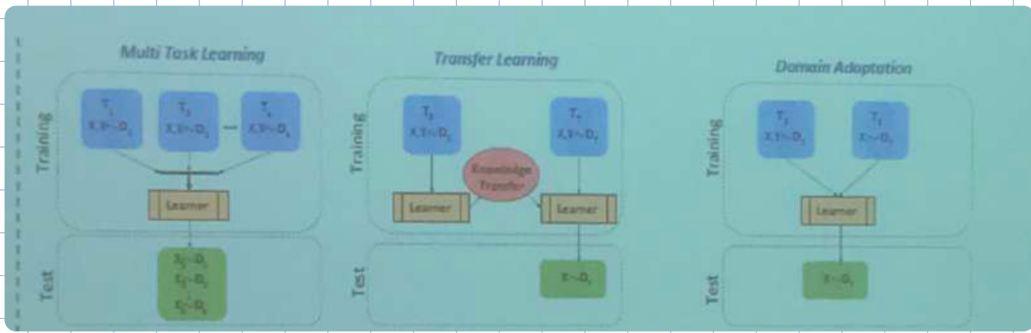
EEML - Day 3 Bucharest

03/07/19

Continual Learning - Tinne Tuytelaars (KU Leuven)

$T_1 \rightarrow T_2 — \dots — T_N$

- Continual = Lifelong = Incremental = Never ending



- Multi-task ⇒ One model → multiple tasks: Each task acts as form of regularizer for the others → auxiliary tasks help a lot if dbps shallow overlap
- Transfer ⇒ Leverage source task insights to solve the target task → From pretrained
- Domain Adaptation ⇒ Single task → shift in distr. ⇒ sometimes no labels for target domain!
  ↳ two losses: task loss + min. repr. loss between domains
  ↳ adversarial loss: model to not be able to distinguish between domains
- Meta-Learning ⇒ Task distribution Few Shot Strategy! → L-to-L
  ↳ At test time apply meta-learner to specific test task ⇒ Initialization
- Online Learning ⇒ Streaming + No train/test time split! ⇄
  ↳ Problem: Samples are not drawn from iid fashion! ⇒ overfits most recent data!
- Continual learning: Switch in tasks → Not all data available at train time

↳ Learn one task after other ⇒ No storing of prev. data
⇒ No large memory footprint ⇒ No forgetting ⇒ Have to relax!

① Regularization-based ⇒ data vs. model
② Rehearsal / Replay-based
③ Network Architecture-based

① REGULARIZATION ⇒ Implicit Schedulers

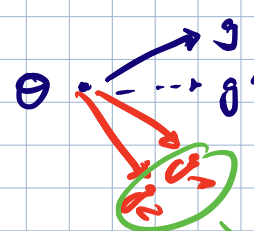
- Knowledge distillation loss ⇒ preservation of responses
  ↳ works well for related tasks — Li & Hoiem (ECCV, 16)
  ↳ Memory: store old model vs. store old predictions
- Penalize changes to important params ⇒ Prior!

$$T_n: \min_{\theta} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(y_{n}, f(x_{n}, \theta^{n})) + \lambda_{\Omega} \sum_{k} \Omega_k (\theta_k^{n} - \theta_k^{n-1})^2$$

  ↳ flexibility vs stability → How to estimate importance weights?
  ↳ Gradients: Elastic Weight Consolidation → loss effect
  ↳ Multiple tasks: Don't add prior per task but find a specific representative one → memory consideration
- Memory-aware synapses: ensure that output does not change too much in parts of input space!
- Synaptic intelligence → Zenke et al (ICML, 2018)

② REHEARSAL ⇒ Representative Memory Buffer

- iCARL (Rebuffi et al, 2017) ⇒ select samples closest to mean of each class
  ↳ knowledge distillation loss old + new data
- Gradient Episodic Memory (GEM) ⇒ Constrain gradient to improve previous tasks
  ↳ Focus on transfer backward/forward!



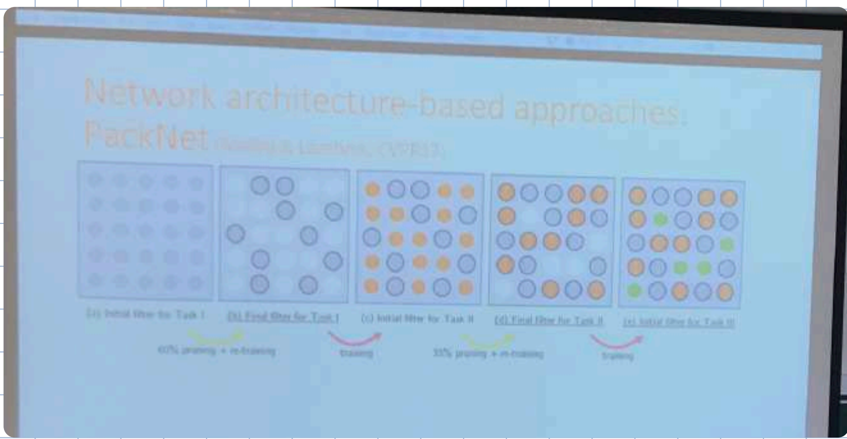
$\boxed{g' \cdot g_1 \geq 0 \quad g' \cdot g_2 \geq 0}$ → project gradient!

- How many exemplars?
  ↳ Fixed # per task (GEM) → need enough exemplars to get good estimate!

↳ Fixed memory (iCARL) ⇒ adapt / change later on

③ ARCHITECTURE ⇒ Explicit Subnetworks

– PackNet (Mallya & Lazebnik, CVPR 17)



↳ Prune + retrain on sparse per task ⇒ freeze
⇓
Mask Task 1

↳ Compression + Adaptation

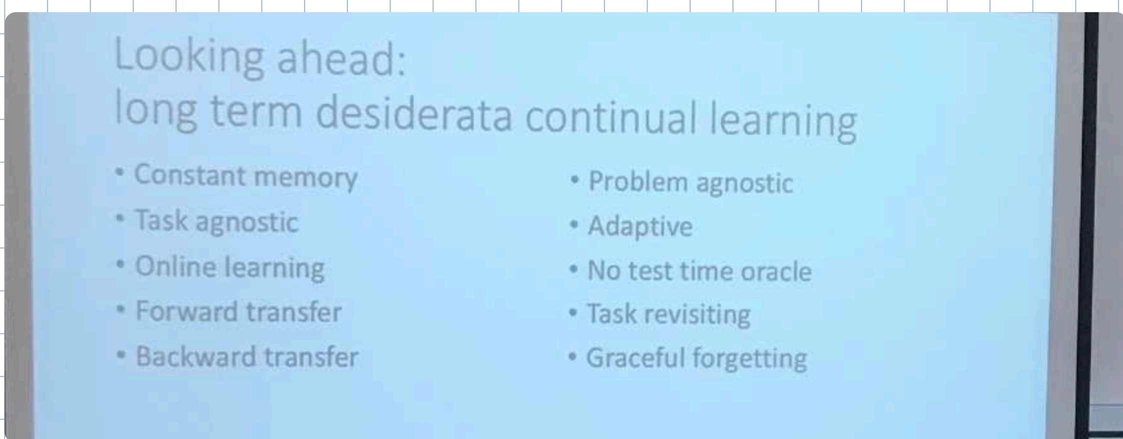
↳ Guaranteed no forgetting!

↳ Need to know # tasks

↳ How to choose how much to prune? ⇒ Open Question!

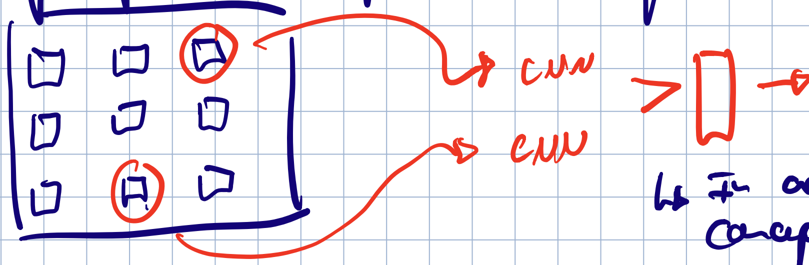
COMPARISON – INSIGHTS

- Importance of hyperparameters: Stability vs. Plasticity
- Tiny ImageNet ⇒ Good baseline dataset!
- PackNet implies no forgetting!
- MAS were robust than EWC
- Order of tasks does not matter too much → First tasks needs to be representative ⇒ Otherwise curriculum seems not to matter
- Large models ⇔ More capacity ↑ faster with the data



Self-Supervised Learning – Andrew Zisserman

- Self-Supervision = Supervision comes from the data → **Proxy tasks!**



→ Predict relative positioning to each other

↳ In order to solve you need conceptual knowledge

↳ Goal: "ImageNet"-like features without supervision

- FROM IMAGES

→ PASCAL VOC ⇒ Learn embedding unsupervised and test on downstream task!

→ Problem of network learning to cheat!

↳ Chromatic Abberation → Cable from lens! ⇒ Use only one color channel!

→ Loss: Coloring → Give grey and let network output colored image

→ Exemplar Networks → Classify many different transfos of one patch as same patch ↳ learn invariance!

→ Multi-Task Self-Supervised Training

↳ Features fed into classifier that is then trained supervised

→ Image Transformations ⇒ Predict rotations

→ Jigsaw Patch Puzzle ⇒ Noroozi & Favaro, 2016

↳ 9 patch ordering / permutation

↳ Can control complexity of predictions: How many colors to predict? // How many patches to permute?

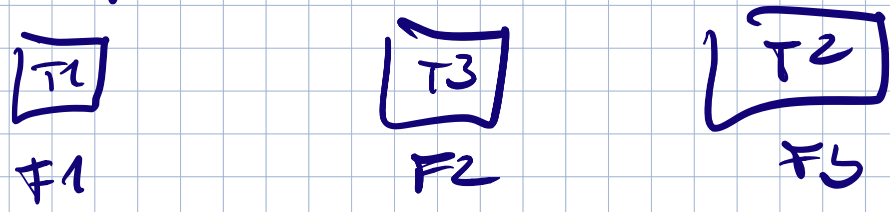
→ Benefits from more data + sequentially more complex problem!

→ Think about in terms of Meta-Learning ⇒ Weight Init!

- **FROM VIDEOS**

→ Strong correlation in time ⇒ Order / direction / tracking

→ "Shuffle & Learn" ⇒ Network does binary decision



↳ Same video: there has to be info

↳ Important: Need to give net a hard set of examples ⇒ form of preprocessing to design examples

↳ human motion: Need to learn landmarks to detect action

→ Order prediction ⇒ basically sort → harder than binary

→ Coldplay ⇒ 'Arrow of time' videos → Predict flow!

→ Again be careful with details:

↳ Zoom in the videos

↳ often camera angle moves down ⇒ Stabilize camera

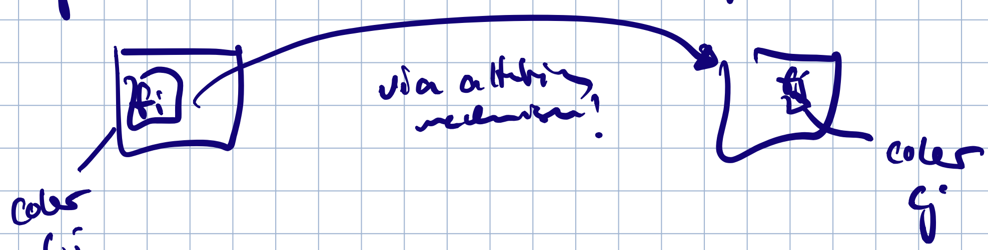
→ were correlated on black bars

- **TEMPORAL COHERENCE OF COLOUR** ⇒ **TRACKING**

→ Self-supervised tracking ⇒ semantic correspondence

→ Vondrick et al (ECCV, 2018)

↳ Reference Frame ↔ Target Frame



LEARN AUX. LOSS

?

Explainable AI / Interpretability

- **AUDIO - VISUAL STREAMS**

① Predict synchronization
② Predict correspondence] Proxy Tasks!

↳ Contrastive loss → Min. distance positive pairs, Max. dist. neg. pairs

↳ Synchronization → predict where source of audio comes from

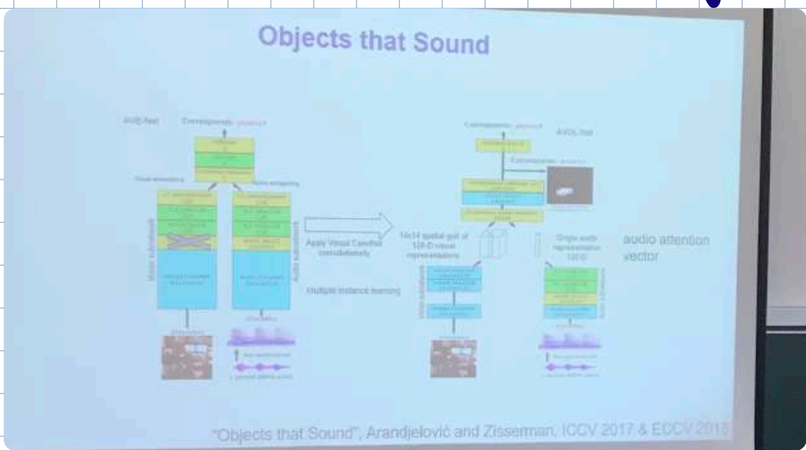
↳ Image(s) → VISUAL NETWORK → $\mathbb{R}^d$

Audio → AUDIO NETWORK → $\mathbb{R}^d$

] Contr. Loss

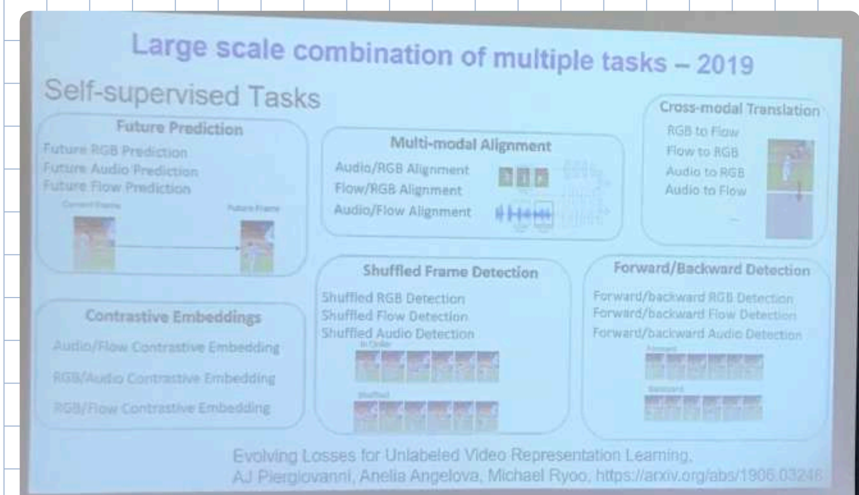
↳ Enforce networks to learn tight embeddings!

↳ Correspondence → single image + audio



→ How to go from single frame to detection

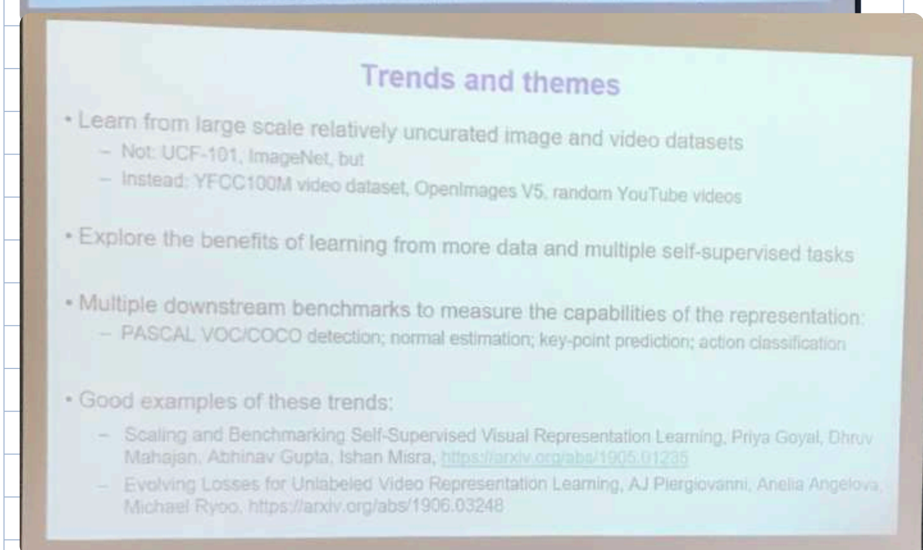
↳ No temporal notion!



→ Putting all different proxy tasks together

↳ Form of k-shot improvement!

↳ + lot more data efficient



→ Towards causality // learning physics / gravity!

Bayesian Learning - Dimitry Vetrov → Variational Inference!

$$p(Z|X) = \frac{p(X|Z)\,p(Z)}{\int p(X|Z)\,p(Z)\,dZ}$$ → Hard Integration

$$\log p(X) = \int q(Z) \log p(X)\, dZ$$

$$= \int q(Z) \log \frac{p(X,Z)}{q(Z)}\, dZ + \int q(Z) \log \frac{q(Z)}{p(Z|X)}\, dZ$$

$$= \boxed{\mathcal{L}(q)} + KL(q(Z) \,\|\, p(Z|X))$$

ELBO → stochastic optimization

- VI: Inference = optimization!

ϕ parametrizes variational distribution

→ $p(T, W|X) = p(T|X, W)\,p(W)$ → (DISCRIMINATIVE MODEL)

$$\phi^* = \arg\max \int q(W|\phi) \log \frac{p(T_{tr}, W|X_{tr})}{q(W|\phi)}\, dW$$

$$= \int q(W|\phi) \log p(T_{tr}|X_{tr}, W)\, dW - \int q(W|\phi) \log \frac{q(W|\phi)}{p(W)}\, dW$$

if we only opt. this average to converge to δ-fct. at ML

REGULARIZER!

$$= \sum_{i=1}^{n} \int q(W|\phi) \log p(t_i|x_i, W)\, dW - KL[q(W|\phi) \,\|\, p(W)]$$

apply minibatch!

Still intractable → MC estimate

Analytical Computation!

→ REPARAMETRIZATION TRICK: $\int r(\varepsilon) \log p(t_i|x_i, W(\varepsilon, \phi))\, d\varepsilon$

where $\varepsilon \sim \mathcal{N}(0, I)$, $W = \mu + \varepsilon \circ \sigma$

→ Use MC estimation to get stochastic gradient wrt. $\phi = \{\mu, \sigma\}$

EEML – Day 4 Bucharest

↳ INVERSE RL

RL / Planning with Humans – Anca Dragan (Berkeley)

- Example with arm pushing back ("following optimal policy") after being pushed by human arm
  ↳ does not respect human expressed intention
  - ↳ Reward fct. has to be specified
  - ↳ Multi-Agent setting ⇒ what if other agents are humans?!

PERSON IN ENV ①

END-USER ②

DESIGNER ③

- Value Iteration ⇒ do it yourself if you have access to the dynamics!
- Inverse RL : from traces $\xi \to R$
  - ↳ Inverse RL vs. Imitation Learning

Inverse RL:
- Reward fct. approx.
- Easy to compose diff. rewards
- Easy to regularize!
  - ↳ ~~Regularize~~ but too deep nets

Imitation Learning:
- Policy fct. approx.
- Problem: errors add exponential
- Hard to incorporate diff. people
  - ↳ all possible trajectories

↳ Problem: find $R(s,a)$ s.t. $R(\xi_D) \geq R(\xi)\ \forall \xi$ → gone

$R(s,a) = \theta^T \phi(s,a)$

↳ $R(\xi_D) \geq \max_\xi [R(\xi) + \ell(\xi, \xi_D)]$

$$\max_\theta \left[R(\xi_D) - \max_\xi \left[\theta^T \phi(\xi) + \ell(\xi, \xi_D)\right]\right]$$

→ max-margin planning

→ bonus for being far from demonstration

↳ form of regularizers

↳ can't just do 0 rewards

↳ optimize via GD!

⇒ subgradient

argmax $= \xi^*$

$$\nabla_\theta = \phi(\xi_D) - \phi(\xi_D^*)$$

- Suboptimal demonstrations ⇒ Want to be Bayesian!

$$P(\xi_D \mid \theta) \propto \frac{e^{\beta\theta^T\phi(\xi_D)}}{\sum_\xi e^{\beta\theta^T\phi(\xi)}}$$ → allow small prob for all trajectories

↳ $\beta=0$: Random demonstrator → uniform distr.
  $\beta<0$: Adversarial demonstrator
  $\beta\to\infty$: Deterministic demonstrator

- Problem: ξ are all possible trajectories → lead to compute/enumerate ⇒ Use softmax value iteration

↳ Want to do bayesian inference: $b'(\theta) \propto b(\theta)\, P(\xi_D \mid \theta)$
  ↳ but often revert to Max LH:

$$\max_\theta \log \frac{e^{\beta\theta^T\phi(\xi_D)}}{\sum_\xi e^{\beta\theta^T\phi(\xi)}} = \beta\theta^T\phi(\xi_D) - \log\sum_\xi e^{\beta\theta^T\phi(\xi)}$$

$$\nabla_\theta = \beta\left[\phi(\xi_D) - E_{\xi\sim\theta}\,\phi(\xi)\right]$$ → Exp. feature value produced by current reward

- Multi-agent problem: Robot waits forever → problem: Want robot to probe other agents!
  ↳ Distributional shift ⇒ Need back and forth
  robot probes human → human adapts → robot learns →