Open World Lifelong Learning
A Continual Machine Learning Course

Teacher

Dr. Martin Mundt,

hessian.AI-DEPTH junior research group leader on Open World Lifelong Learning (OWLL)
& researcher in the Artificial Intelligence and Machine Learning (AIML) group at TU Darmstadt

Time

Every Tuesday 17:30 - 19:00 CEST

Course Homepage

http://owll-lab.com/teaching/cl_lecture

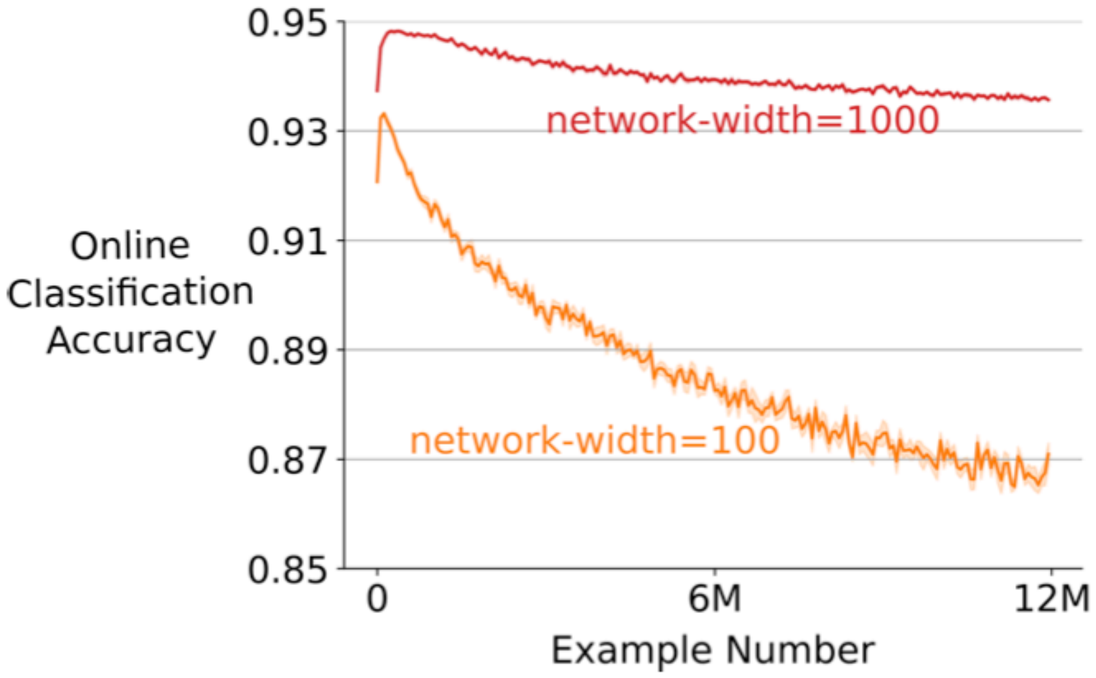
https://www.youtube.com/playlist?list=PLm6QXeaB-XkA5-lVBB-h7XeYzFzgSh6sk

Week 4: Rehearsal - Knowledge Retention Part 2

Recall: How to avoid forgetting?

The undesired trivial solution: large amounts of parameters + data accumulation

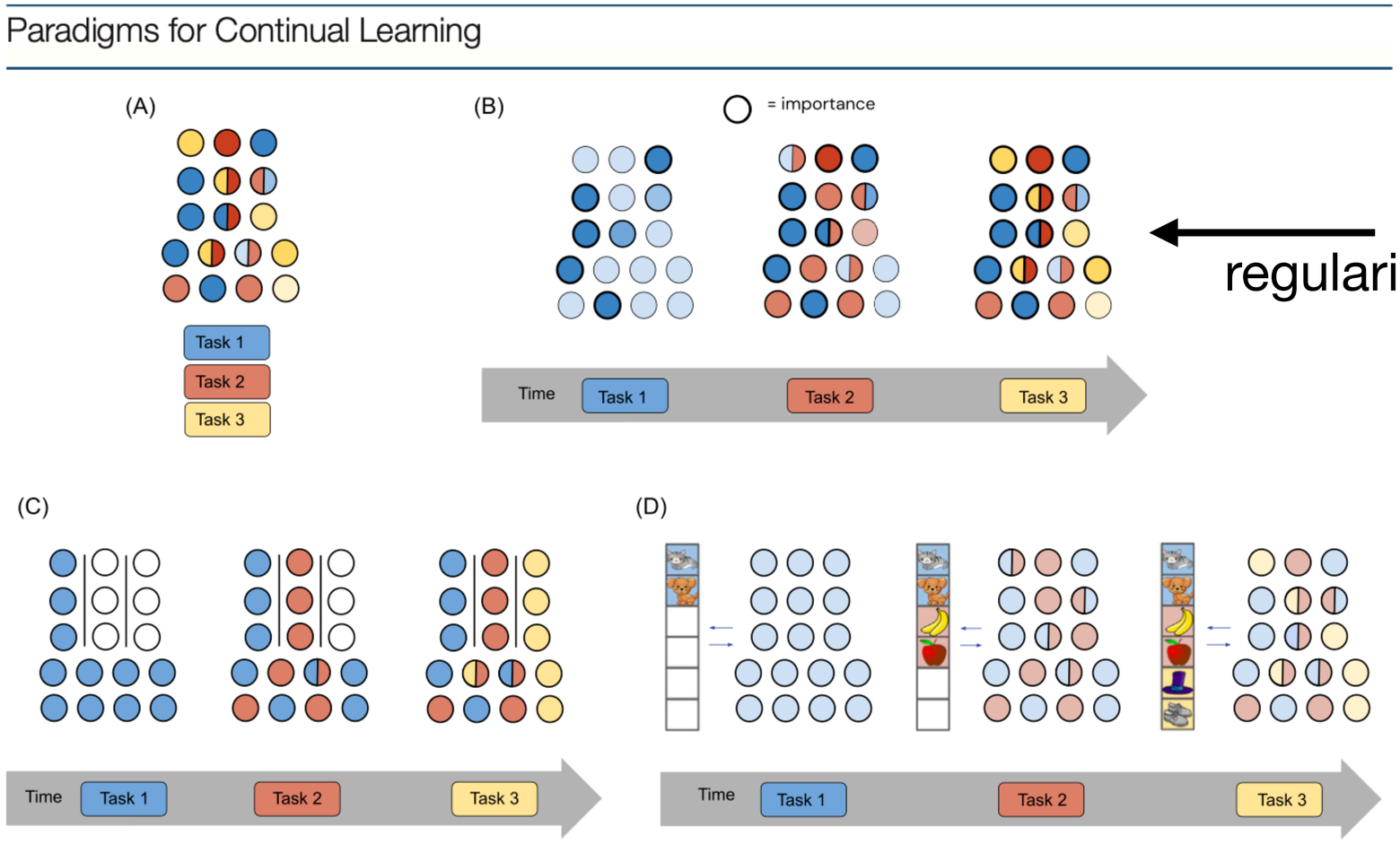
But also, caution, if we are constrained by capacity, we won't learn indefinitely!



Continual Backprop: Stochastic Gradient Descent with Persistent Randomness, Dohare et al, arXiv preprint:2108:06325

For now, we will still continue assuming that have sufficiently many parameters

Recall: How to avoid forgetting?



Last time
regularization/distillation

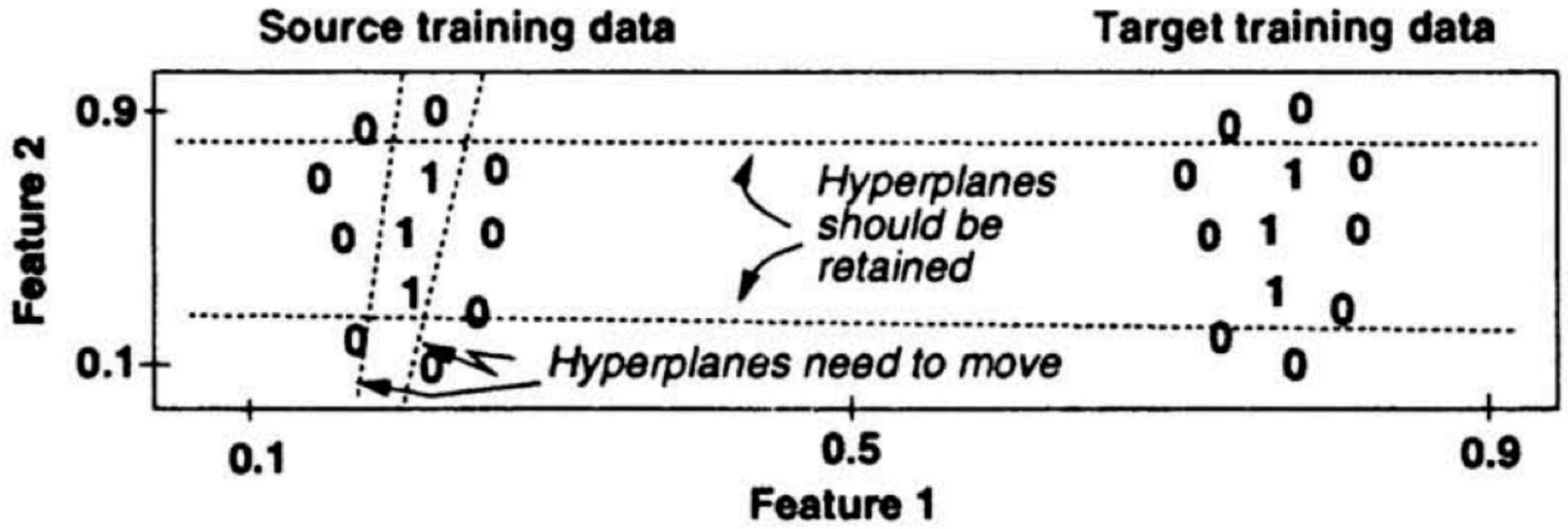
Today
(Pseudo-)Rehearsal

Hadsell et al, “Embracing Change: Continual Learning in Deep Neural Networks”, Trends in Cognitive Sciences 24:12, 2020

Figure 1. (A) Independent and identically distributed learning methods are standard for nonsequential, multitask learning. In this regime, tasks are learned simultaneously to avoid forgetting and instability. (B) Gradient-based approaches preserve parameters based on their importance to previously learned tasks. (C) Modularity-based methods define hard boundaries to separate task-specific parameters (often accompanied by shared parameters to allow transfer). (D) Memory-based methods write experience to memory to avoid forgetting.

Rehearsal: intuition

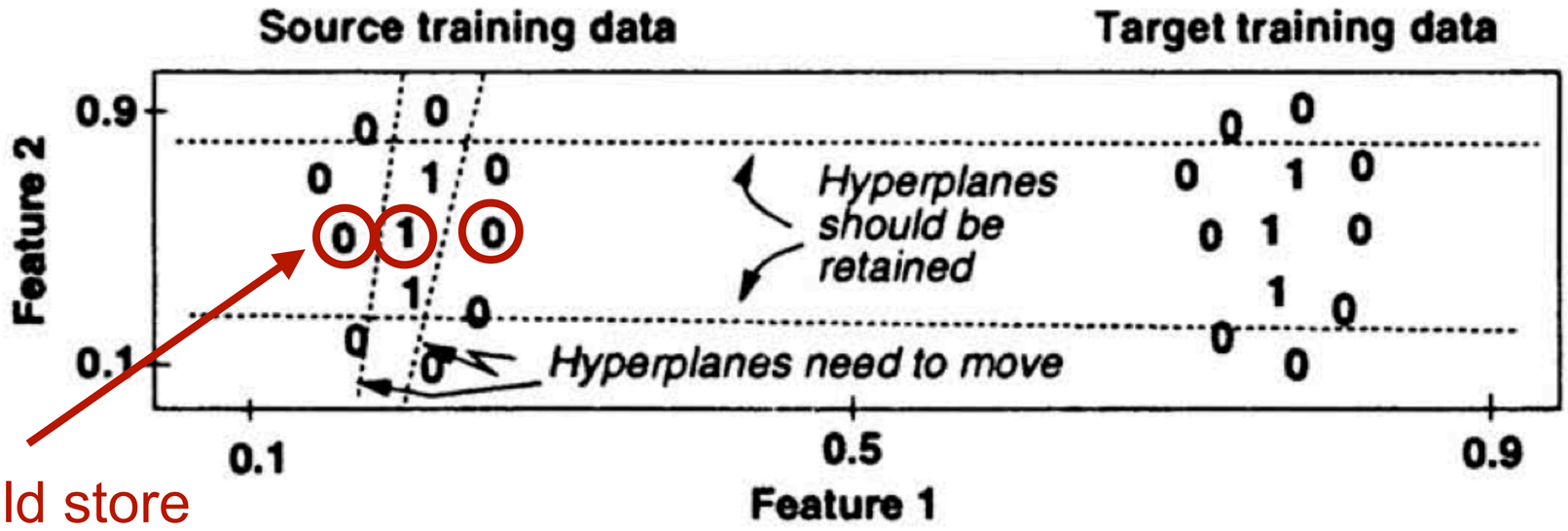
Assuming privacy is not a concern & that we have auxiliary memory:
some data is more relevant than other, can we retain a subset?



"Discriminability-Based Transfer between Neural Networks", L. Y. Pratt, NeurIPS 1992

Rehearsal: intuition

Assuming privacy is not a concern & that we have auxiliary memory:
some data is more relevant than other, can we retain a subset?

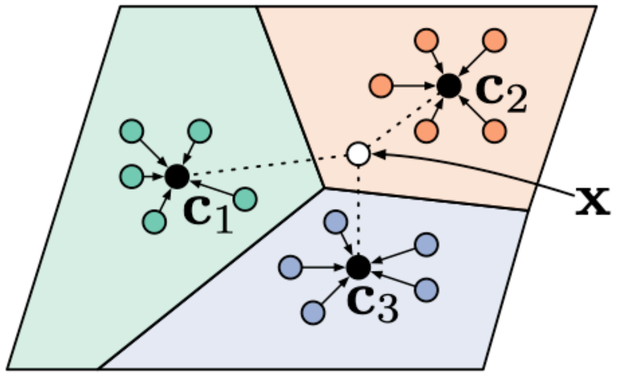


Maybe we could store these few examples?

"Discriminability-Based Transfer between Neural Networks", L. Y. Pratt, NeurIPS 1992

Nearest means classifiers

Let’s start with an example this time and then discuss desiderata



(a) Few-shot

Recall lecture 2: few shot learning

“Prototypical Networks for Few-shot Learning”,
Snell et al, NeurIPS 2017

- Get prototype as mean vector of each class
- Given a distance function d, classify according to distances to the prototypes in embedding space
- Note, nomenclature is inconsistent: prototypes, exemplars, core sets (will be defined later)

iCaRL: classification

iCaRL: incremental classifier and representation learning

- Stores a subset of data in a fixed size memory buffer
- Classifies based on nearest class means
- Consecutively replaces parts of memory buffer with new examples

Algorithm 1 iCaRL CLASSIFY

input x // image to be classified
require $\mathcal{P} = (P_1, \ldots, P_t)$ // class exemplar sets
require $\varphi : \mathcal{X} \to \mathbb{R}^d$ // feature map
  for $y = 1, \ldots, t$ **do**
    $\mu_y \leftarrow \frac{1}{|P_y|} \sum_{p \in P_y} \varphi(p)$ // mean-of-exemplars
  end for
  $y^* \leftarrow \underset{y=1,\ldots,t}{\operatorname{argmin}} \|\varphi(x) - \mu_y\|$ // nearest prototype
output class label y^*

Rebuffi et al, “iCaRL: Incremental Classifier and Representation Learning”, CVPR 2017

iCaRL: picking exemplars

How is our memory buffer filled with specific examples?

- Iteratively: one by one, based on "herding" (Welling ICML 2009)
- Pick exemplars to best approximate the overall mean
- For a size of k exemplars: loop k times

Algorithm 4 iCaRL CONSTRUCTEXEMPLARSET

input image set $X = \{x_1, \ldots, x_n\}$ of class y
input m target number of exemplars
require current feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$
  $\mu \leftarrow \frac{1}{n} \sum_{x \in X} \varphi(x)$ // current class mean
  for $k = 1, \ldots, m$ **do**
    $p_k \leftarrow \underset{x \in X}{\operatorname{argmin}} \left\| \mu - \frac{1}{k} [\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)] \right\|$
  end for
  $P \leftarrow (p_1, \ldots, p_m)$
output exemplar set P

Rebuffi et al, "iCaRL: Incremental Classifier and Representation Learning", CVPR 2017

iCaRL: replacing exemplars

Our memory buffer is limited, how do we later remove samples?

- Memory buffer is like a prioritized list
- Later picked exemplars for a task “weigh” less
- Simply cut and repopulate

Algorithm 5 iCaRL REDUCEEXEMPLARSET

input m // target number of exemplars
input $P = (p_1, \ldots, p_{|P|})$ // current exemplar set
$P \leftarrow (p_1, \ldots, p_m)$ // *i.e.* keep only first m
output exemplar set P

Rebuffi et al, “iCaRL: Incremental Classifier and Representation Learning”, CVPR 2017

iCaRL: training

How do we train incrementally?

- Concatenate dataset with exemplars/ interleave exemplars into training
- Pick new exemplars (not shown on the right) + replace existing
- Additionally use knowledge distillation

Algorithm 3 iCaRL UPDATEREPRESENTATION

input $X^s, \ldots, X^t$ // training images of classes $s, \ldots, t$
require $\mathcal{P} = (P_1, \ldots, P_{s-1})$ // exemplar sets
require Θ // current model parameters
// form combined training set:

$$\mathcal{D} \leftarrow \bigcup_{y=s,\ldots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\ldots,s-1} \{(x,y) : x \in P^y\}$$

// store network outputs with pre-update parameters:
for $y = 1, \ldots, s-1$ **do**
$q_i^y \leftarrow g_y(x_i)$ for all $(x_i, \cdot) \in \mathcal{D}$
end for
run network training (*e.g.* BackProp) with loss function

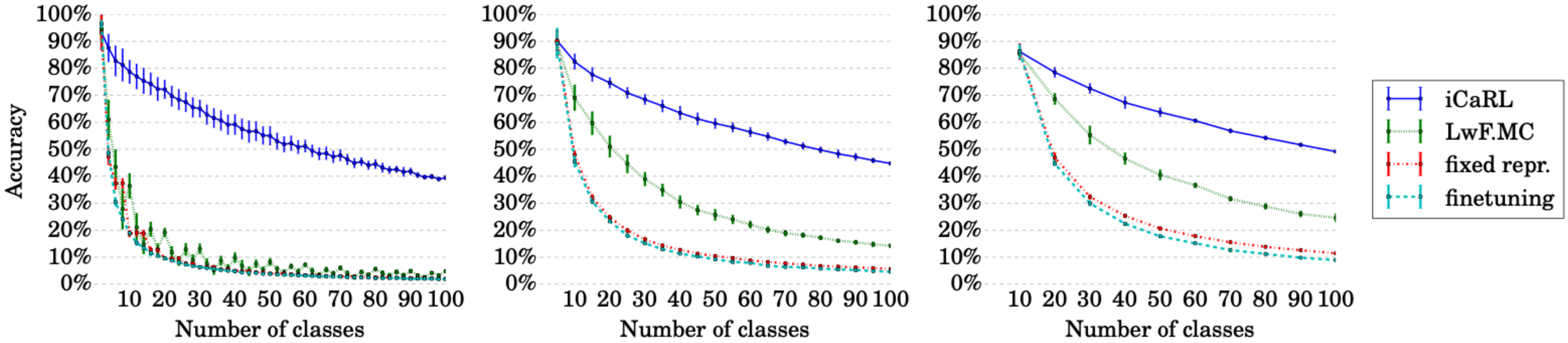
$$\ell(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Big[\sum_{y=s}^{t} \delta_{y=y_i} \log g_y(x_i) + \delta_{y\neq y_i} \log(1-g_y(x_i)) + \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1-q_i^y) \log(1-g_y(x_i)) \Big]$$

that consists of *classification* and *distillation* terms.

Rebuffi et al, “iCaRL: Incremental Classifier and Representation Learning”, CVPR 2017

iCaRL & knowledge distillation

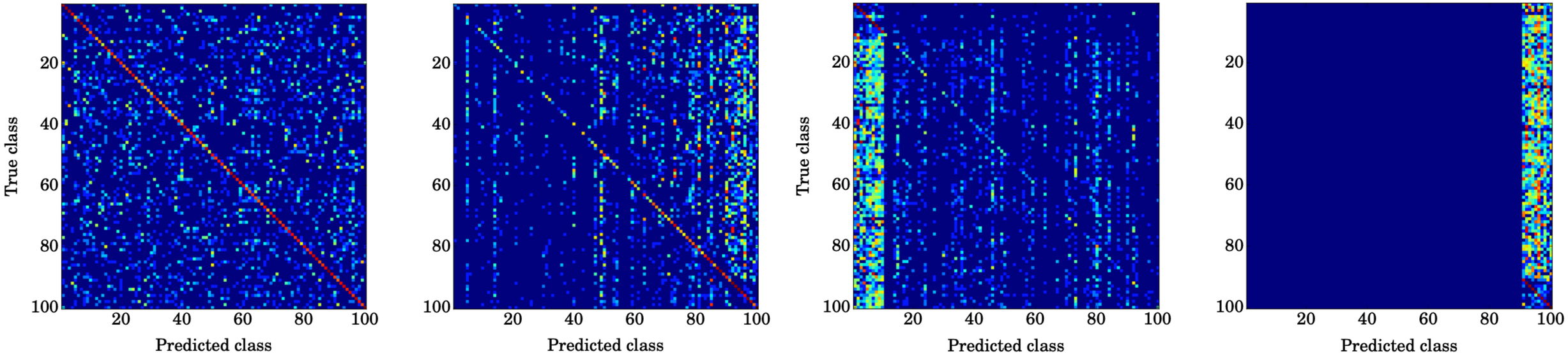
Example where classes of CIFAR100 are learned incrementally: exemplars are crucial



Rebuffi et al, “iCaRL: Incremental Classifier and Representation Learning”, CVPR 2017

iCaRL & knowledge distillation

Confusion matrices empirically confirm our intuition

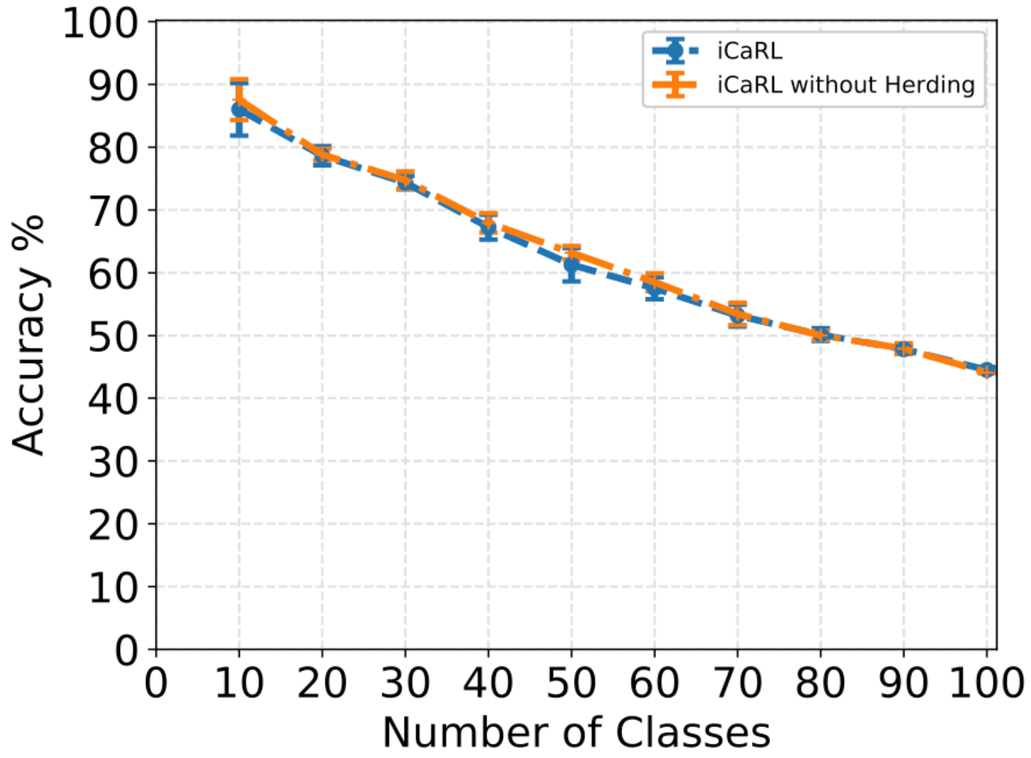


(a) iCaRL (b) LwF.MC (c) fixed representation (d) finetuning

Rebuffi et al, “iCaRL: Incremental Classifier and Representation Learning”, CVPR 2017

Role of extraction algorithm

Does the herding selection algorithm outperform random selection?



Javed et al, “Revisiting Distillation and Incremental Classifier Learning”, ACCV 2018

Role of memory budget

How expected are our observations?

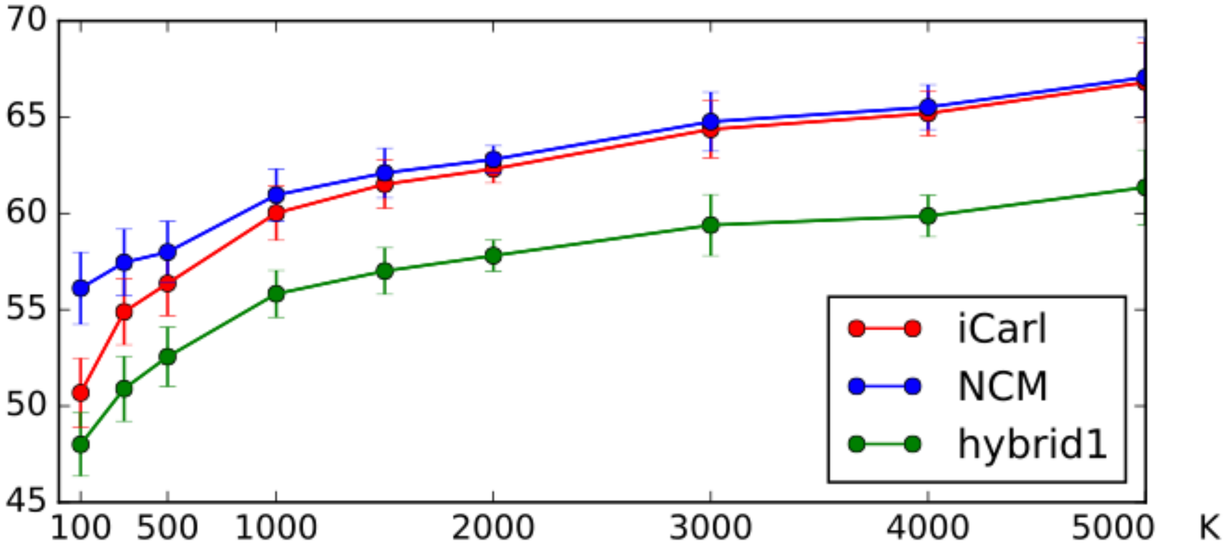


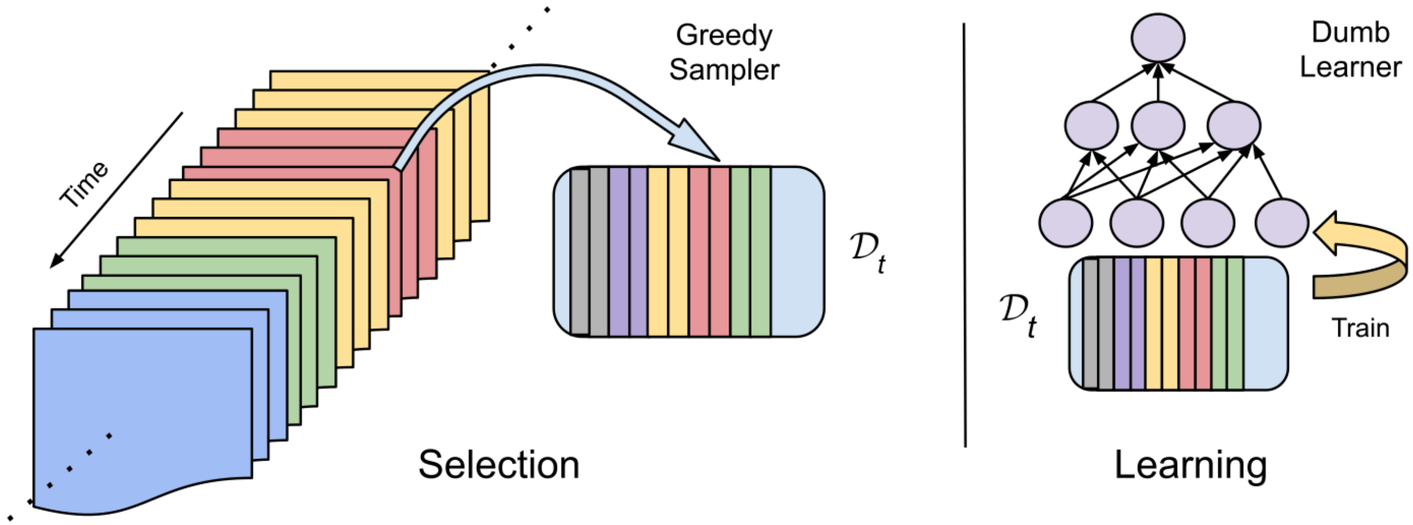
Figure 4: Average incremental accuracy on iCIFAR-100 with 10 classes per batch for different memory budgets K.

Rebuffi et al, “iCaRL: Incremental Classifier and Representation Learning”, CVPR 2017

Role of memory

Is it really just the data subset that we retain?

A “dumb learner” comparison suggests that we may get similar performance if we just train on the exemplar subset



Prabu et al, “GDumb: A Simple Approach that Questions our Approach to Continual Learning”, ECCV 2020

Role of memory

Are memory buffers of real examples the solution to continual learning?

"*While it is an effective method in ANNs, rehearsal is unlikely to be a realistic model of biological learning mechanisms, as in this context the actual old information (accurate and complete representation of all items ever learned by the organism) is not available. Pseudorehearsal is significantly more likely to be a mechanism which could actually be employed by organisms as it does not require access to this old information, it just requires a way of approximating it.*"

R. French, "Pseudo-recurrent Connectionist Networks:
An Approach to the Sensitivity-Plasticity Dilemma",
Connection Science 9:4, 1997

(Pseudo-)rehearsal

A rehearsal alternative

"*Pseudorehearsal is based on the use in the rehearsal process of artificially constructed populations of "pseudoitems" instead of the "actual" previously learned items. A pseudoitem is constructed by generating a new input vector (setting at random 50% of input elements to 0 and 50% to 1 as usual), and passing it forward through the network in the standard way. Whatever output vector this input generates becomes the associated target output*"

A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal", Journal of Neural Computing 7: 123-146, 1995

Pseudo-rehearsal

Complementary learning systems theory
(McClelland et al, Psychological Review 102:3, 1995), *simplified picture here*

- Hippocampus: short-term adaptation & rapid learning of novel information
- Neocortex: slow learning, to consolidate & build up overlapping representations
- Hippocampus “plays back” over time to neocortex

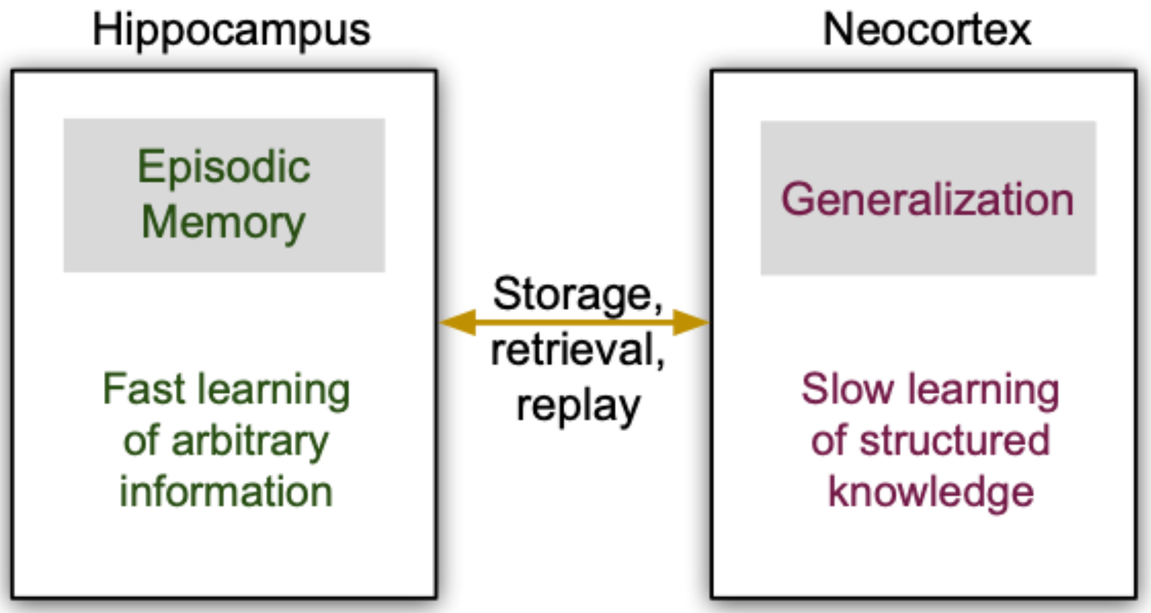
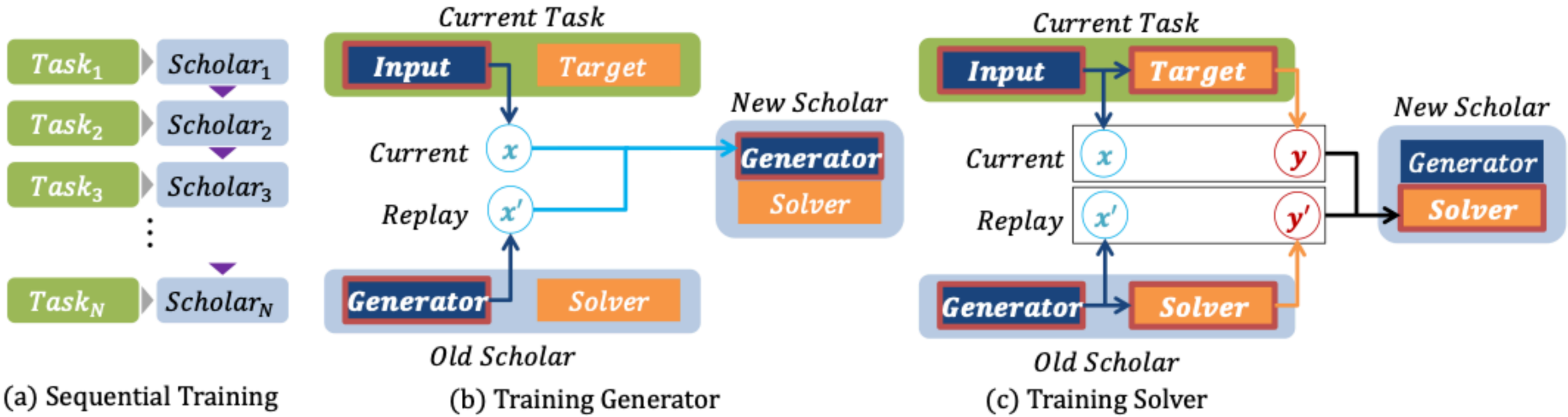


Figure from Parisi et al, “Continual Lifelong Learning with Neural Networks: A Review”, Neural Networks 113, 2019

Deep Generative Replay

We could train two machine learning models:

a “generator” (there are many types) + “task solver” -> then alternate training

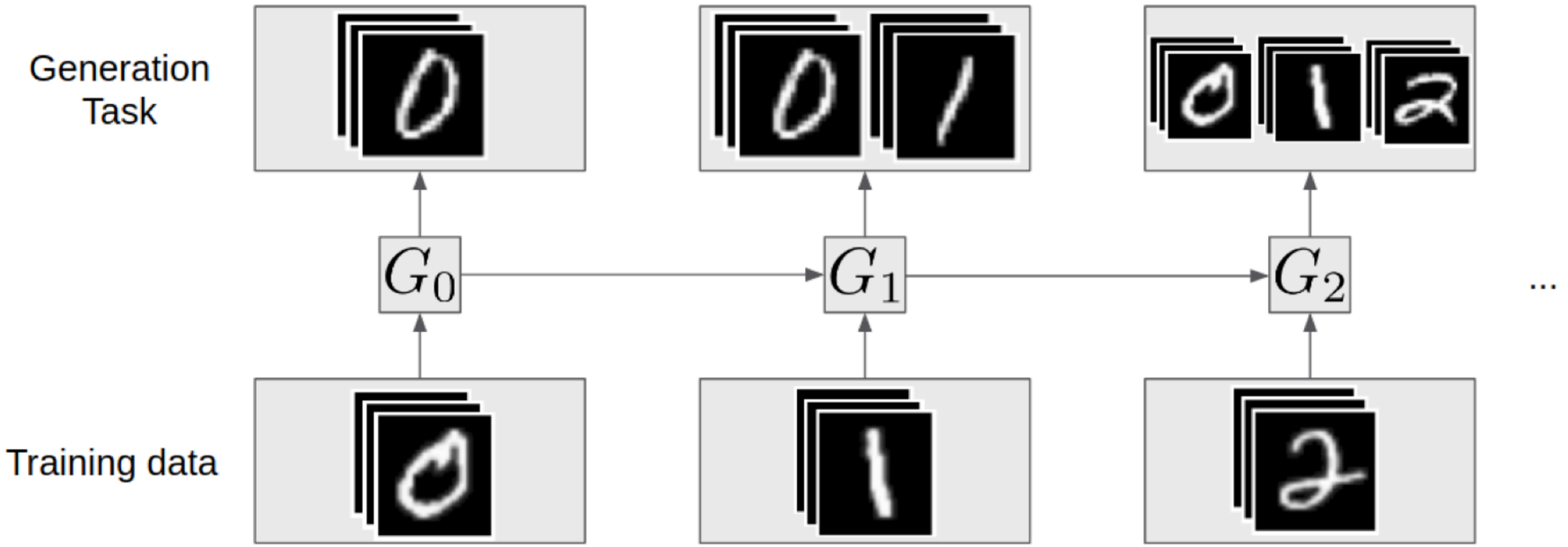


Shin et al, “Continual Learning with Deep Generative Replay”, NeurIPS 2017

Deep Generative Replay

We could train two machine learning models:

a “generator” (there are many types) + “task solver” -> then alternate training

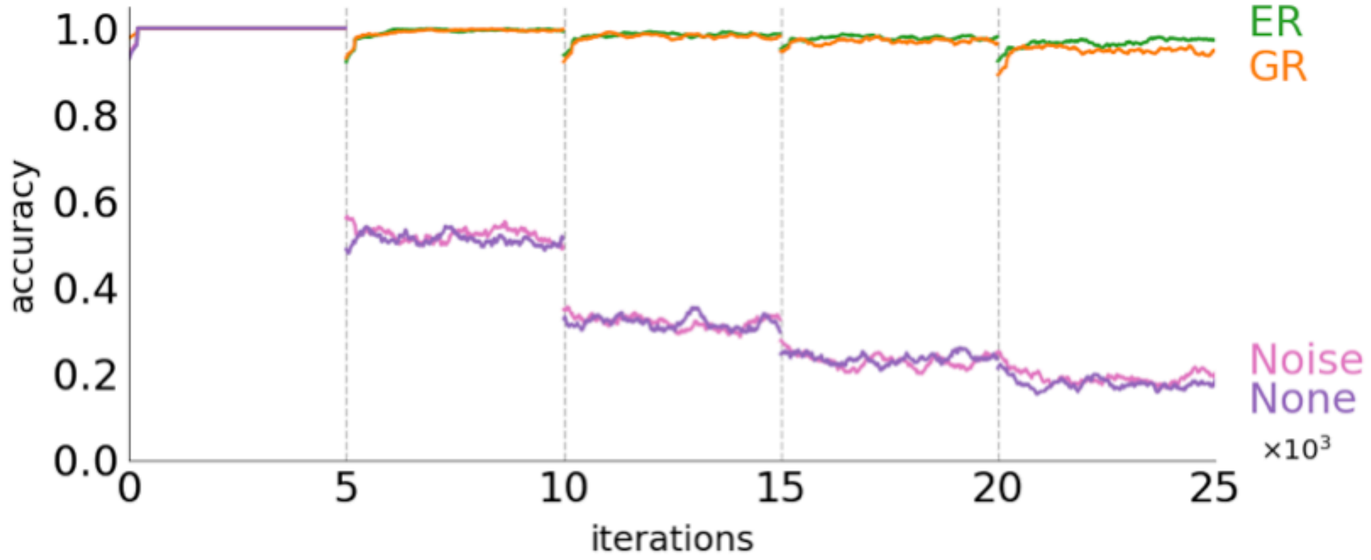


Lesort et al, “Generative Models from the perspective of Continual Learning”, IJCNN 2019

Exemplar/Generative Rehearsal

- Exemplar Rehearsal (ER) and Generative (Pseudo-)Rehearsal (GR) can work equally well if we have a powerful generator

- In contrast, randomly rehearsed sampled noise patterns will no longer work on complex tasks



Shin et al, “Continual Learning with Deep Generative Replay”, NeurIPS 2017

A short interlude: generative models

Generative models

- A **discriminative** model typically learns something like p(y|x)

- A **generative** model also learns about the data distribution p(x) & the process by which data is created (the generative factors)

- Having a generative model **does not mean we cannot also solve discriminative** tasks

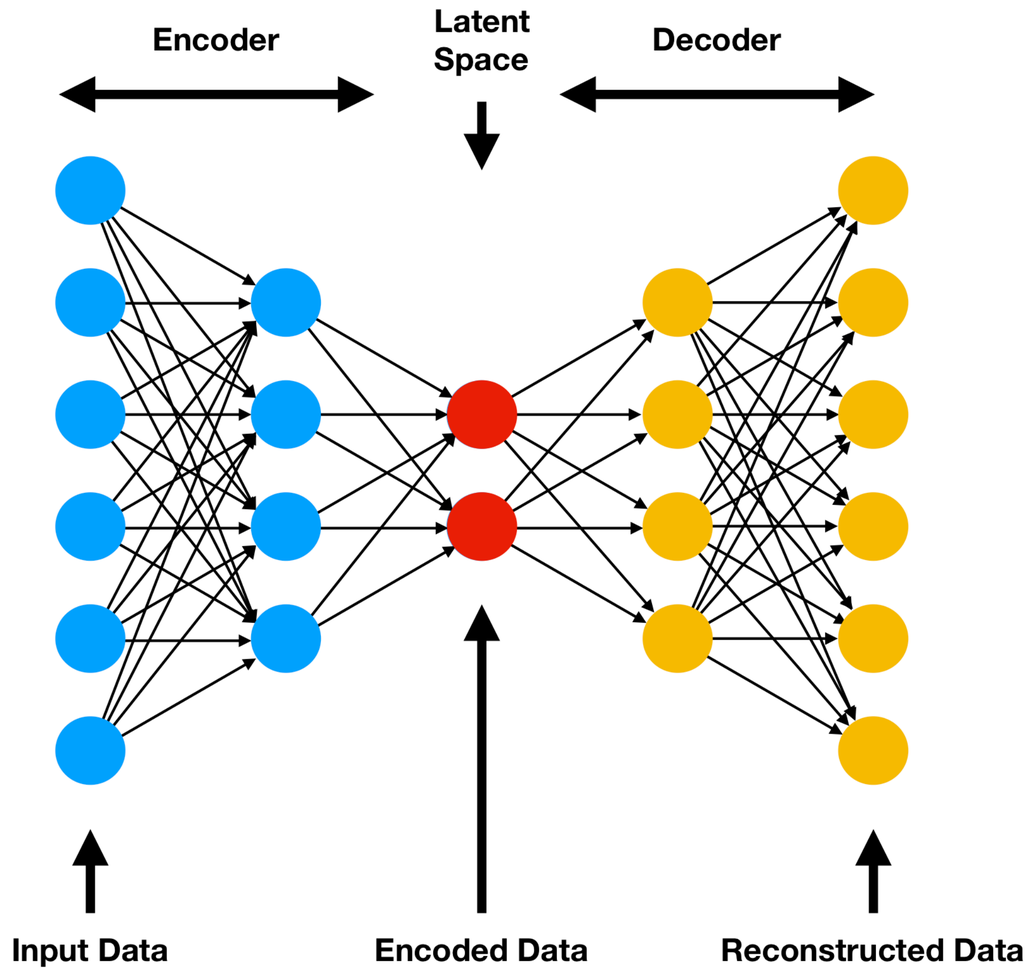
$$p(x,y) = p(y|x)p(x)$$

Autoencoders

Let’s take a look at **autoencoders**

Why? To later see that we don’t necessarily need two separate models:

- Learn a representation, an “encoding” of the data
- An **encoder** maps to a “code”: “**latent** variables”
- A **decoder** learns to **reconstruct** the input

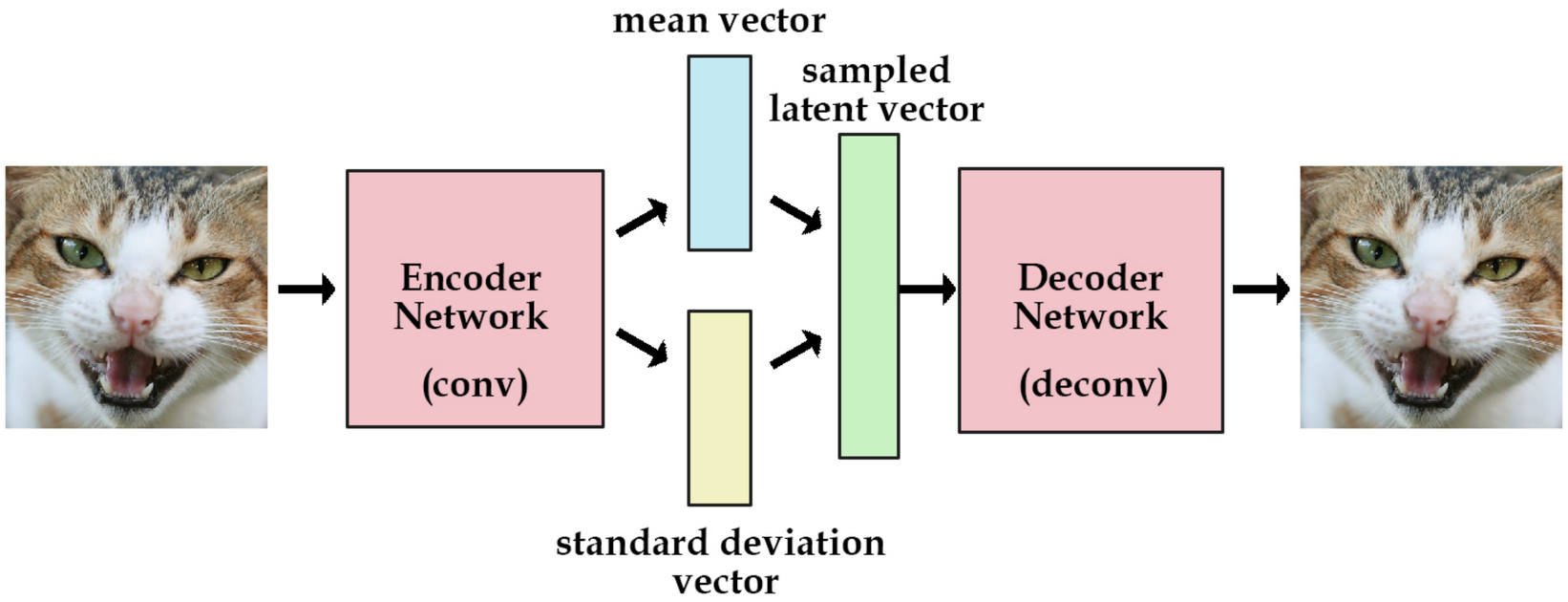


https://www.compthree.com/blog/autoencoder/

Variational Autoencoders

The autoencoders latent embedding/variables may be difficult to grasp if unconstrained

But we could constrain the latent space to follow a specific distribution, e.g. a **Variational Autoencoder**: “Auto-encoding Variational Bayes”, Kingma & Welling, ICLR 2014



https://kvfrans.com/content/images/2016/08/vae.jpg

Variational Autoencoders

More formally, let us consider:

- A dataset with variable **x**
- Data is generated by a random process involving unobserved random variable **z**
- **z** is generated from some prior distribution $p_\theta(z)$
- A value x is generated from some conditional distribution $p_\theta(x \mid z)$

Variational Autoencoders

More formally, let us consider:

- A dataset with variable **x**
- Data is generated by a random process involving unobserved random variable **z**
- **z** is generated from some prior distribution $p_\theta(z)$
- A value x is generated from some conditional distribution $p_\theta(x \mid z)$

But, the parameters and values of latent variables **z** are not known to us.

$$p_\theta(x) = \int p_\theta(x, z)dz \text{ is intractable}$$

VAE: ELBO derivation

1. The densities of the marginal and joint distribution are related through Bayes rule:

$$p_\theta(z \mid x) = \frac{p_\theta(x, z)}{p_\theta(x)} = \frac{p_\theta(x \mid z) p_\theta(z)}{p_\theta(x)}$$

VAE: ELBO derivation

1. The densities of the marginal and joint distribution are related through Bayes rule:

$$p_\theta(z \mid x) = \frac{p_\theta(x, z)}{p_\theta(x)} = \frac{p_\theta(x \mid z) p_\theta(z)}{p_\theta(x)}$$

2. Make use of the logarithm on both sides to write as a sum:

$$\log p_\theta(x) = \log p_\theta(x \mid z) + \log p_\theta(z) - \log p_\theta(z \mid x)$$

VAE: ELBO derivation

1. The densities of the marginal and joint distribution are related through **Bayes rule**:

$$p_\theta(z \mid x) = \frac{p_\theta(x, z)}{p_\theta(x)} = \frac{p_\theta(x \mid z) p_\theta(z)}{p_\theta(x)}$$

2. Make use of the **logarithm** on both sides to write as a sum:

$$\log p_\theta(x) = \log p_\theta(x \mid z) + \log p_\theta(z) - \log p_\theta(z \mid x)$$

3. We do not know our real posterior $p_\theta(z \mid x)$. We will make use of **variational inference** and introduce an **approximation** $q_\phi(z \mid x)$:

$$\log p_\theta(x) = \int q_\phi(z \mid x) \left[\log p_\theta(x \mid z) + \log p_\theta(z) - \log p_\theta(z \mid x) \right] dz$$

VAE: ELBO derivation

4. We add and subtract $q_\theta(z \mid x)$:

$$\log p_\theta(x) = \int q_\phi(z \mid x) \Big[\log p_\theta(x \mid z) + \log p_\theta(z) - \log p_\theta(z \mid x) + \log q_\phi(z \mid x) - \log q_\phi(z \mid x) \Big] \, dz$$

VAE: ELBO derivation

4. We add and subtract $q_\theta(z\,|\,x)$:

$$\log p_\theta(x) = \int q_\phi(z\,|\,x)\Big[\log p_\theta(x\,|\,z) + \log p_\theta(z) - \log p_\theta(z\,|\,x) + \log q_\phi(z\,|\,x) - \log q_\phi(z\,|\,x)\Big]\, dz$$

5. We make use of the (reverse) **Kullback Leibler divergence** between distributions:

$$KL(Q\,||\,P) = \int_{\infty}^{\infty} Q(x)\log\frac{Q(x)}{P(x)}$$

VAE: ELBO derivation

4. We add and subtract $q_\theta(z\,|\,x)$:

$$\log p_\theta(x) = \int q_\phi(z\,|\,x)\Big[\log p_\theta(x\,|\,z) + \log p_\theta(z) - \log p_\theta(z\,|\,x) + \log q_\phi(z\,|\,x) - \log q_\phi(z\,|\,x)\Big]\, dz$$

5. We make use of the (reverse) **Kullback Leibler divergence** between distributions:

$$KL(Q\,||\,P) = \int_{\infty}^{\infty} Q(x)\log\frac{Q(x)}{P(x)}$$

and replace the integral with an **expectation over samples**:

$$\log p_\theta(x) = KL\left[q_\theta(z\,|\,x)\,||\,p_\theta(z\,|\,x)\right] + \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x\,|\,z)\right] - KL\left[q_\phi(z\,|\,x)\,||\,p_\theta(z)\right]$$

VAE: ELBO derivation

$$\log p_\theta(x) = KL\left[q_\theta(z\,|\,x)\,||\,p_\theta(z\,|\,x)\right] + \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x\,|\,z)\right] - KL\left[q_\phi(z\,|\,x)\,||\,p_\theta(z)\right]$$

6. We still cannot evaluate the **first term** on the right hand side, but it is **strictly positive**

   -> we can optimize the remaining terms

$$\log p_\theta(x) = KL\left[q_\theta(z\,|\,x)\,||\,p_\theta(z\,|\,x)\right] + \mathcal{L}(\theta, \phi; x)$$

VAE: ELBO derivation

$$\log p_\theta(x) = KL\left[q_\theta(z\,|\,x)\,||\,p_\theta(z\,|\,x)\right] + \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x\,|\,z)\right] - KL\left[q_\phi(z\,|\,x)\,||\,p_\theta(z)\right]$$

6. We still cannot evaluate the **first term** on the right hand side, but it is **strictly positive**
   -> we can optimize the remaining terms

$$\log p_\theta(x) = KL\left[q_\theta(z\,|\,x)\,||\,p_\theta(z\,|\,x)\right] + \mathcal{L}(\theta,\phi;x)$$

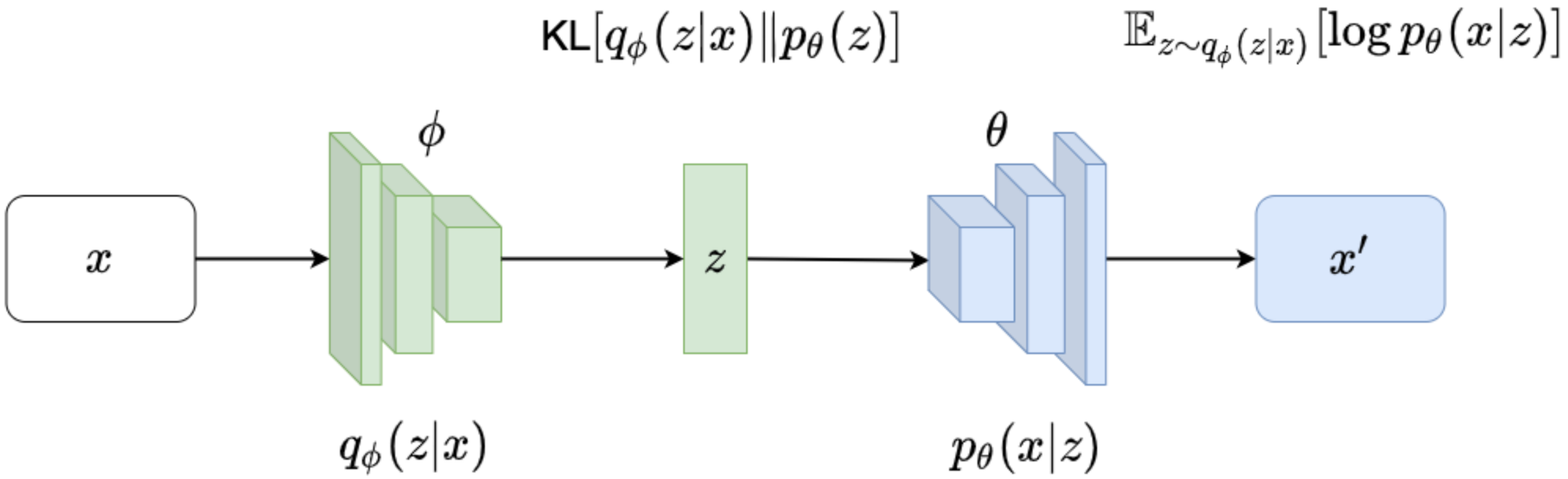
7. We now have a variational **lower-bound** to p(x) (the “evidence lower bound”, or ELBO)

$$\log p_\theta(x) \geq \mathcal{L}(\theta,\phi;x) = \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x\,|\,z)\right] - KL\left[q_\phi(z\,|\,x)\,||\,p_\theta(z)\right]$$

VAE: ELBO derivation

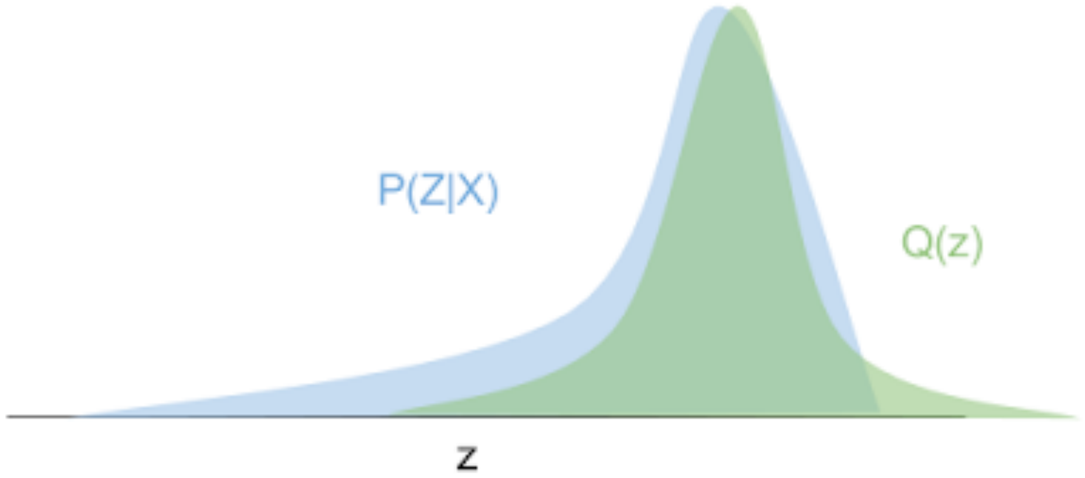
$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x \,|\, z) \right] - KL \left[q_\phi(z \,|\, x) \,||\, p_\theta(z) \right]$$

- The 1. term is the expected **reconstruction error** given by the log-likelihood (with sampling)
- The 2. term is a **KL divergence** encouraging the approximate posterior to be close to a prior



VAE: KL divergence

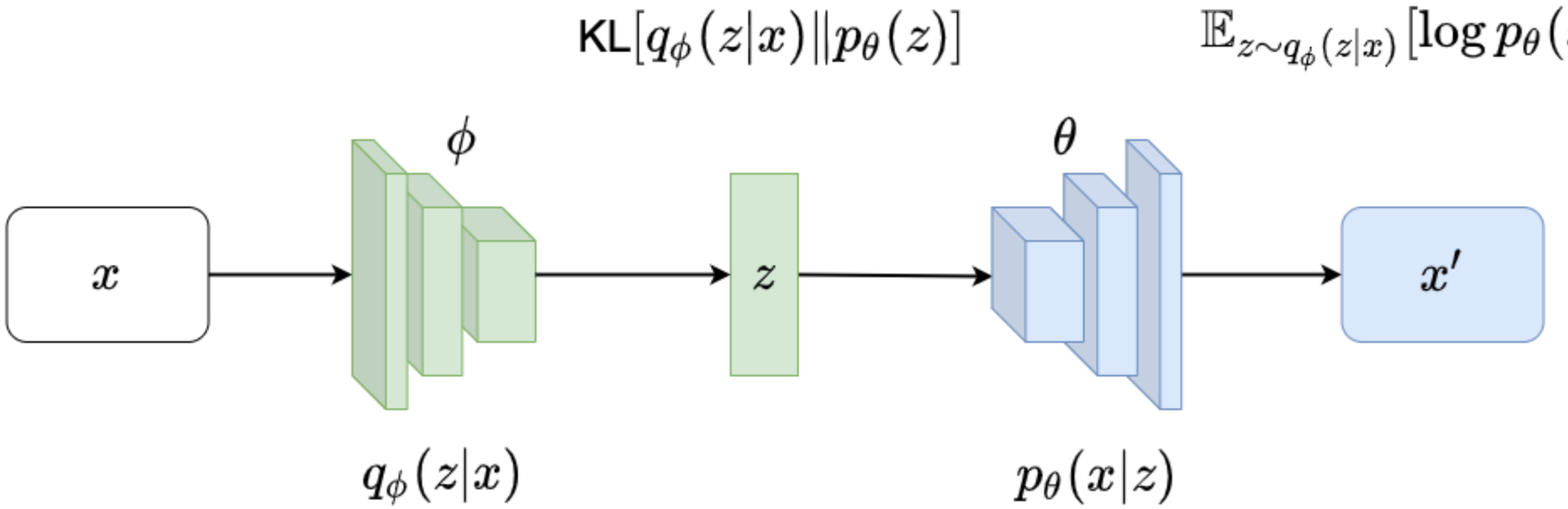
(Reverse) Kl divergence measures the **amount of information** (in nets, or units of 1/log2 bits) required to "**distort**" p(z) into q(z)



If you want to read more about variational inference and the subtleties of KL divergence: https://blog.evjang.com/2016/08/variational-bayes.html

VAE: summary

- The recognition model, a **probabilistic encoder**, $q_\phi(z \mid x)$ is a variational approximation to the intractable posterior $p_\theta(z \mid x)$ -> given a datapoint x it produces a distribution over possible values of z from which it could have been generated
- The **probabilistic decoder**, $p_\theta(x \mid z)$ produces a distribution over possible values of x given z

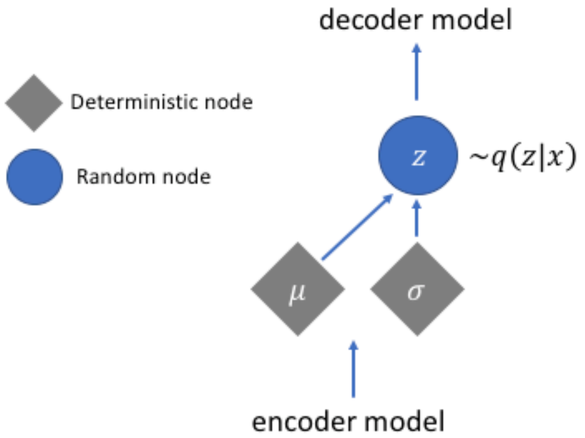


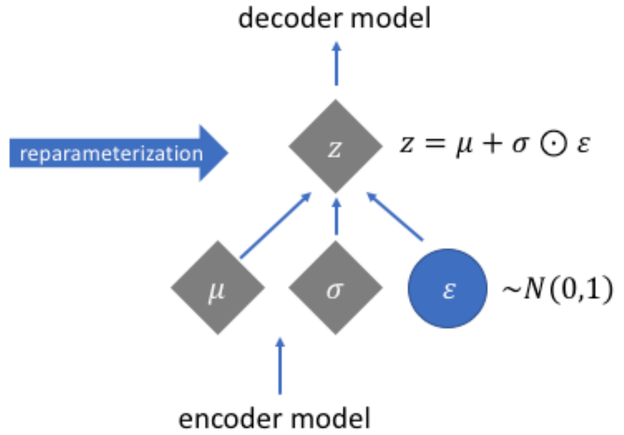
VAE: reparameterization

Example - approx. posterior as multivariate Gaussian:

$$\log q_\phi(z \mid x) = \log \mathcal{N}(z; \mu, \sigma I)$$

- Use **reparameterization trick** to generate samples from $q_\phi(z \mid x)$ with z as deterministic variable
- Sample posterior $z \sim q_\phi(z \mid x)$ using $z = \mu + \sigma \circ \varepsilon$ with **samples** $\varepsilon \sim \mathcal{N}(0, I)$ (as one choice of prior)

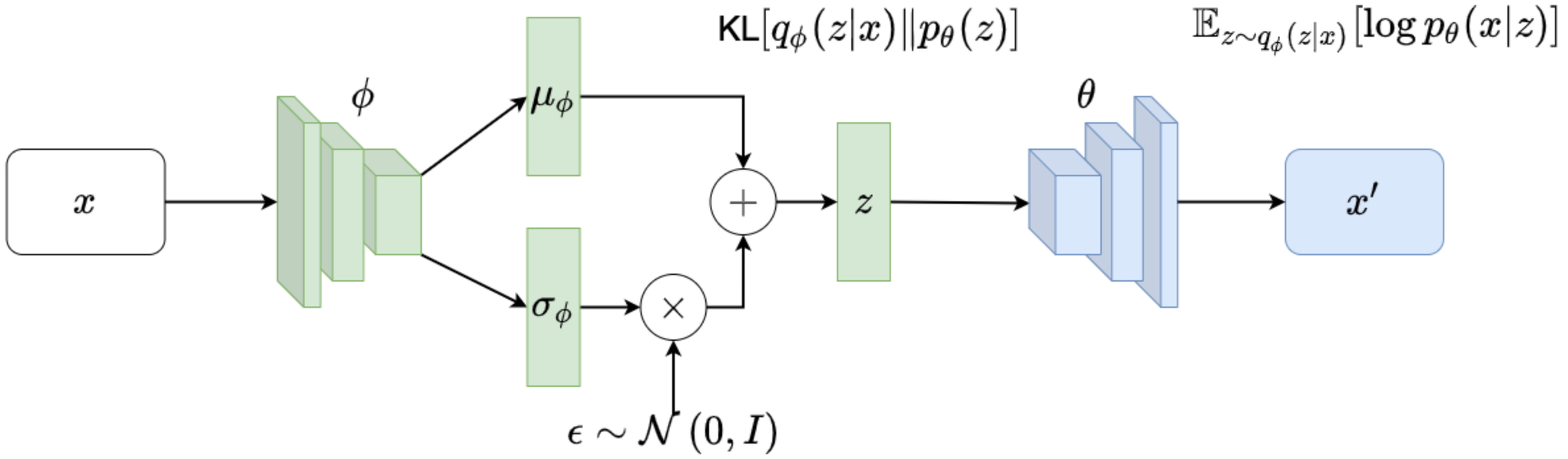




https://www.jeremyjordan.me/variational-autoencoders/

VAE: reparameterization

- Use **reparameterization** to generate samples from $q_\phi(z\,|\,x)$ with z as deterministic variable
- Sample posterior $z \sim q_\phi(z\,|\,x)$ using $z = \mu + \sigma \circ \varepsilon$ with **samples** $\varepsilon \sim \mathcal{N}(0,I)$ (as one choice)

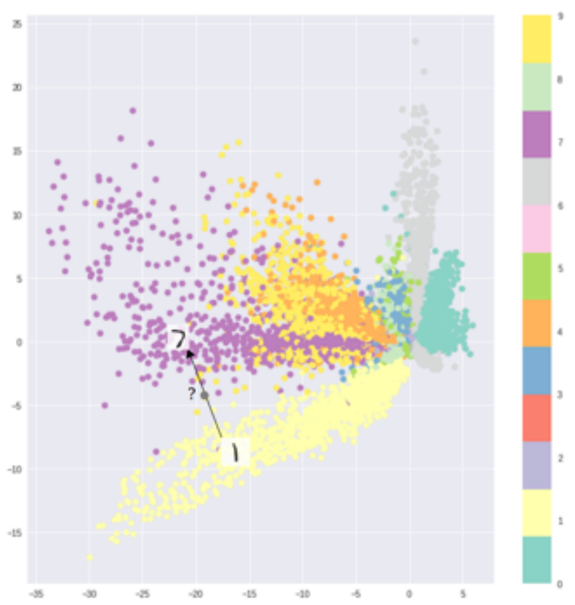


Why was this detour important: back to continual learning

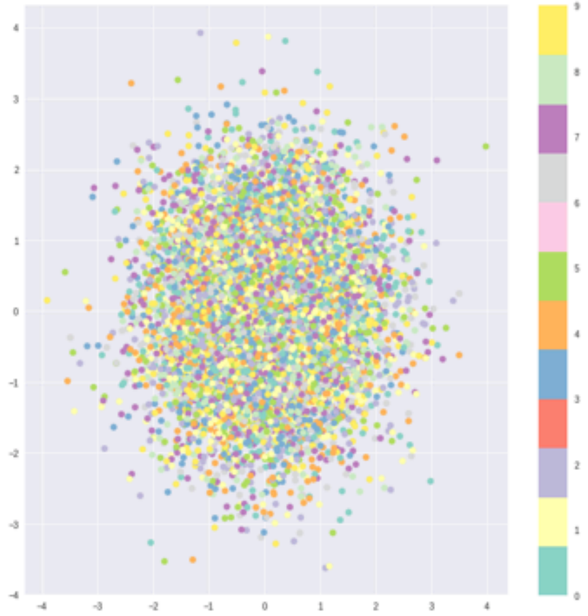
What have we gained?

Why did we go through all this math? And why is it important for continual learning?

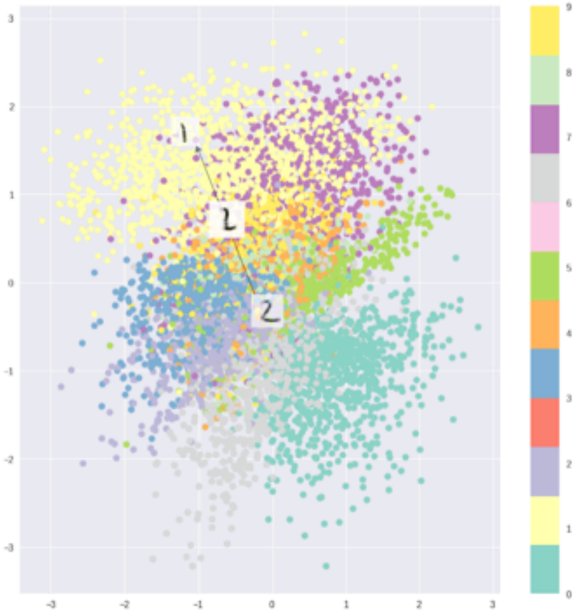
Only reconstruction loss



Only KL divergence



Combination

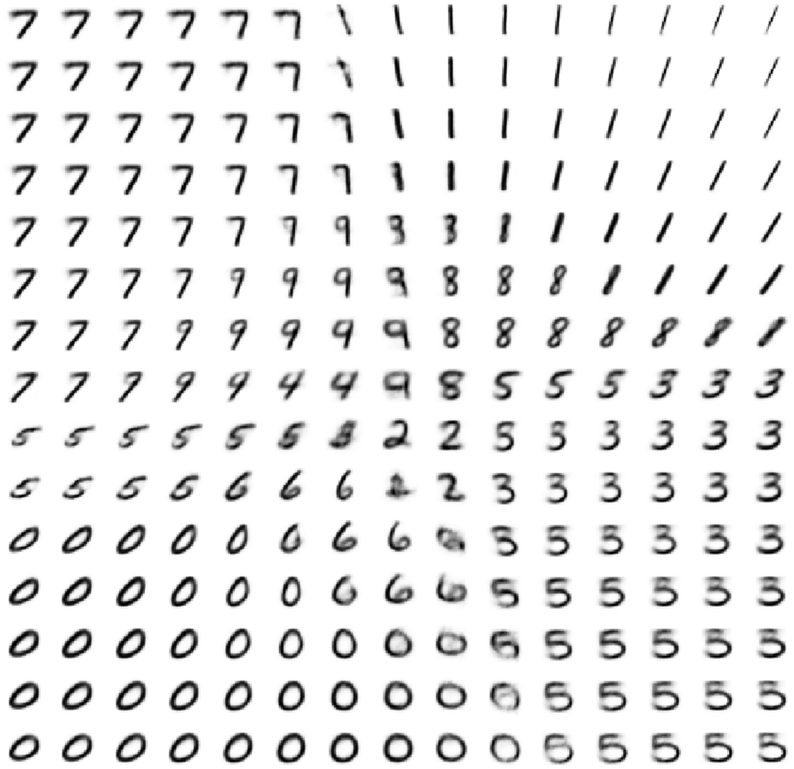


https://www.jeremyjordan.me/variational-autoencoders/ and https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

What have we gained?

Why did we go through all this math? And why is it important for continual learning?

- We can sample from a trained model: $z \sim p(z)$, here $\mathcal{N}(0,I)$, and generate (decode) x
- We also have the approximate posterior that we could regularize in continual learning



https://www.compthree.com/blog/autoencoder/

Variational Continual Learning

$$\mathscr{L}(\theta, \phi; x) = \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x \,|\, z) \right] - KL \left[q_\phi(z \,|\, x) \,||\, p_\theta(z) \right]$$

The "likelihood focused" perspective:
generative/pseudo rehearsal

- Generate old tasks' data and
  concatenate it with new task data
- Primarily optimize "the likelihood" (left)

See Nguyen et al, "Variational Continual Learning" ICLR 2018 & follow-ups like Farquhar et al "A Unifying Bayesian View of Continual Learning", NeurIPS workshops 2018

Variational Continual Learning

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x \,|\, z) \right] - KL \left[q_\phi(z \,|\, x) \,||\, p_\theta(z) \right]$$

The "likelihood focused" perspective:
generative/pseudo rehearsal

- Generate old tasks' data and concatenate it with new task data
- Primarily optimize "the likelihood" (left)

The "prior focused" perspective:
regularization/distillation

- Only use new task data
- Use the posterior of an old task as the new task's prior $KL \left[q_t(z) \,||\, q_{t-1}(z) \right]$

See Nguyen et al, "Variational Continual Learning" ICLR 2018 & follow-ups like Farquhar et al "A Unifying Bayesian View of Continual Learning", NeurIPS workshops 2018

Variational Continual Learning

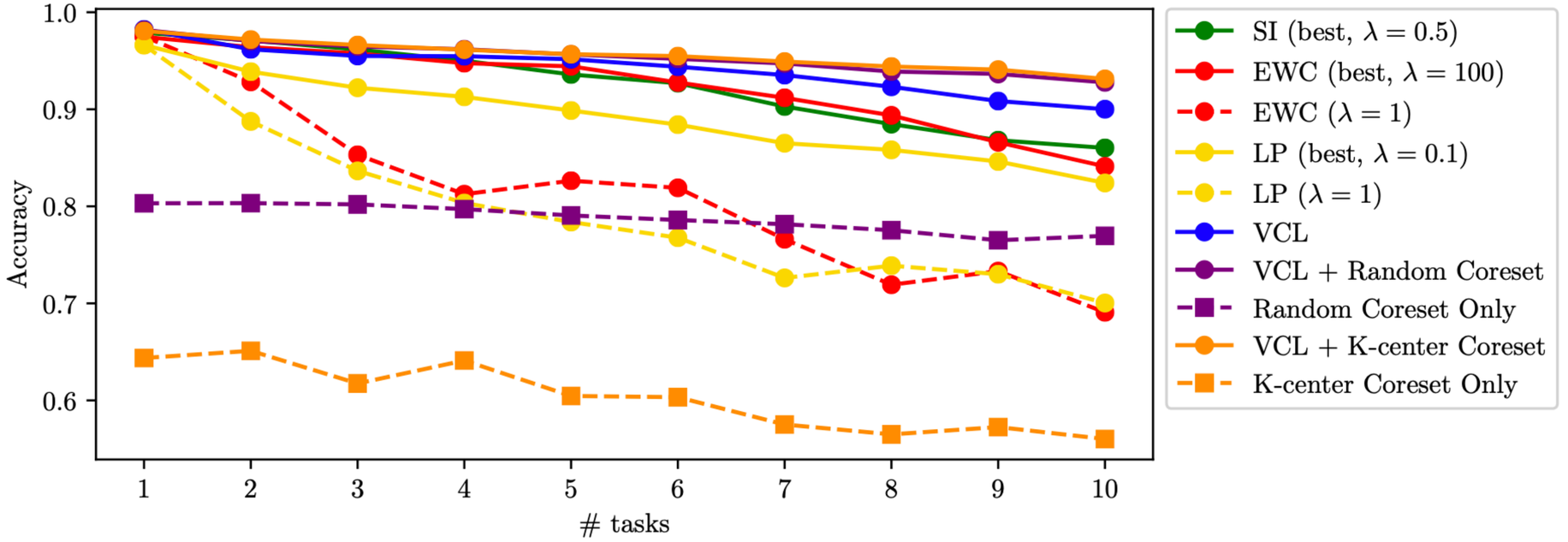


Figure 2: Average test set accuracy on all observed tasks in the Permuted MNIST experiment.

Nguyen et al, “Variational Continual Learning” ICLR 2018

Variational Continual Learning

Optionally also store real data subsets (or a core set)

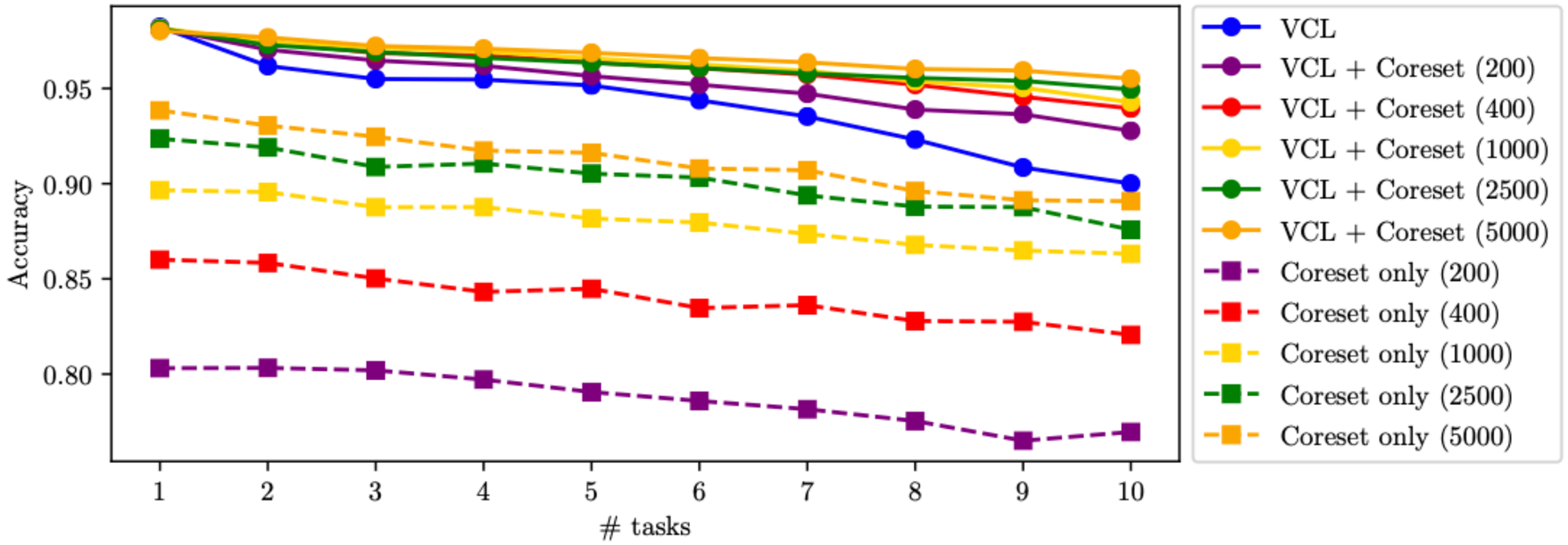


Figure 3: Comparison of the effect of coreset sizes in the Permuted MNIST experiment.

Nguyen et al, “Variational Continual Learning” ICLR 2018

Formally: core sets

What is a core set?

The term core set is often loosely employed in modern literature to be synonymous to exemplars and sub sets of data

"*coresets are small, (weighted) summaries of large data sets such that solutions found on the summary itself are* ***provably competitive*** *with solutions found on the full data set*"

$$|\texttt{cost}(P, Q) - \texttt{cost}(C, Q)| \leq \varepsilon \cdot \texttt{cost}(P, Q)$$

Note how this is specific to some data, a set of questions/queries, models + loss/cost functions

Good introductions are: Bachem et al, "Practical Coreset Constructions for Machine Learning" (2017) or Jubran et al, "Introduction to Coresets: Accurate Coresets" (2019)

Picking a “core set”

Why could we potentially pick better data subsets/exemplars now?

- Example of a 2-D latent space with 4 classes/clusters
- Random or k-means (depending on the amount of k) may not mirror the distribution well

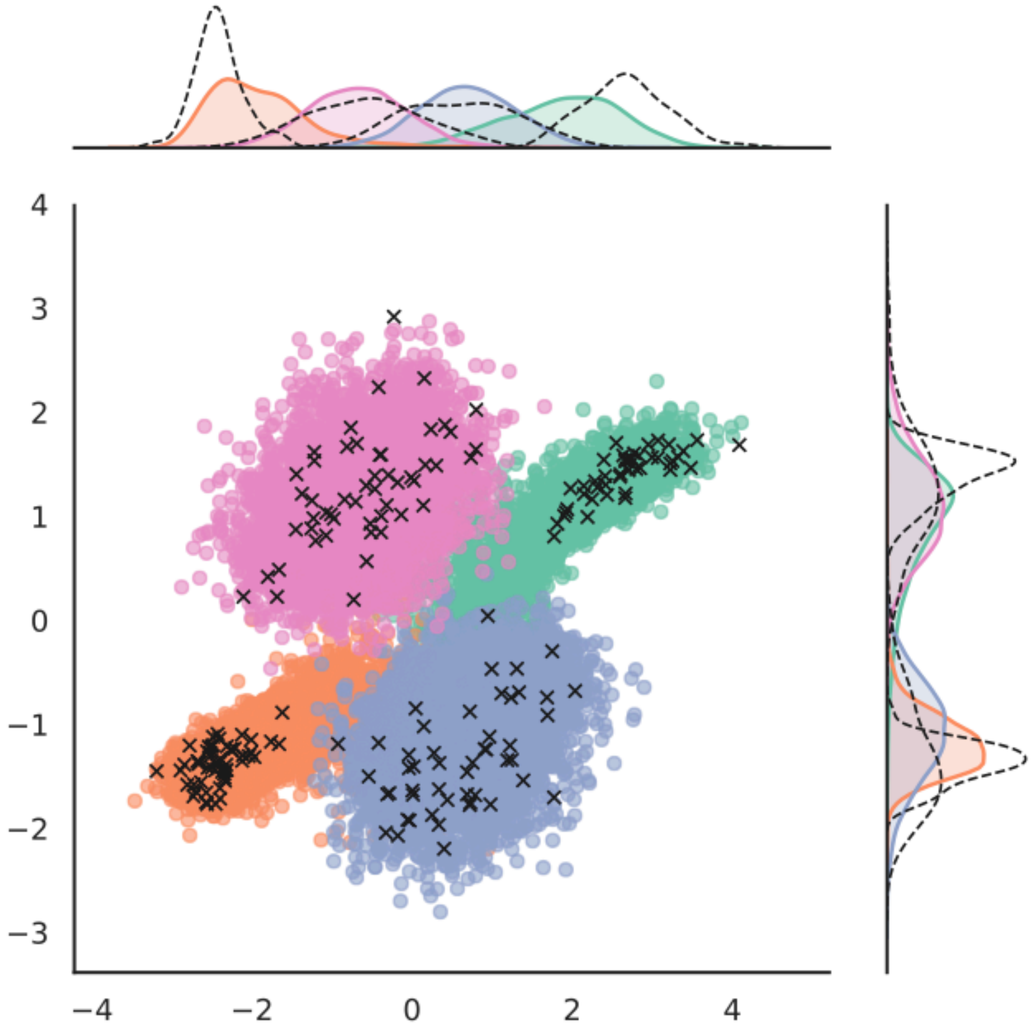


Figure from “A Wholistic View of Deep Neural Networks: Forgotten Lessons and the Bridge to Active and Open World Learning”, Mundt et al 2020

Picking a “core set”

Why could we potentially pick better data subsets/exemplars now?

- If we know our approx. posterior, we could sample and pick instances that lie close the samples
- (It’s not actually that easy, for various reasons, but the intuition is that we are somewhat aware of p(x) now)

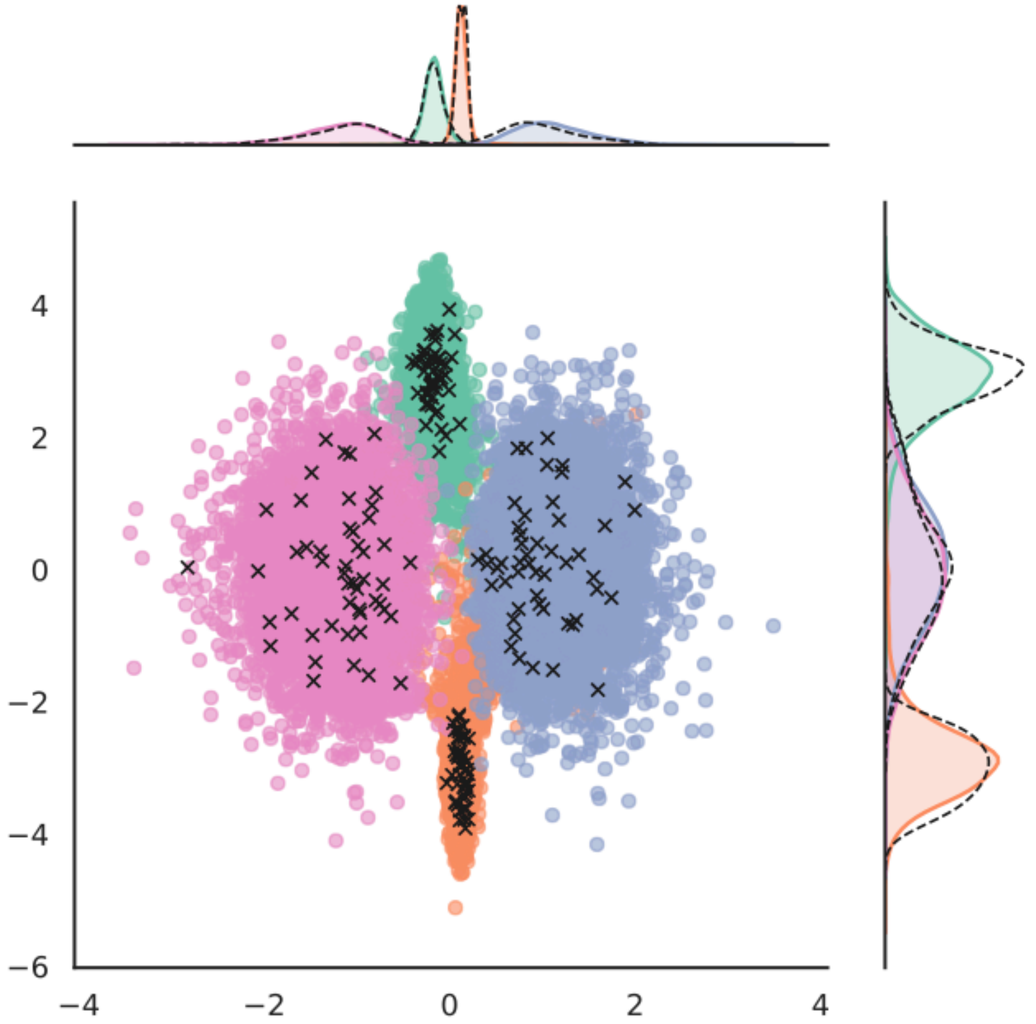


Figure from “A Wholistic View of Deep Neural Networks: Forgotten Lessons and the Bridge to Active and Open World Learning”, Mundt et al 2020

CURL: task specific Gaussians

We could now also use task-specific priors more explicitly

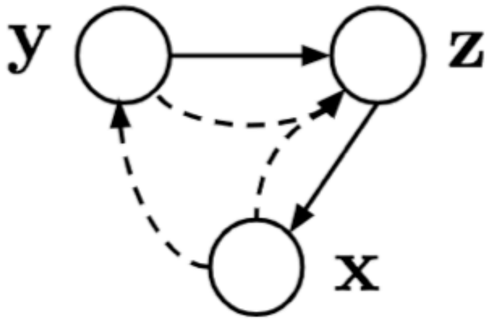


Figure 1: Graphical model for CURL. The categorical task variable $\mathbf{y}$ is used to instantiate a latent mixture-of-Gaussians $\mathbf{z}$, which is then decoded to $\mathbf{x}$.

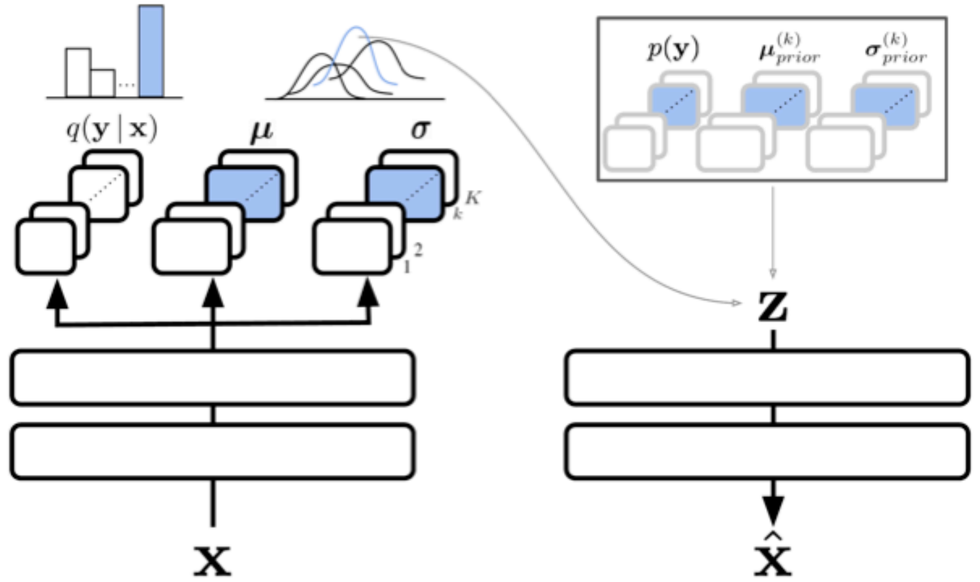


Figure 2: Diagram of the proposed approach, showing the inference procedure and architectural components used.

Rao et al, “Continual Unsupervised Representation Learning”, NeurIPS 2019

Have we solved forgetting? Why are we not done?

Combining ideas

Why may we need these multiple perspectives?

Generators also suffer from forgetting, errors can “snowball” rapidly

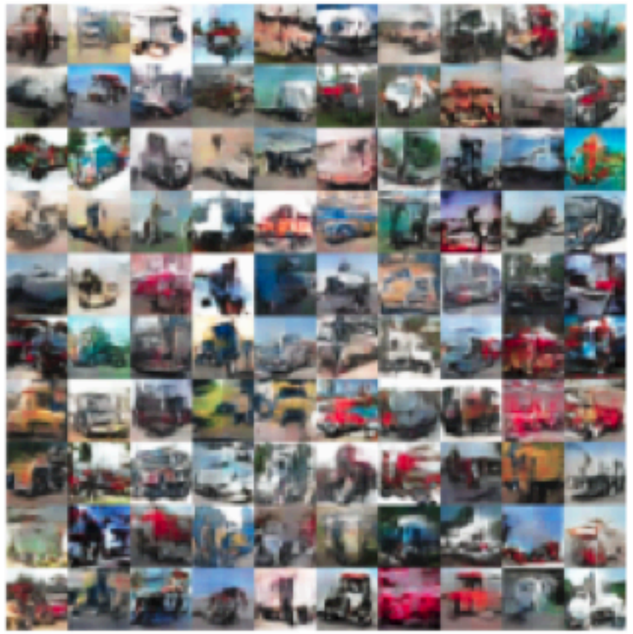
Lesort et al, “Generative Models from the perspective of Continual Learning”, IJCNN 2019

Combining ideas

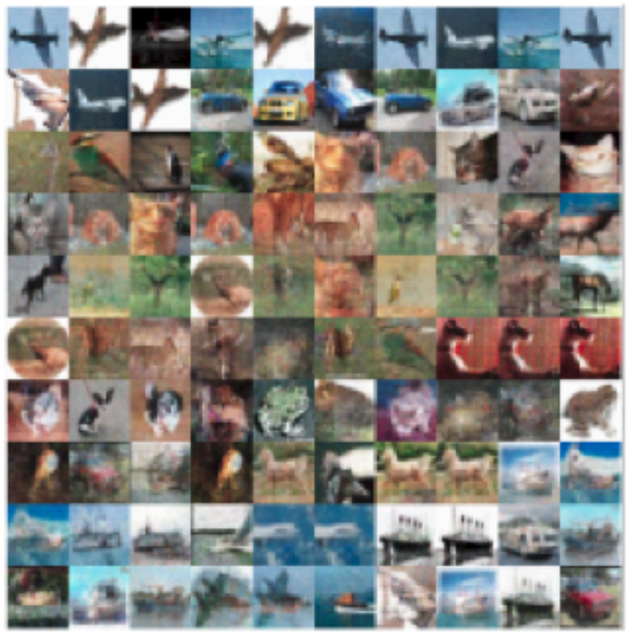
Why may we need these multiple perspectives?

Fine-tuning alone forgets, regularization is a trade-off and data rehearsal can overfit

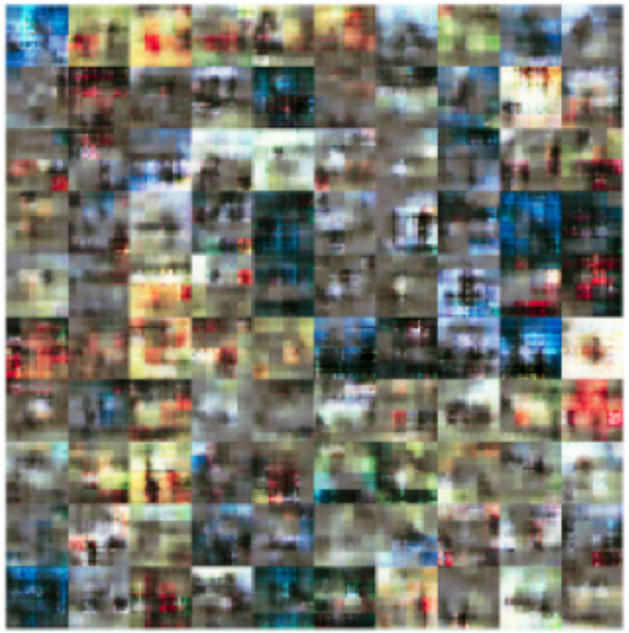
Fine-tuning



Rehearsal



Generative Replay



Lesort et al, “Generative Models from the perspective of Continual Learning”, IJCNN 2019

In summary: what do we want?

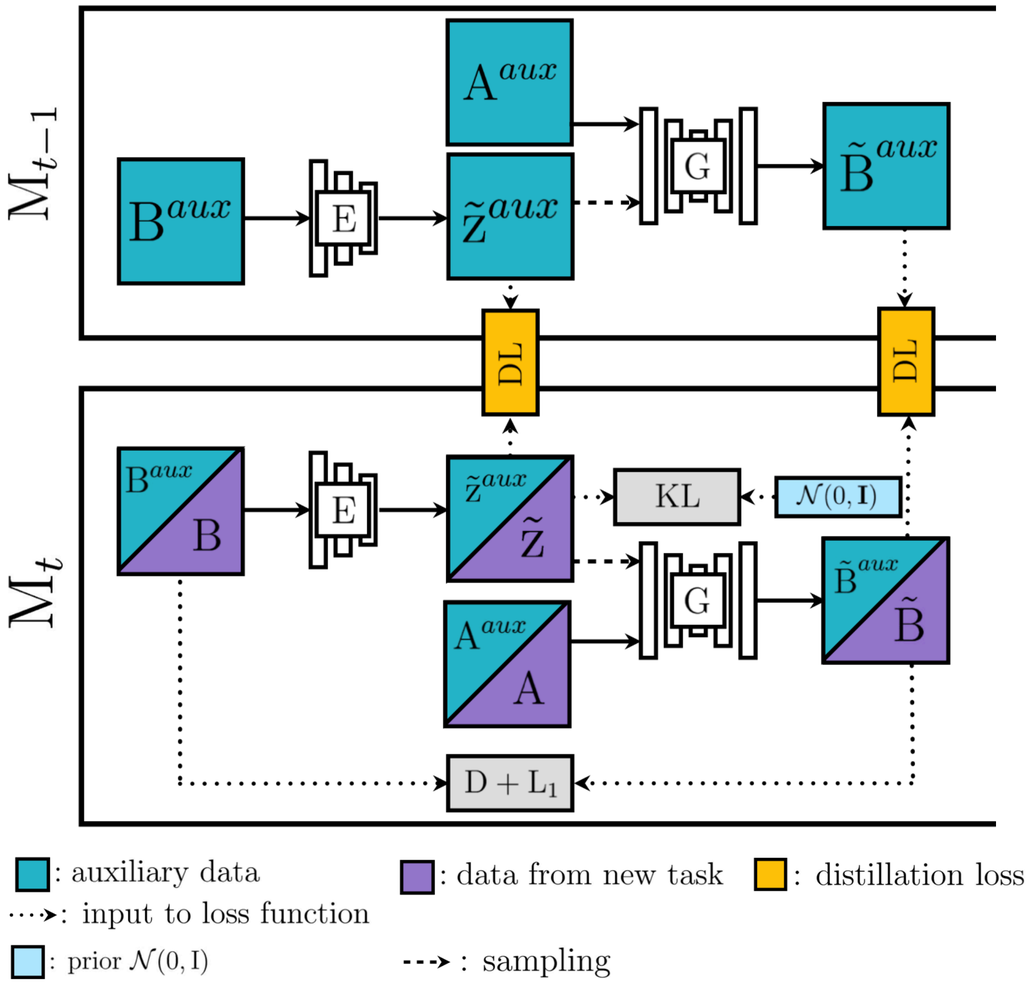
What could our expectations be, what might we desire?

In summary: what do we want?

What could our expectations be, what might we desire?

- Constant memory budget?
- Pragmatically? A selection algorithm that outperforms randomly stored data points?
- A way to shrink the memory buffer to add new tasks, e.g. recursively select exemplars?
- Ideally? Knowledge of the distribution(s) and a subset with approximation guarantees?
- A natural formulation to allow (pseudo-)rehearsal, regularization…?
- …. many more …?

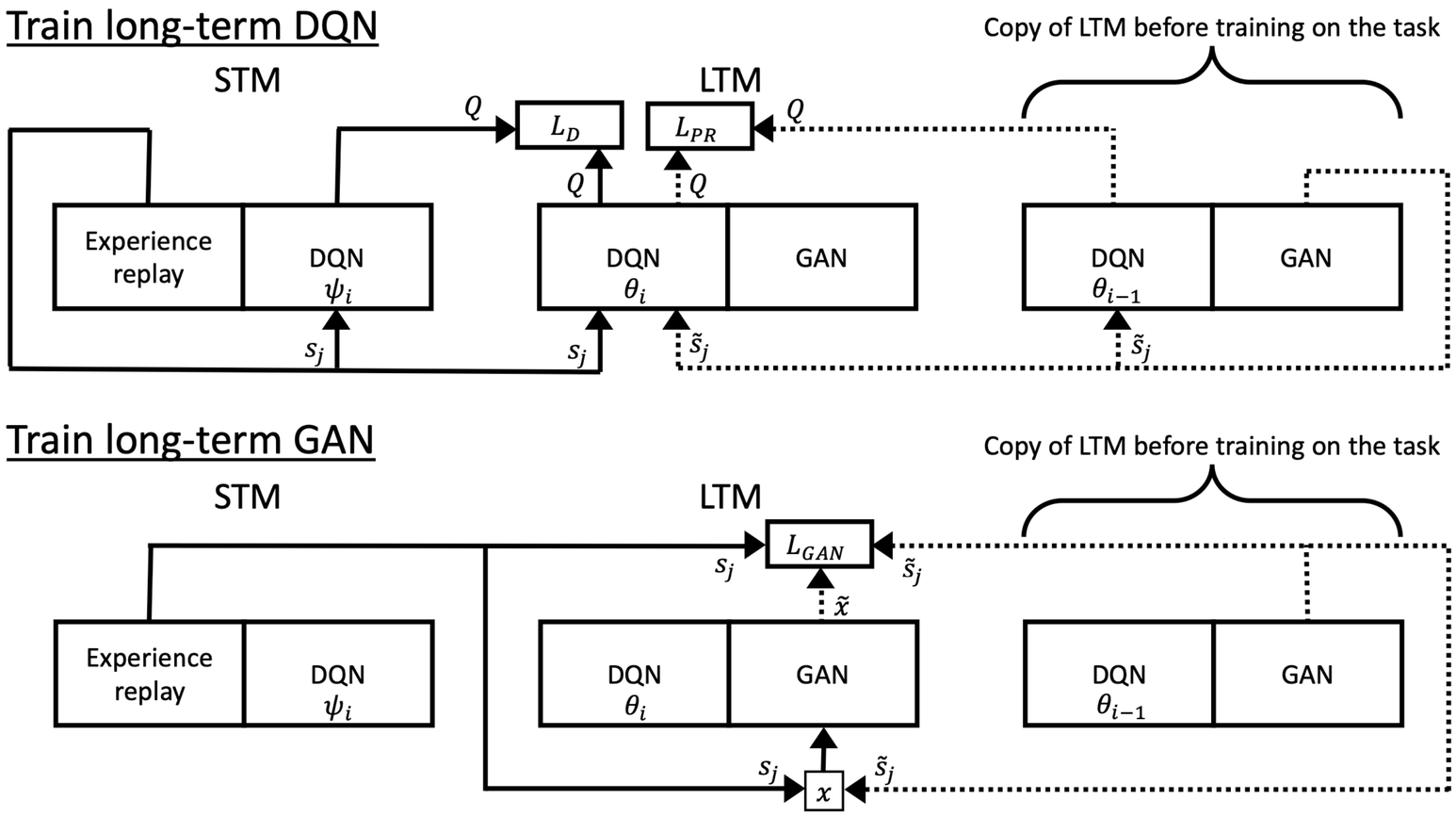
Combining ideas



- We can (naturally) combine ideas
- In most of what we have seen so far however, the “architecture” has remained static.
- The assumption that we can continue accumulate knowledge because we have enough capacity has been prevalent.

Zhai & Chen et al, “Lifelong GAN: Continual Learning for Conditional Image Generation”, ICCV 2019

Combining ideas



Disclaimer

- There are A LOT of works that use rehearsal, which are not mentioned here
- They also come in all shapes of supervised/ unsupervised/reinforced for different modalities

Atkinson, “Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting”, Neurocomputing 428:7, 2021