a measure of effectiveness. The studies indicate that the methodology results in selective task retention when appropriate learning parameters are used. New task consolidation efficacy suffered using the same learning parameters. Experimentation also suggests that virtual instances (input-output pairs constructed from the consolidated domain knowledge) that correspond to real instances improves efficacy of retention and new task consolidation. Experimentation also indicates that representational transfer allows more effective retention, but at the cost of less effective new task consolidation.

**Annotation:**

**Introduction** The principle objective of this thesis is to identify the network structure and learning parameters required for long-term retention using a context sensitive multiple task learning(csMTL)-based consolidated domain knowledge(CDK) network. This thesis presents a machine lifelong learning system using a context-sensitive multiple task learning-based consolidated domain knowledge network. The long-term motivation in the design of this system is to create a machine that can effectively retain knowledge of previously learned tasks and use this knowledge to more effectively and efficiently learn new tasks.

**Inductive Transfer** Some machine learning algorithms use previously acquired knowledge to facilitate the learning of new knowledge or solving new or related tasks. The use of previous knowledge is referred to as inductive transfer(Caruana 1997).

**Multiple Task Learning(MTL)** A multiple task learning network, or MTL network, like a standard single task learning (STL) artificial neural network, uses a feed-forward, multi-layer network and the back-propagation of error learning algorithm. An MTL network, as shown in the figure 4 represents additional tasks as additional outputs, and all tasks are trained simultaneously, in parallel.
Limitations of MTL  MTL faces limitations when applied for lifelong machine learning problems (Silver, Poirier, and Currie 2008). Adding a new output for every new task is the source of the problems. A MTL network’s method of inductive transfer relies on task relatedness; for transfer to be effective, the tasks themselves must be sufficiently related in order to exploit the shared internal structure. It fails to account for relatedness on an instance level; if a task shares half of all its instances along with their outputs with another task, it would not have a meaningful correlation with the other task from the task-level perspective upon which an MTL network relies.

Context-sensitive Multiple Task Learning  csMTL differs from MTL in that, instead of additional outputs for each task, additional inputs are used, which indicate the instance context. The csMTL network is a feed-forward network, using back-propagation of error for training, using the hidden layer as feature detectors, similar to a multiple task learning network. The significant difference is the use of context input nodes, instead of additional output nodes used in MTL. The goal of the network is not to optimize the results for any single task, but to learn and exploit their internal representations within the network to make learning new tasks easier, and to improve upon the domain knowledge of older tasks.

Task consolidation with csMTL  A consolidated domain knowledge (CDK) network is used to generate the output for each previously learned task from a specified input, and a short-term network is used similarly for the new task. In this way, a virtual instance is generated that contains the input and an output for each task. The MTL-based CDK network makes use of both functional and representational transfer. Functional transfer includes training instances and learning parameters, and representational transfer includes the structure of the network and the values of the free parameters of the hypothesis, which are the
weight values for ANNs. The intention is that prior functional knowledge will be maintained or improved, but representational knowledge will necessarily change as new task knowledge is integrated into the CDK network.

Transfer from the MTL CDK network can be either functional or representational. Representational transfer is simple; use the weight values and configuration of the CDK network as initial weight values and configuration of the short-term network of the new task. Functional transfer is more complex; the CDK network is used to generate virtual instances. Many virtual instances are generated, and trained in parallel with the new task instances using the short-term network; that is the functional transfer.

Figure 2: Context-sensitive Multiple Task Learning with Consolidated Domain Knowledge


Abstract: We consider the problem of learning deep generative models from data. We formulate a method that generates an independent sample via a single feedforward pass through a multilayer perceptron, as in the recently proposed generative adversarial networks (Goodfellow et al. 2014). Training a generative adversarial network, however, requires careful optimization of a difficult minimax program. Instead, we utilize a technique from statistical hypothesis testing known as maximum mean discrepancy (MMD), which leads to a simple objective that can be interpreted as matching all orders of statistics between a dataset and samples from the model, and can be trained by backpropagation. We further boost the performance of this approach by combining our generative network with an auto-encoder network, using MMD to learn to generate codes that can then be decoded to produce samples. We show that the combination of
these techniques yields excellent generative models compared to baseline approaches as measured on MNIST and the Toronto Face Database.

**Annotation:**

**Maximum Mean Discrepancy**  Maximum mean discrepancy is evaluated to compare statistics (mean, variance) between two datasets and if they are similar then the samples are likely to come from the same distribution. The maximum mean discrepancy (MMD) is a distance-measure between distributions $P(X)$ and $Q(Y)$ which is defined as the squared distance between their embeddings in the reproducing kernel Hilbert space (If the difference between two functions are small then they are close for the points in those functions as well).

**Implementation**  The model is constructed by first training an auto-encoder and producing code representations of the data, then freezing the auto-encoder weights and learning a GMMN to minimize MMD between generated codes and data codes. A gaussian parzen window is used to evaluate the kernel density estimation (probability density function) for the training data as well as the test data.


**Abstract:**  We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

**Annotation:**
Introduction  "In the proposed adversarial nets framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles." "In this article, we explore the special case when the generative model generates samples by passing random noise through a multilayer perceptron, and the discriminative model is also a multilayer perceptron. We refer to this special case as adversarial nets. In this case, we can train both models using only the highly successful backpropagation and dropout algorithms (Hinton et al. 2012) and sample from the generative model using only forward propagation. No approximate inference or Markov chains are necessary."

Generative Adversarial Net  The adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons. To learn the generator’s distribution $p_g$ over data $x$, we define a prior on input noise variables $p_z(z)$, then represent a mapping to data space as $G(z; \theta_g)$, where $G$ is a differentiable function represented by a multilayer perceptron with parameters $\theta_g$. We also define a second multilayer perceptron $D(x; \theta_d)$ that outputs a single scalar. $D(x)$ represents the probability that $x$ came from the data rather than $p_g$. We train $D$ to maximize the probability of assigning the correct label to both training examples and samples from $G$. We simultaneously train $G$ to minimize $\log(1 - D(G(z)))$: In other words, $D$ and $G$ play the following two-player minimax game with value function $V(G,D)$: \[ \min(G) \max(D) \quad V(G,D) = E_{x \sim p_{data}}(x) \log D(x) \] + $E_{z \sim p_z}(z) \log(1 - D(G(z)))$.

Advantages and disadvantages  The advantages of generative adversarial nets are that Markov chain or inference are not needed during learning. Backpropagation is used for gradient learning and a diverse range of functions can be used in the model.

The disadvantage of generative adversarial nets is that there is no specific representation for $p_g(x)$. But the representation
achieved from the adversarial net is pretty sharp compared to the other frameworks.


Abstract: Deep multi-layer neural networks have many levels of non-linearities allowing them to compactly represent highly non-linear and highly-varying functions. However, until recently it was not clear how to train such deep networks, since gradient-based optimization starting from random initialization often appears to get stuck in poor solutions. Hinton et al. recently proposed a greedy layer-wise unsupervised learning procedure relying on the training algorithm of restricted Boltzmann machines (RBM) to initialize the parameters of a deep belief network (DBN), a generative model with many layers of hidden causal variables. This was followed by the proposal of another greedy layer-wise procedure, relying on the usage of autoassociator networks. In the context of the above optimization problem, we study these algorithms empirically to better understand their success. Our experiments confirm the hypothesis that the greedy layer-wise unsupervised training strategy helps the optimization by initializing weights in a region near a good local minimum, but also implicitly acts as a sort of regularization that brings better generalization and encourages internal distributed representations that are high-level abstractions of the input. We also present a series of experiments aimed at evaluating the link between the performance of deep neural networks and practical aspects of their topology, for example, demonstrating cases where the addition of more depth helps. Finally, we empirically explore simple variants of these training algorithms, such as the use of different RBM input unit distributions, a simple way of combining gradient estimators to improve performance, as well as on-line versions of those algorithms.

Annotation: Deep architectures are not always better than the shallow architectures (Larochelle and Bengio 2008). But if the problem is complex enough and there is enough training data to represent that complexity, then deep architecture has been proved to produce better results (Larochelle, Erhan, et al. 2007).

8. (Nair and Hinton 2010) Vinod Nair and Geoffrey E. Hinton (2010). “Rectified Linear Units Improve Restricted Boltzmann Machines”. In:
Abstract: Restricted Boltzmann machines were developed using binary stochastic hidden units. These can be generalized by replacing each binary unit by an infinite number of copies that all have the same weights but have progressively more negative biases. The learning and inference rules for these "Stepped Sigmoid Units" are unchanged. They can be approximated efficiently by noisy, rectified linear units. Compared with binary units, these units learn features that are better for object recognition on the NORB dataset and face verification on the Labeled Faces in the Wild dataset. Unlike binary units, rectified linear units preserve information about relative intensities as information travels through multiple layers of feature detectors.

Annotation:

Introduction In this paper, it has been shown that, for comparing faces and recognizing objects, binary hidden units of RBM can be replaced by implementing noisy rectified linear units. Then the performance is much better for recognizing objects compared to the RBM using binary hidden units.

Rectified Linear Units Real Neurons can be modeled by making a small alteration to binomial units and it becomes more useful for practical applications. For infinite copies of binomial unit with the same weight vector w and bias b and a small fixed offset to the bias for each unit, the sum of the probabilities of the copies is very close to the following form:

\[ \sum_{i=1}^{N} \sigma(x - i + 0.5) \approx \log(1 + e^x) \] where \( x = vw^T + b \).
Siamese architecture (Chopra, Hadsell, and LeCun 2005) is used for the experiment. In the figure, the feature extractor $F_W$ contains one hidden layer of noisy rectified linear units pre-trained as an RBM with parameters $W$. $F_W$ is applied to the face images $I_A$ and $I_B$, and the cosine distance $d$ between the resulting feature vectors $F_W(I_A)$ and $F_W(I_B)$ is computed. The probability of the two faces having the same identity is then computed as

$$Pr("Same") = \frac{1}{1+\exp((-wd+b))}$$

where $w$ and $b$ are scalar learnable parameters.


Abstract: The uniformity of the cortical architecture and the ability of functions to move to different areas of cortex following early damage strongly suggests that there is a single basic learning algorithm for extracting underlying structure from richly-structured, high-dimensional sensory data. There have been many attempts to design such an algorithm, but until recently they all suffered from serious computational weaknesses. This chapter describes several of the proposed algorithms and
shows how they can be combined to produce hybrid methods that work efficiently in networks with many layers and millions of adaptive connections.

Annotation:

Introduction In this report, different algorithms have been investigated to show how they can be combined to produce an efficient multilayer network.

Strategies
(a) Denial which ignores the face hand-coding the feature detectors or the weights in the network may cause solution for small problems, but it is not a scalable solution.
(b) Evolution which creates jitters in the weight or in the feature detectors to check if this improves the performance. Then again, for large number of weights or feature detectors, it is costly to check for a shift in performance for every change in weight or feature detector.
(c) Procrastination which considers that adding layers will retain more features from the input and hence the classification process will benefit from a higher layer. But it has to be ensured that in each layer the feature detectors are able to detect different regularities in the pattern of the same input data.
(d) Using calculus to calculate cross entropy at each hidden unit. This computation can be made very efficient by first computing, for each hidden unit, how the cross-entropy changes as the activity of that hidden unit is changed. When the affect of hidden unit activation on the cross entropy is known, then the weights can be adjusted to decrease the sum of cross entropies for all the training examples. First, it is necessary to choose initial random values for all the weights. If these values are small, it is very difficult to learn deep networks because the gradients decrease multiplicatively as we backpropagate through each hidden layer. If the initial values are large, we have randomly chosen a particular region of the weight-space and we may well become trapped in a poor local optimum within this region.
(e) It turns out that we can learn both sets of weights by starting with small random values and alternating between two phases of learning. In the "wake" phase, the recognition weights are used to drive the units bottom-up, and the binary states of units in adjacent layers can then be used to train the generative weights. In the "sleep" phase, the top-
down generative connections are used to drive the network, so it produces fantasies from its generative model.


Abstract: Humans have an impressive ability to reason about new concepts and experiences from just a single example. In particular, humans have an ability for one-shot generalization: an ability to encounter a new concept, understand its structure, and then be able to generate compelling alternative variations of the concept. We develop machine learning systems with this important capacity by developing new deep generative models, models that combine the representational power of deep learning with the inferential power of Bayesian reasoning. We develop a class of sequential generative models that are built on the principles of feedback and attention. These two characteristics lead to generative models that are among the state-of-the-art in density estimation and image generation. We demonstrate the one-shot generalization ability of our models using three tasks: unconditional sampling, generating new exemplars of a given concept, and generating new exemplars of a family of concepts. In all cases our models are able to generate compelling and diverse samples having seen new examples just once providing an important class of general-purpose models for one-shot machine learning.

Annotation: Introduction  Humans have the capability to analyze a given knowledge and then generate different variations on the observed data. The same principle has been used in deep generative model to implement one-shot generalization. The paper uses the concept of analysis-synthesis process of feedback and attention to generate variations of examples of the observed data.

Two principles are central to our approach: feedback and attention. These principles allow the models we develop to reflect the principles of analysis-by-synthesis, in which the analysis of observed information is continually integrated with constructed interpretations of it (Yuille and Kersten 2006; Nair, Susskind, and Hinton 2008; Erdogan, Yildirim, and Jacobs 2015). Analysis is realized by attentional mechanisms that allow us to selectively process and route information from the observed data into the model. Interpretations of the data are then obtained by
sets of latent variables that are inferred sequentially to evaluate the probability of the data. The aim of such a construction is to introduce internal feedback into the model that allows for a "thinking time" during which information can be extracted from each data point more effectively, leading to improved inference, generation and generalization. We shall refer to such models as sequential generative models.

**Sequential Generative Model**  A sequential generative model is a natural extension of the latent variable models used in Variational Autoencoders. Instead of generating the K latent variables of the model in one step, these models sequentially generate T groups of k latent variables (K = kT), i.e. using T computational steps to allow later groups of latent variables to depend on previously generated latent variables in a non-linear way.

**Feedback and Attention**  Focusing on a specific part of knowledge, analyzing that and then sequentially building up the interpretation is the basis of analysis and synthesis process which is used to implement the model.

Figure 4: The model generates examples provided the first row

**Limitation**  The model does one-shot generalization, but does not do one-shot learning. That is the model is not updated after observing a new input, rather an inference process is used in test time to perform inferential tasks on the new data points.

Abstract: Memory units have been widely used to enrich the capabilities of deep networks on capturing long-term dependencies in reasoning and prediction tasks, but little investigation exists on deep generative models (DGMs) which are good at inferring high-level invariant representations from unlabeled data. This paper presents a deep generative model with a possibly large external memory and an attention mechanism to capture the local detail information that is often lost in the bottom-up abstraction process in representation learning. By adopting a smooth attention model, the whole network is trained end-to-end by optimizing a variational bound of data likelihood via auto-encoding variational Bayesian methods, where an asymmetric recognition network is learnt jointly to infer high-level invariant representations. The asymmetric architecture can reduce the competition between bottom-up invariant feature extraction and top-down generation of instance details. Our experiments on several datasets demonstrate that memory can significantly boost the performance of DGMs on various tasks, including density estimation, image generation, and missing value imputation, and DGMs with memory can achieve state-of-the-art quantitative results.

Annotation:

Deep Generative Model with Memory Although current DGMs are able to extract high-level abstract representations, they may not be sufficient in generating high-quality input samples. This is because more abstract representations are generally invariant or less sensitive to most specific types of local changes of the input.

It remains a challenge for deep generative models to generate real data, especially for images that have complex structures. Simply increasing the model size is apparently not wise, as it may lead to serious over-fitting without proper regularization as well as heavy computation burden.

The overall architecture of the proposed model is an interleave between stochastic layers and deterministic layers, where each deterministic layer is associated with an external memory to capture local variant information. An attention mechanism is used to record information in the memory during learning and retrieve information from the memory during data generation. This attention mechanism can be trained because the invariant information and local variant information are correlated, e.g., both containing implicit label information. Both the memory and attention mechanisms are parameterized as differentiable components with some smooth nonlinear transformation func-
tions. Such a design enables learning the whole network end-to-end by developing a stochastic variational method, which introduces a recognition network without memory to characterize the variational distribution. A probabilistic deep generative model (DGM) with a possibly large external memory as well as a soft attention mechanism is presented in the paper.

Discussion
In the generation/reconstruction of data samples, memory helps to retain the detail information, which is often lost in the abstraction procedure.


Abstract: We update Complementary Learning Systems theory, which holds that intelligent agents must possess two learning systems, instantiated in mammals in neocortex and hippocampus. The first gradually acquires structured knowledge representations while the second quickly learns specifics of individual experiences. We extend the role of replay of hippocampal memories in the theory, noting that replay allows goal dependent weighting of experience statistics. We also address recent challenges to the theory and extend it by showing that recurrent activation of hippocampal traces can support some forms of generalization and that neocortical learning can be rapid for information consistent with known structure. Finally, we note the relevance of the theory to the design of artificial intelligent agents, highlighting connections between neuroscience and machine learning.

Annotation:
Complementary Learning Systems Theory According to the complementary learning systems (CLS) theory (McClelland, McNaughton, and Reilly n.d.), two complementary systems are required for effective learning process. One acts as the repository for information on the environment, located at neocortex and the second one works to quickly attain information on recent experiences and individual items, located at hippocampus.

Structured Knowledge Representation in Neocortex This system is referred to as parametric rather than instance-based. Learning in such a system will be slow for two reasons:
(a) As each experience represents a single sample in the environment, a slow learning rate is required to effectively aggregate information underlying the population statistics.

(b) The effective adjustment of each connection depends on the values of all the other connections. So the initial learning will be slower where the changes to the connection weights will be noisy as all the other connections have not settled down with values yet.

Such a parametric system is very advantageous, but will not perform well by itself due to two reasons:

(a) Such a system cannot quantify the evaluation of a single experience no matter how significant it is. For example, an encounter with a lion in a watering hole is a very significant experience and further such encounter should be avoided. But setting the proper adjustments to the connection weights evaluating this single experience will not be possible in such a system.

(b) The quick adjustment of connection weights to consolidate new information can disrupt the existing knowledge base severely.

Instance-based Representation in the Hippocampal System
Such a non-parametric system can alleviate the limitations of neocortical system in taking into consideration the importance of an individual experience. But by itself hippocampal systems are not better performer either. It has the limitation of capacity and generalization ability.

Joint contribution to task performance
Performance on different tasks are dependant on both the neocortical and hippocampal system with varying degree depending on the task. Neocorical memory helps with relating the background knowledge of the task and hippocampal systems helps with associating the context with the task.

Integration of Knowledge in Neocortex from Hippocampal Representation
Knowledge of an experience is slowly integrated into neocortical system from hippocampal representation by replaying the contents of the experience mixed with other contents of experiences. How the other events or experiences are chosen for the interleaving during the replay is still an open problem. This process is referred as Systems Level Consolidation (Franklan and Bontempi 2005) which minimizes the disruption of existing knowledge in the neocortical system.
by interleaving other events while replaying the contents of the event to be learned.

**Role of Hippocampus in avoiding the environment statistics**  Hippocampal system does not provide an exact representation of the event during replay, rather it replays a biased representation of the event. So avoiding the environment statistics, hippocampus provides weight adjustments to accommodate preferential events to shape up the neocortical learning. This shows that the learning has an inclination towards the organism’s objectives rather than the general statistics of the environment. The decision for the reweighting comes from the multimodal sensory inputs while experiencing the event and events occurring before and afterwards.

**CLS Theory and Machine Learning Research**  Deep neural networks mimic the slow learning process of the neocortical system in sharing the same goal of maintaining an effective representation of the statistics of the environment by training on a vast number of examples.

But to enable the continual learning process, the ability to integrate new information without catastrophic forgetting in the existing knowledge base, neural networks are shown to perform better with an external memory like hippocampus. This concept has been proved in Nural Turing Machine(Graves, Wayne, and Danihelka 2014). The external memory has content-addressable properties which functions like pattern completion in the hippocampus.

**Open Questions**
(a) The conditions which activate the weighting adjustments in the hippocampus
(b) The changes to the hippocampal representation corresponding to the changes in the neocorical system
(c) The status of hippocampal memory after systems level consolidation
(d) Conditions for rapid systems level consolidation
(e) Contribution of neocortex to support hippocampus in continual learning
(f) Algorithms and schemes for effective implementation of external memory
Abstract: Continual learning is the constant development of increasingly complex behaviors; the process of building more complicated skills on top of those already developed. A continual-learning agent should therefore learn incrementally and hierarchically. This paper describes CHILD, an agent capable of Continual, Hierarchical, Incremental Learning and Development. CHILD can quickly solve complicated non-Markovian reinforcement learning tasks and can then transfer its skills to similar but even more complicated tasks, learning these faster still.

Annotation:

Continual Learning Continual learning is the process where the agent continues the process of learning across different tasks and experiences and use that knowledge to learn other tasks and then can build on the previously acquired knowledge. In this process, the agent’s learning process is not directly manipulated, rather the outward senses, actions and rewards are used to present training to the agent.

Transfer in Learning Transfer in supervised learning uses the classification knowledge gathered in previous tasks as a bias to learn related tasks. Transfer in reinforcement learning uses the knowledge achieved to gain one goal to achieve other goals effectively. Continual learning uses the skills developed using the previous tasks to gain skills on more complex tasks.

CHILD CHILD is an agent which uses Continual, Hierarchical, Incremental Learning and Development.

Temporal Transition Hierarchies (TTH) TTH is a neural-network based learning system which starts with learning the probability of each event and then as learning progresses creates new units to use preceding contextual information to predict events more accurately.

Structure In TTH, the neural network units are used to represent the transition hierarchies. There are three types of units: a sensory unit, an action unit and high-level unit. The sensory units work as input units whereas the action units and the high-level units work as the non-input units. The high-level units are added as new units when the weight or the connection between
the sensory unit and the action unit is strongly forced to change in opposite directions. The new unit represents the different contexts for which the different weights should be used.

**Experimentation Summary**  Nine mazes were used increasing in complexity from maze 1 to maze 9 and then the TTH combined with Q-learning process is used to train CHILD. The experiments were done in two ways, one was learning the mazes from scratch and the other was using continual learning to learn the mazes. The continual learning process for CHILD performs better than the process using the learning from scratch. As the mazes get more complicated and ambiguous, the performance difference between the two approaches become higher. After training on the fourth maze, the continual learning approach performs reasonably better and faster than the learning from scratch approach.

**Non-catastrophic interference**  To test the ability to retain old information, CHILD was tested on the first maze after being trained on the ninth maze, the average testing performance was 20% worse than its initial ability. But on average, 20% less training steps were required and in two-thirds of the cases, no pre-training was required. So it proves that, CHILD prevents catastrophic interference or forgetting of the previously acquired knowledge.

**Distributed Senses**  CHILD was proved to generalize more effectively in the continual learning process. In some cases, CHILD was able to solve all the mazes just by training on the first two mazes. It developed a rule to proceed following the border of the maze in clockwise direction until reaching the goal. But in most cases, it developed a direct route to the goal at the cost of creating a higher number of new units.

**Summary**  The higher the complexity of the reinforcement tasks, the better CHILD performs with continual learning. According to the author, continual learning is the most reasonable strategy for developing any learning algorithm, as any complex task needs to be learned incrementally and hierarchically by the algorithm to result in an effective performance. Agents should not be built to solve specific tasks, rather they should adapt to the changing problem and learn to build on the previous skills to solve the more complex problems based on the skills acquired for solving the simple tasks.
References


3. James L. McClelland, Bruce L. McNaughton, and Randall C. O’Reilly. “Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory”. In: Psychological Review 102(3), pp. 419–457


