Small or noisy data : forms of weak supervision

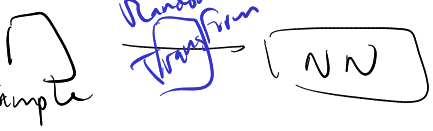
Academic setting :
- Large dataset
- Focus on architectures / training optim°

Real setting : data (amount)

I Small data

- Best case : simulator : train on simulated data, then fine-tune on real data
- data augmentat° :
  - vision : rotate, translate, crop, flip, spatial deformations

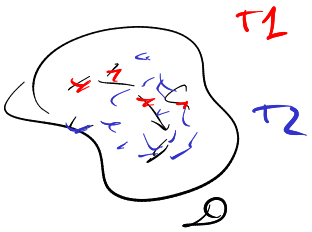
  - pixel intensities : contrast, color distance, noise : suited to the task

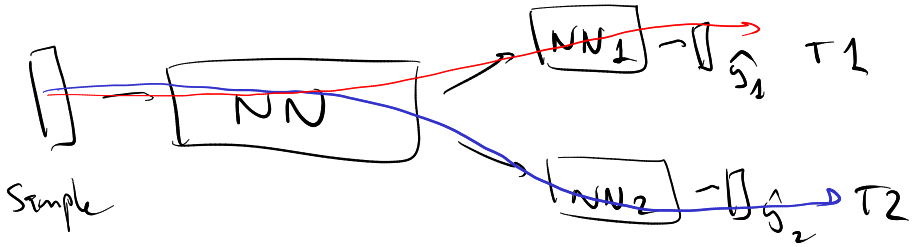


  - CutMix

- multi-tasking

Task 1 : small data
Task 2 : lot of data
} same data → few labels / → many labeled





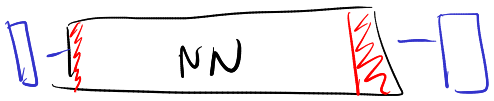
⚠ beware : similarity of tasks

Transfer learning

- learning from scratch : building architecture ... ≃ 6 months
- build on previous models : huggingface
  - benchmarks / challenges

leaderboard
95%
92%
...

1) copy the architecture

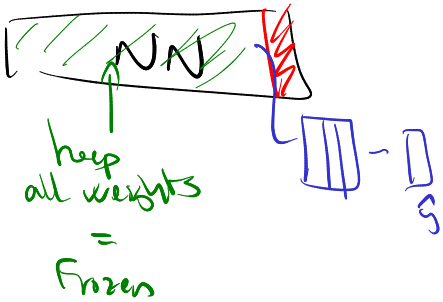


retrain from scratch ~ a few hours/days

+ copy the full processing/optim° chain: - pre-processing (data augm°)
- dropout
- early stopping
- learning rate scheduler

or 2) copy the weights: "pre-trained" model

- a few seconds minutes



or 3) <u>after</u> training the new layers, unfreeze the pretrained model & fine-tune → hours

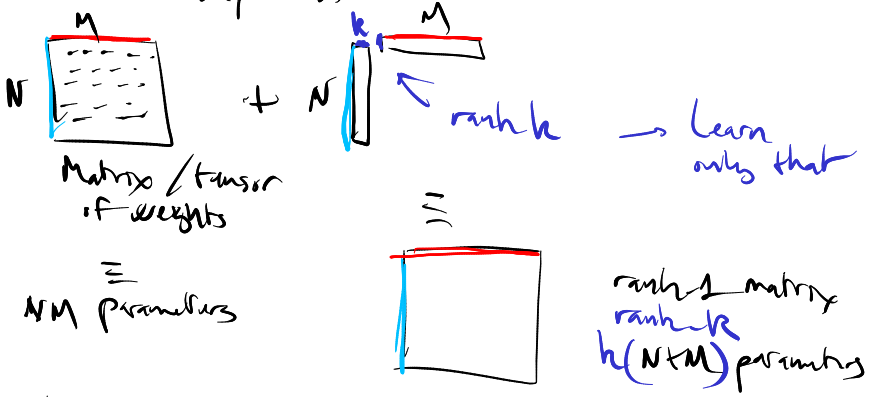
<u>Effect of transfer learning</u>

- if small data: ↯ help in getting relevant features
- if not small data:

← same accuracy but much faster

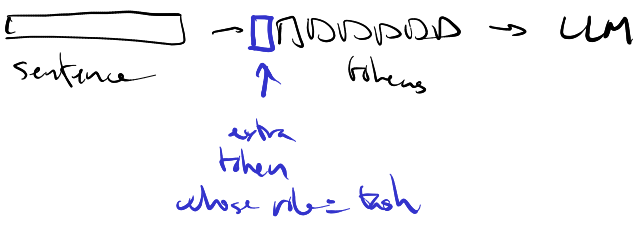
<u>Other ways to fine-tune</u>: (more specific to LLM)

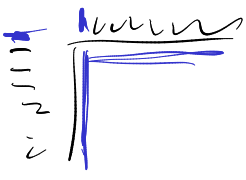
- LoRA: Low-Rank Adaptation



→ Learn only that

- add a few tokens





- prompting
- RAG

Remark about transfer / Foundation models

→ tend to transfer from large, generic models

→ end up with huge model

⇒ reduce models

→ **pruning** : weights / neurons / layers

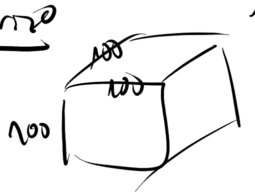
↳ group-Lasso L_1

how: - magnitude of weights
- " of weight gradient
- variation of weight since initialization

then: fine-tune

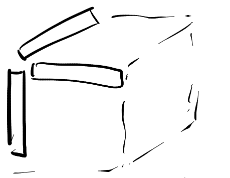
usually: iterative prune ⟲ fine-tune

→ **tensorize**

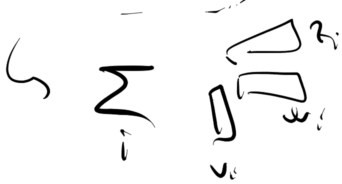


1 000 000 parameters → low-rank tensor?

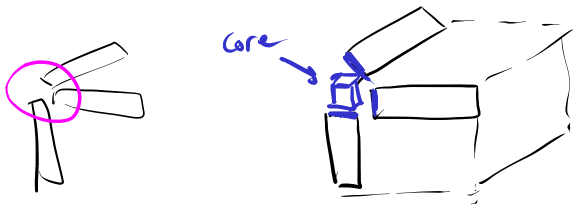
→ **CP decompositions**



rank-1 tensor

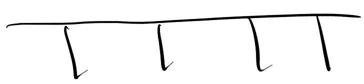


→ **Tucker**:



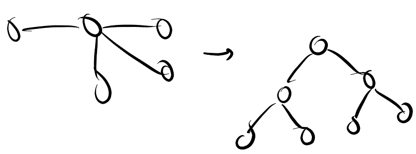
→ **Tensor Train**

sequence of multiplication of tensors



→ combine these approaches + lots of variations

↳ eg: hierarchical Tucker



→ quantize: $w = 2.551783\ldots \simeq 2.6$ float: 32 vs 16 vs ...

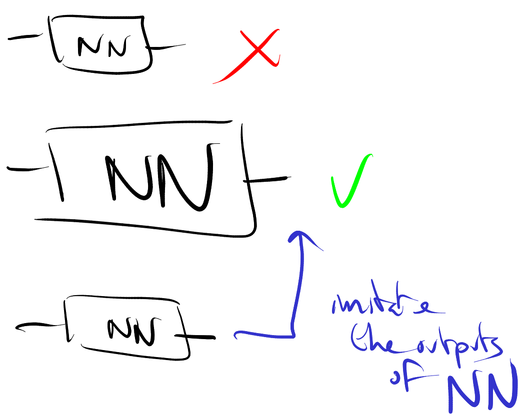
extreme case: binary networks
-1, 0, +1

list of possible weights: {-2.5, -0.1, 0.15 ...}
e.g. 5 values

↳ during training:
cluster the weights and push towards the centers

→ "knowledge distillation": "teacher-student"

Task:



≠ :
- 1 feedback
- as many feedbacks as #classes

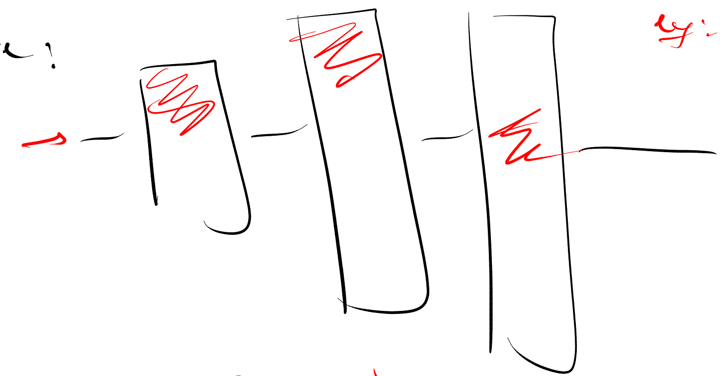
- Lottery ticket H0:

a subnetwork not even trained might do the job

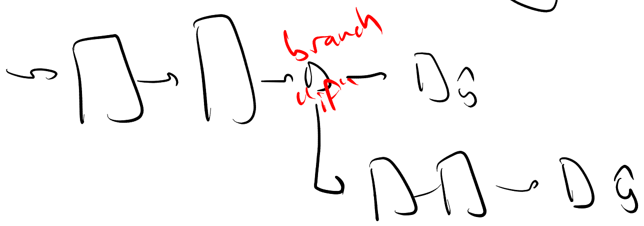
e.g. of small models:
- MobileNet
- SqueezeNet
- EfficientNet

"tiny ML"

↳ dynamic architecture:

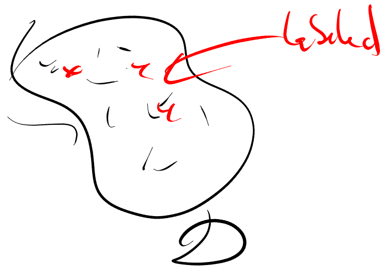


e.g.: "Mixture of Experts"



- Neural Architecture Growth for Frugal AI

II Few labeled ex : weak supervision



1) unsupervised pre-training (on all data)
then fine-tune on real task
with labeled samples

2) train supervisedly
then label a few more samples
(the most reliable ones)
& re-train
⚠ mistake

3) train supervisedly
but check global properties
on the whole dataset

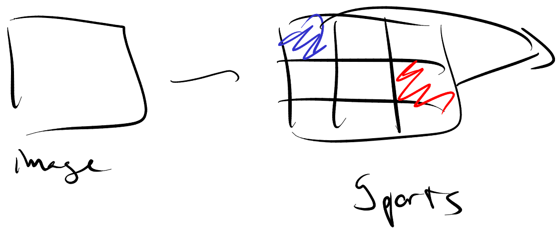
eg: bias, density ...
↳ binary classif pb
20% 80%
A B

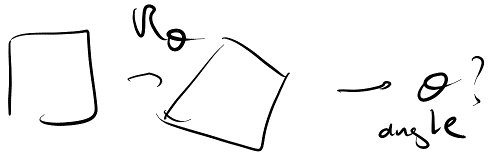
→ calibration

"Self-supervision"

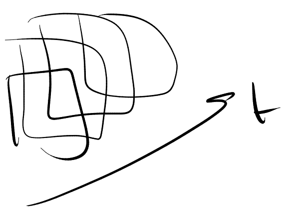
- create "dummy" task with labels automatically generated
↑ supervised

eg:



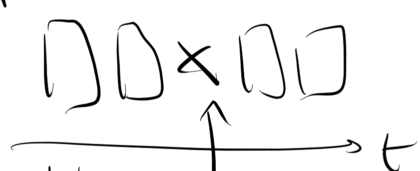


video:



- pick frames: ask order
& shuffle

- harder: predict a missing frame

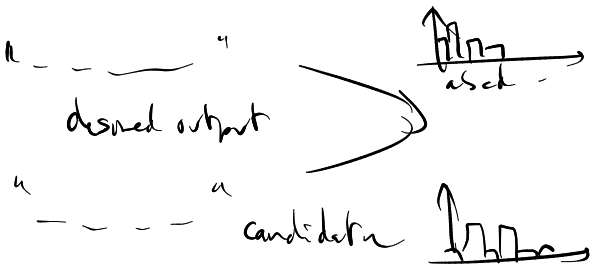


- hardest: predict next frame

→ useful in NLP

eg: translation

1) not many ex of translations
2) good metric? → BLEU / BLUE



Self-supervising:

Time is ∅ ?

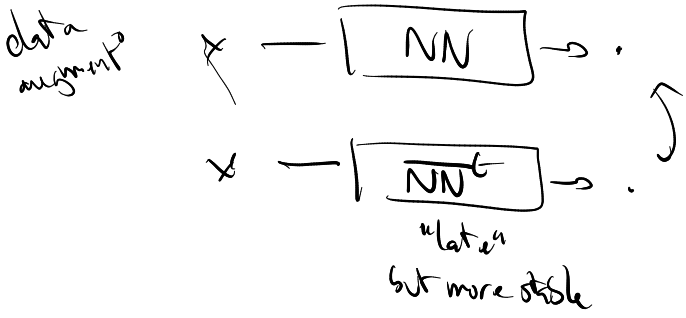
missing word

Clusterfit / Dino:

↳ dummy task → NN → cluster representation

NN2 → predict clusters

↳ Dino:



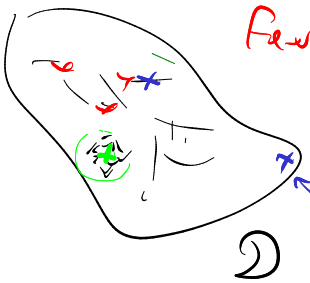
Sample x → Loss: $\|g_{(x)} - \hat{g}_{(x')}\|$

x' = augmented version (x)

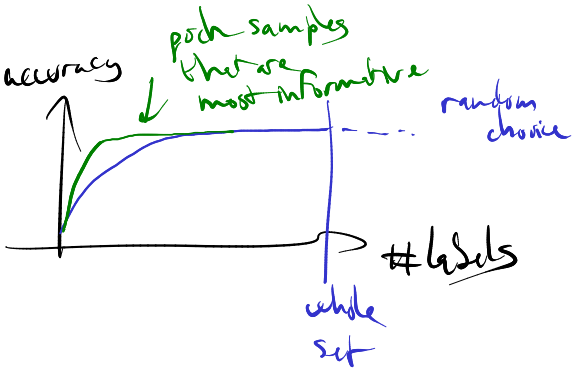
Learn features that are invariant to transform group

ash for variance on outputs

Active learning:

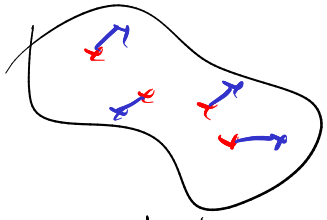


Few labeled samples because $ costs

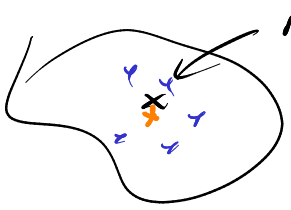


Noisy data:

potentially big data, but labels are noisy



label space



same sample regression

if noise centered:

$$\inf_{g} \sum_i \|\hat{g} - y_i\|^2$$

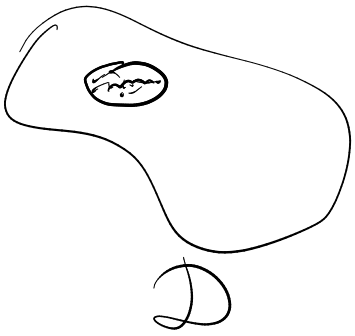
argmin

noisy labels

$$\hat{y}^* = \frac{1}{N} \sum_i y_i$$

$$\|\hat{y}^* - y_{real}\| \sim \frac{1}{\sqrt{N}} \sigma$$

noise variance



Classification

ImageNet: change labels

10% → random labels

25%

50%

99% →

×100 training set size ⇒ in order to keep #rightly-labeled N: