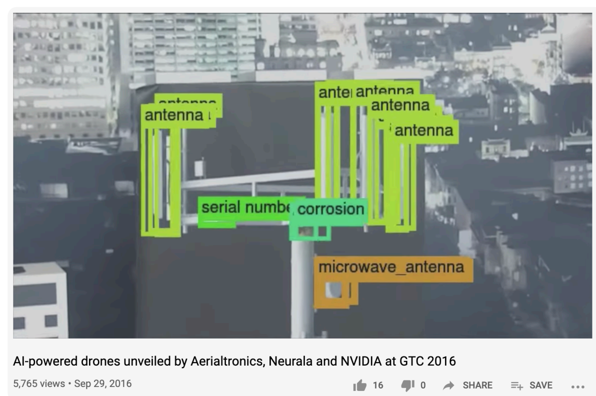
Continual Learning and Catastrophic Forgetting

by Paul Hand
Northeastern University

Outline:

- Context + initial approaches
- Evaluating algorithms
- Algorithms for CL

Example context for continual learning



Other example: autonomous vehicles

Can you simply train on new data?

Task A: $D_A = \{(x_i, y_i)\}$ Task B: $D_B = \{(x_i, y_i)\}$

First,

$$\min_{\theta} \sum_{x_i, y_i \in D_A} L(\hat{y}_\theta(x_i), y_i)$$ initialize randomly

Then,

$$\min_{\theta} \sum_{x_i, y_i \in D_B} L(\hat{y}_\theta(x_i), y_i)$$ initialize w/ soln to above task

Failure mode: catastrophic forgetting / interference

Typically, good performance at B
worse performance at A

How can you mitigate forgetting?

Train from scratch w/ new data and old data

Drawbacks

- Many industry nets take days – weeks to train
- Waste of power and compute
- Original training data may be unavailable
- Need access to GPUs / Cloud / steady internet

Humans can learn incrementally, so it is possible to do

Replay old training data w/ new data

- requires storing old data
- Storage costs grow linearly w/ tasks

Dilemma: plasticity – stability

Reviews: Parisi et al. 2019
Chen and Liu 2018

Evaluating Continual Learning Algorithms

(Kemker et al. 2017)

Data permutation tasks

img   class 0–9

Task 1: Training data – MNIST (x_i, y_i)
Given x_i, predict y_i

Task 2: Fix an image permutation P_2

Training data – MNIST $(P_2 x_i, y_i)$
Given $P_2 x_i$, predict y_i

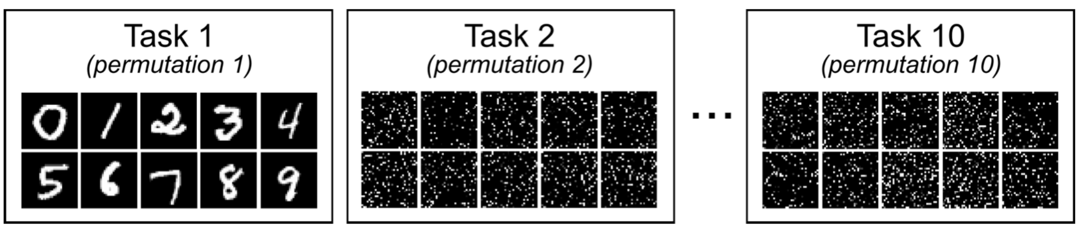


Figure 2: Schematic of permuted MNIST task protocol.

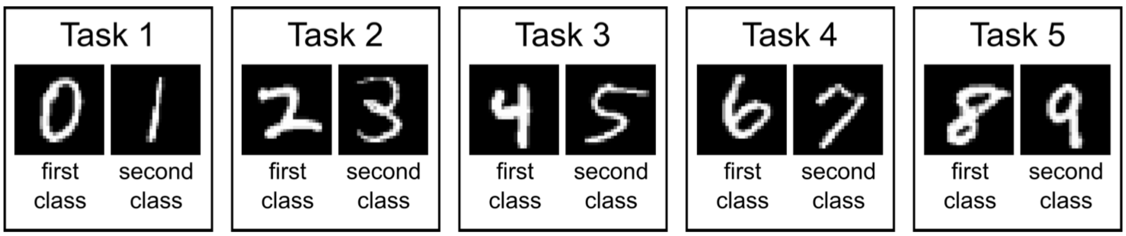
(Van de Ven and Tolias 2019)

Comments:

Each task is equally difficult

Incremental Class learning

Learn a base task set, then learn additional classes

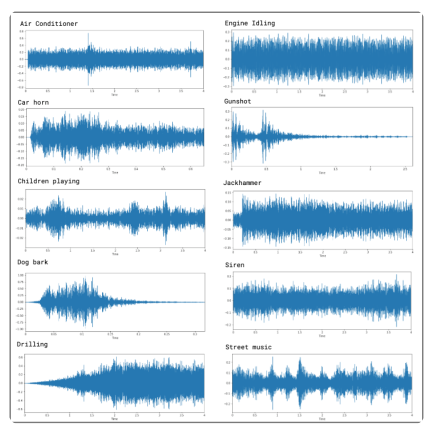


(Van de Ven and Tolias 2019)

Shared features w/ new classes

Multimodal learning

Learn an image classification task then learn audio classification



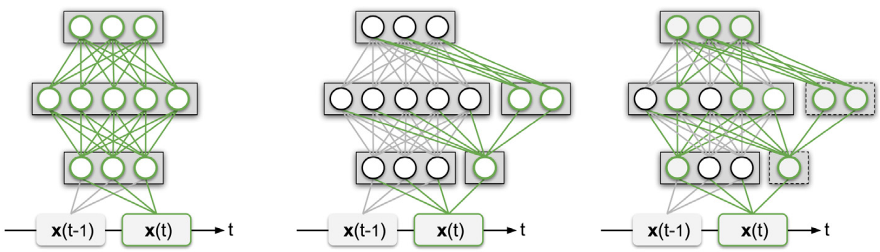
Different features must be learned

Approaches for continual learning

- Train whole net w/ regularization
- Dynamic architectures (add neurons)
- Complementary Learning Systems (memory + replay)

(and more)

G.I. Parisi, R. Kemker, J.L. Part et al. / Neural Networks 113 (2019) 54–71



a) Retraining with regularization

b) Training with network expansion

c) Selective network retraining and expansion

$\underbrace{\text{a)}}_{\text{regularization}}$ $\underbrace{\text{b) c)}}_{\text{dynamic architectures}}$

Regularization approaches

Update network weights but penalize changes in order to minimize forgetting

(Li + Hoiem 2018)

Learning without Forgetting (LwF)

Consider predictor with shared parameters across tasks and some task specific parameters

At new task, update:

shared params, new params, AND old params

So that

output of old task on new data doesn't change too much.

LEARNINGWITHOUTFORGETTING:

Start with:

θ_s: shared parameters

θ_o: task specific parameters for each old task

X_n, Y_n: training data and ground truth on the new task

Initialize:

$Y_o \leftarrow \mathrm{CNN}(X_n, \theta_s, \theta_o)$ // *compute output of old tasks for new data*

$\theta_n \leftarrow \mathrm{RANDINIT}(|\theta_n|)$ // *randomly initialize new parameters*

Train:

Define $\hat{Y}_o \equiv \mathrm{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_o)$ // *old task output*

Define $\hat{Y}_n \equiv \mathrm{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_n)$ // *new task output*

$$\theta_s^*,\ \theta_o^*,\ \theta_n^* \leftarrow \underset{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n}{\mathrm{argmin}} \left(\lambda_o \mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \right)$$

parameter for plasticity-stability

modified cross-entropy loss

cross-entropy loss

weight decay

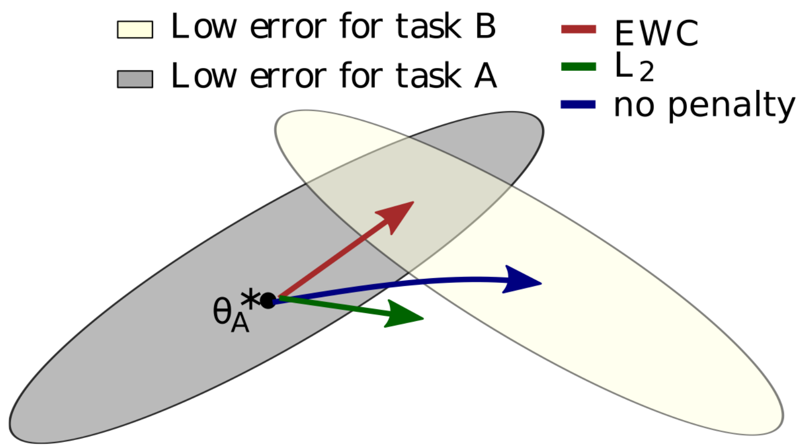
Note: does not require seeing old data!

Elastic Weight Consolidation (EWC)

(Kirkpatrick et al. 2017)

When training on task B, identify weights that were important to A and penalize updates to those weights

Try to stay in low error region for A



Minimize

$$L(\theta) = L_B(\theta) + \sum_i \frac{\lambda}{2} F_i (\theta_i - \theta^*_{A,i})^2$$

parameter

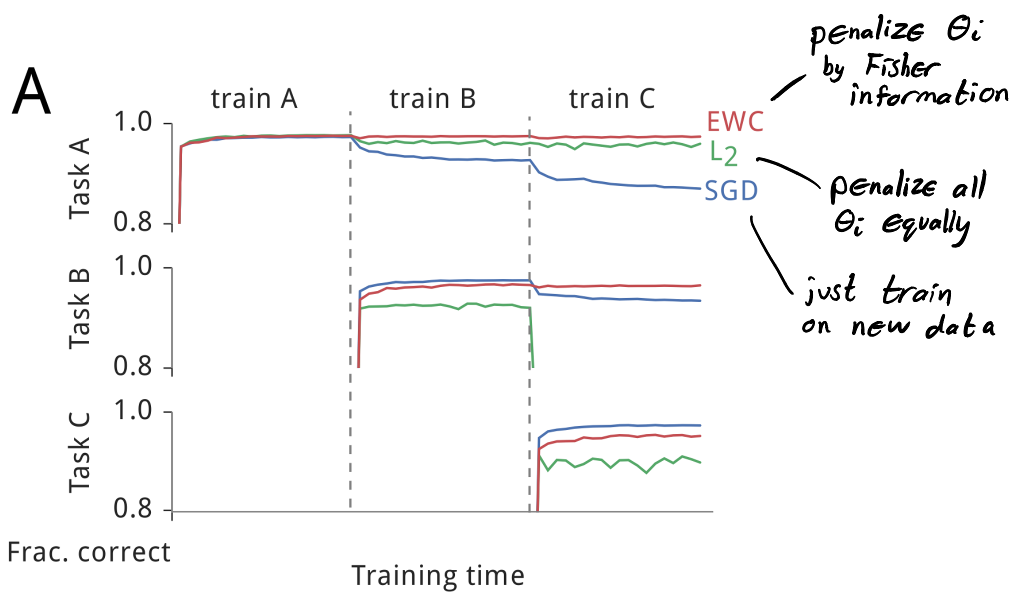
Solution to task A

where

$$F_i = \mathbb{E}_x \left(\partial_{\theta_i} \log p_\theta(x) \right)^2$$

diagonal entry of Fisher information matrix of predictor at θ_A^*

Motivated by a Bayesian learning perspective

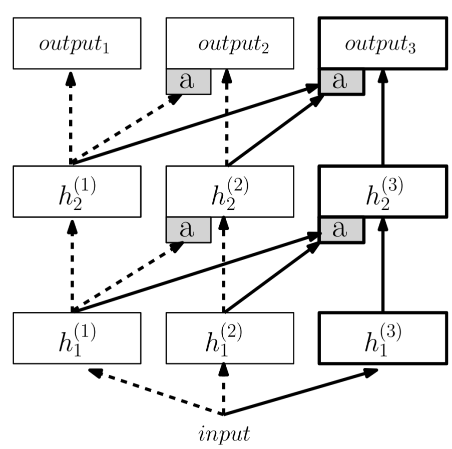


Progressive Neural Networks

(Rusu et al. 2016)

For each new task,

- add neurons
- add output layer
- add lateral connections
- dont modify old weights

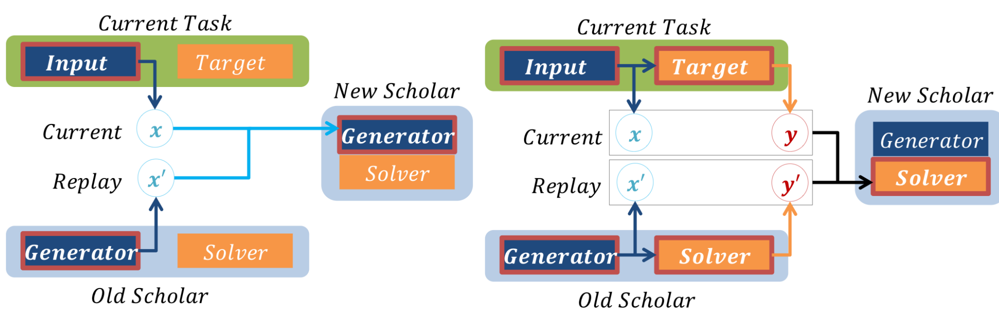


Generative Replay

(Shin et al. 2017)

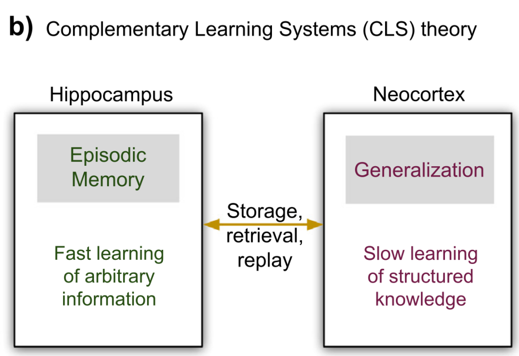
Train a generative model to output synthetic data that follows same distribution as training data.

Replay synthetic data along w/ new data



(b) Training Generator (c) Training Solver

Takes inspiration from human learning



(Parisi et al. 2019)