Deep Learning III: Data Woes

Dr Henry Gouk

UDRC Research Associate

2022 UDRC-EURASIP Summer School

THE UNIVERSITY of EDINBURGH
informatics

Overview

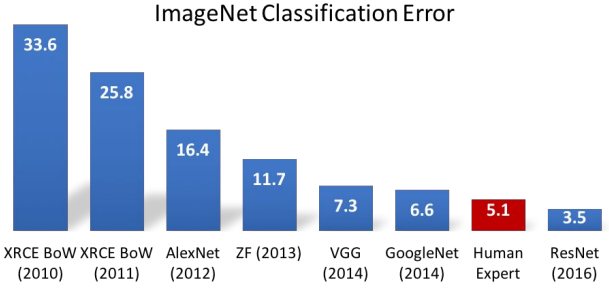
Learning with Limited Labelled Data

- Transfer Learning
- Meta-Learning
- Self-Supervised Learning

Dealing with Domain Shift

- Domain Generalisation
- Adversarial Domain Shift

Deep Learning Success



DeepMind

"Human-level control through deep reinforcement learning"

letter

Deep Q-Learning

5

Mechanism for Deep Learning Success?

Caffe

theano

TensorFlow

PyTorch

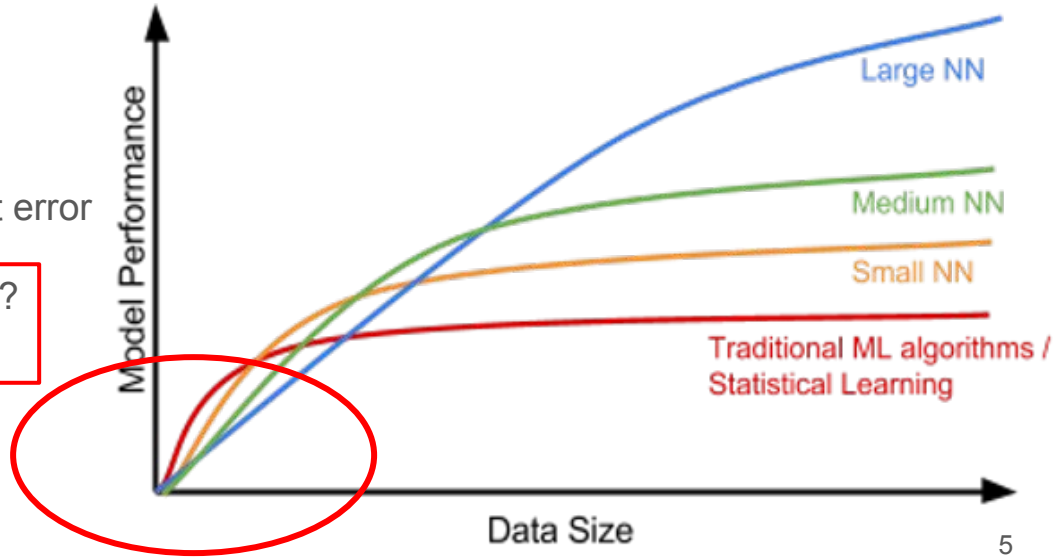
```
from torchvision import models
model = models.resnet18(pretrained=True)
```

```
pred = model(image)

loss = cross_entropy(pred, true_label)

loss.backward()

optimiser.step()
```

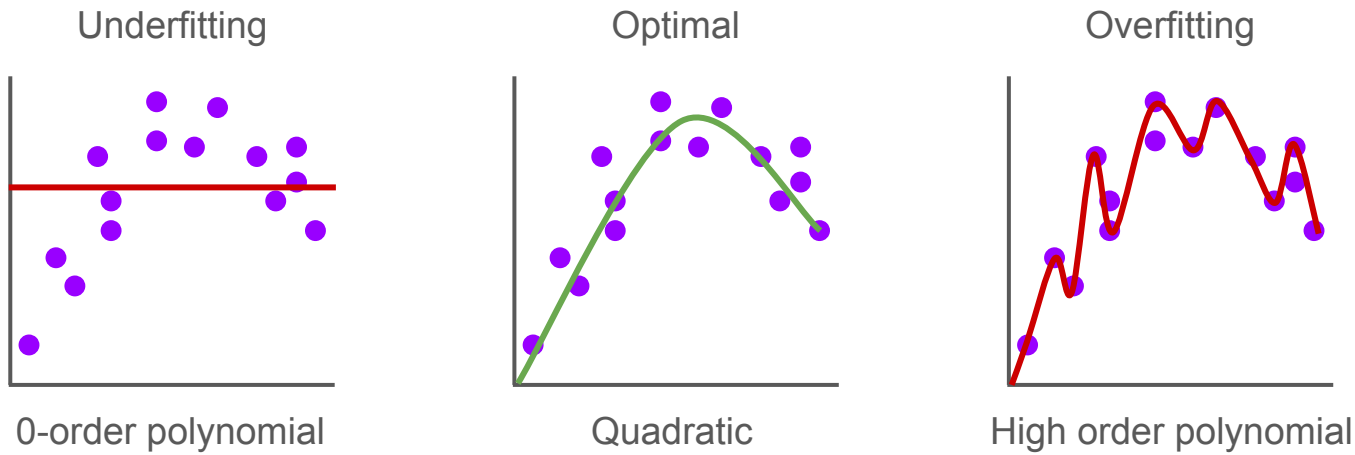
Mechanism for Deep Learning Success?

$$\text{Test Error} < \text{Training Error} + \frac{\text{Model Capacity}}{\text{Train Set Size}}$$

- Increase model capacity?
  - better train error, worse test error
- Increase train set size?
  - (maybe) worse train error, better test error
- Increase model capacity and train set size?
  - better train and test error!



Why is Low-Data Learning Difficult? Overfitting



Question: How to diagnose over- vs under-fitting?

Underfitting?

- High train error
- High test error

Overfitting?

- Low train error
- High test error

How to Avoid Overfitting?

Classic Solutions

- Try several models with different capacities
- Evaluate validation set performance for each
- Pick the model with best validation performance

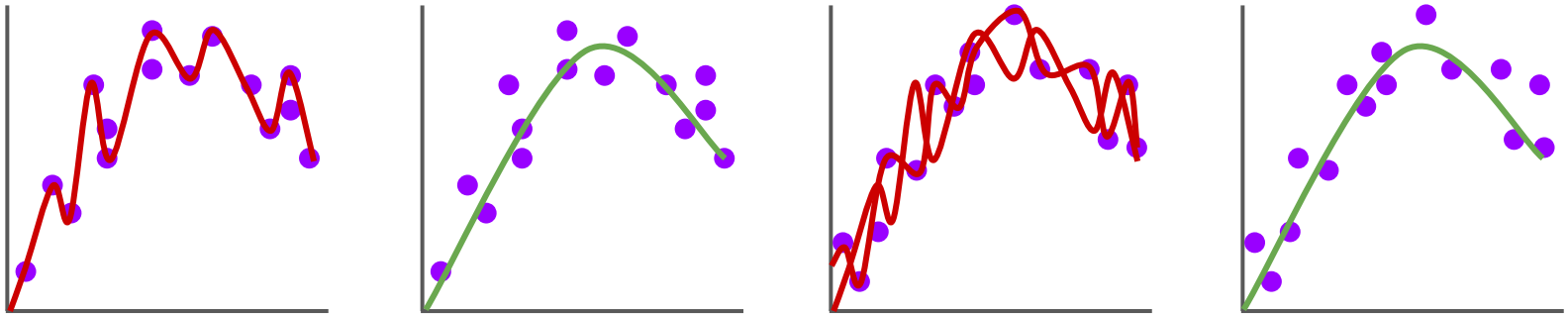
Deep learning Issues

- Too many parameters impact capacity
- In context of data scarcity: would pick a simple model that doesn’t provide deep learning level of performance

How to Avoid Overfitting?

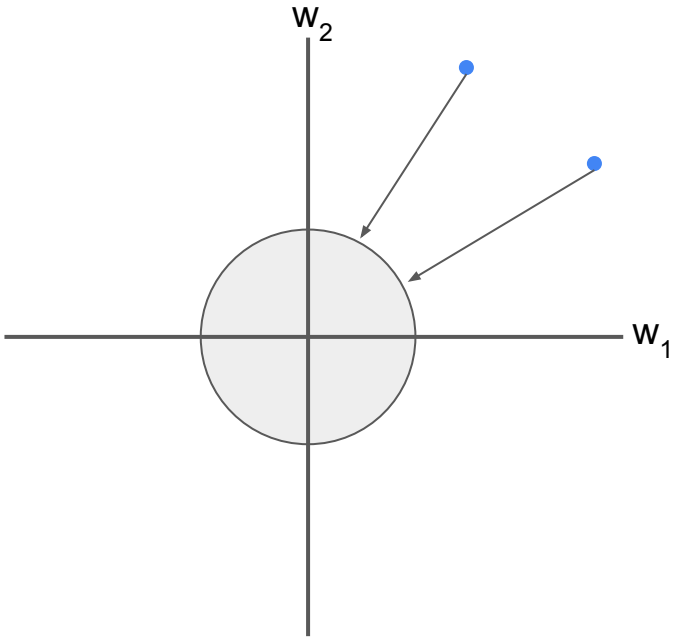
Ask yourself:

Would my pipeline give me a similar model if I collected a new training set?



How can we control modelling capacity?

Controlling Modelling Capacity: Weight Decay



Minimise Training Loss + $||w||^2$

Problem: might just make us underfit

Benefit of weight decay is often quite marginal in neural networks

How can we more intelligently allocate modelling capacity?

Overview

Learning with Limited Labelled Data

- **Transfer Learning**
- Meta-Learning
- Self-Supervised Learning

Dealing with Domain Shift

- Domain Generalisation
- Adversarial Domain Shift

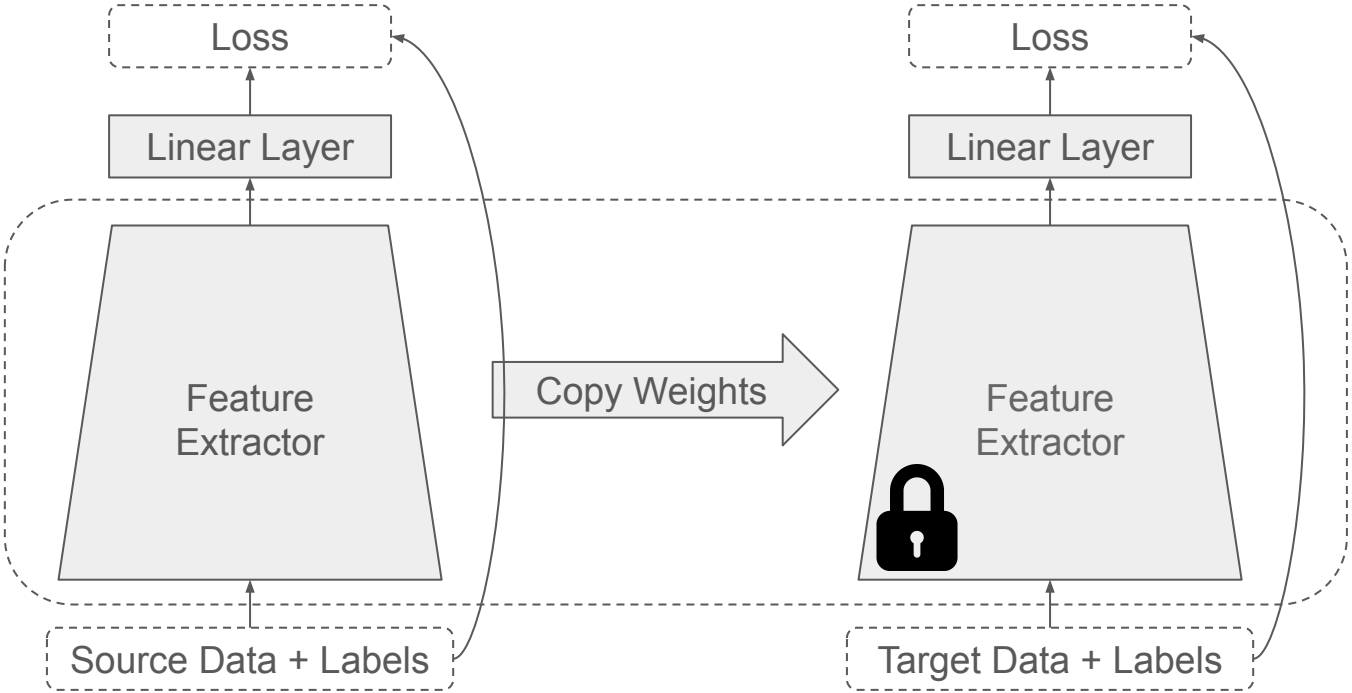
Transfer Learning

"*The application of skills, knowledge, and/or attitudes that were learned in one situation to another learning situation*" (Perkins, 1992)

Note

- Fine-tuning ≠ Transfer Learning
- Fine-tuning ⊂ Transfer Learning

Transfer Learning: Linear Readout



Tuning the model

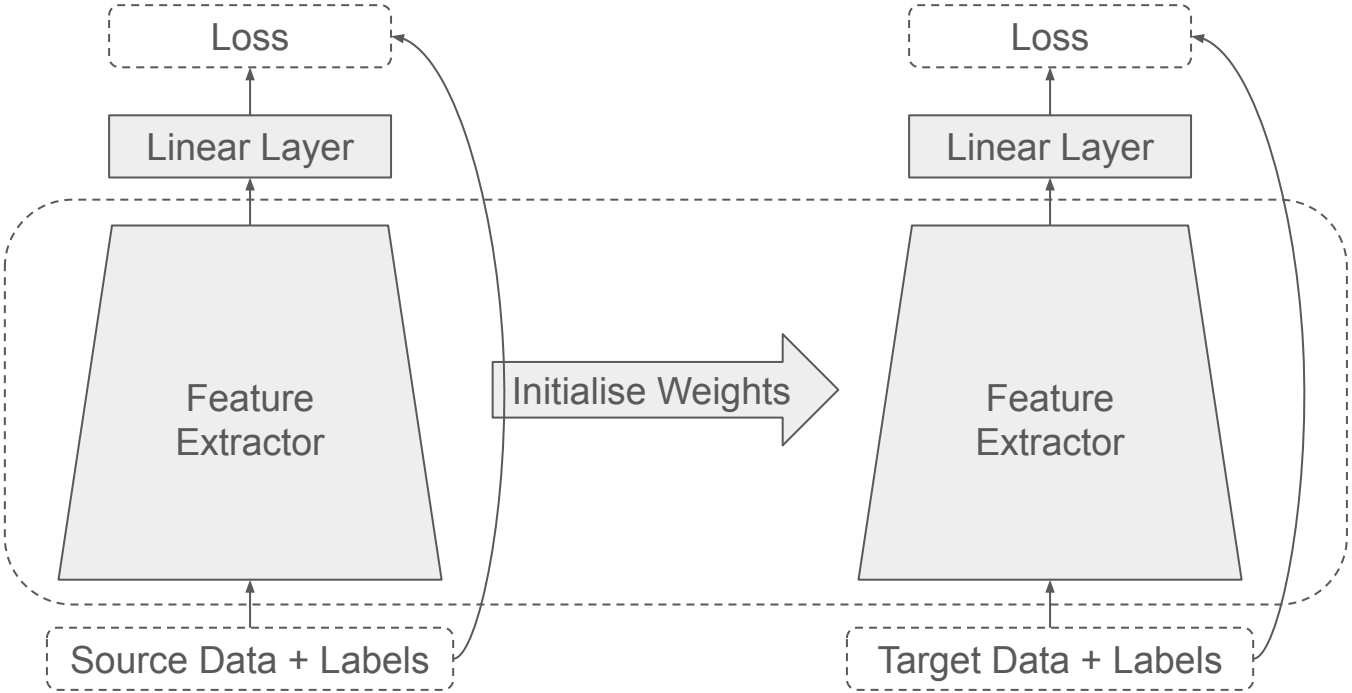
Use standard methods to optimise linear model

Conventional validation/cross validation is typically sufficient

Often work wells for other types of “heads” as well

Bonus: data augmentation probably still beneficial

Transfer Learning: Fine-Tuning



Back to the mess of deep learning design choices:

- Which optimiser to use?
- How to tune optimiser parameters?
- Should we still freeze some layers?
- Should we do early stopping?

Transfer Learning: Fine-Tuning Considerations

How are we allocating modelling capacity?

- Trying to keep weights near informative initialisation
- Contrast with weight decay: keeping weights near uninformative initialisation

Fine-tuning “tricks”:

- Use a small learning rate
- Do early stopping
- Freeze some layers
  - Early layers if task shift
  - Later layers if input distribution shift

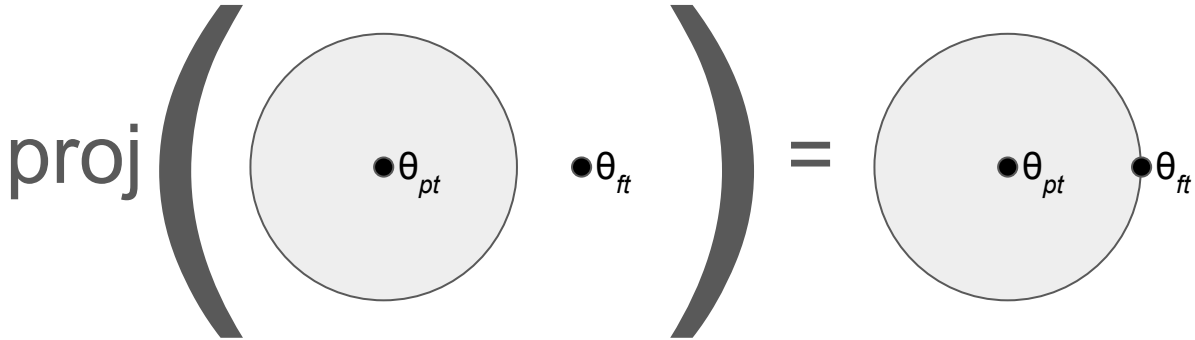
Why not add an explicit regulariser like weight decay?

Transfer Learning: Advanced Fine-Tuning

Penalty Term

$$\min_{\theta_{ft}} \mathcal{L}(f_{\theta_{ft}}(\vec{x}), y) + \lambda d(\theta_{ft}, \theta_{pt})$$

Projection Function



Could also:

- Penalise deviations in activations
- Choose which layers to freeze/unfreeze

Transfer Learning: Advanced Fine-Tuning

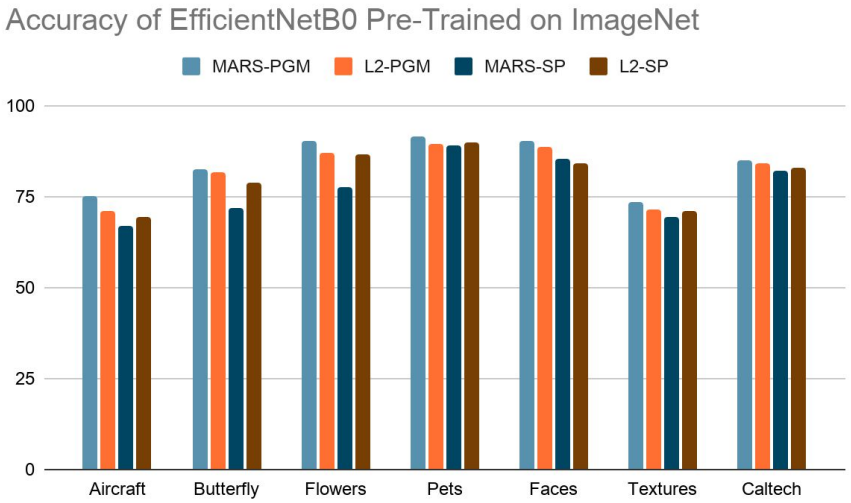
How to measure distance in weight space?

$$d_{mars}(W, V) = \max_i \sum_j |W_{ij} - V_{ij}|$$

Capacity $\propto$ Distance, θ_{pt}

$$d_{frob}(W, V) = \sqrt{\sum_{ij} (W_{ij} - V_{ij})^2}$$

Capacity $\propto$ Distance, θ_{pt}, no. units



-PGM denotes projection method
-SP denotes penalty method

Further Reading

- Li et al. Explicit Inductive Bias for Transfer Learning with Convolutional Networks. ICML 2018.
- Gouk et al. Distanced-Based Regularisation of Deep Networks for Fine-Tuning. ICLR 2020.

Overview

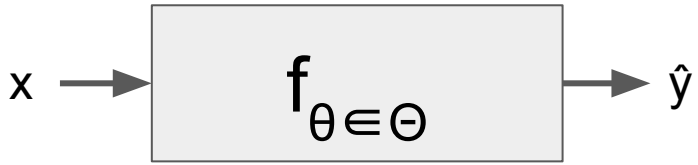
Learning with Limited Labelled Data

- Transfer Learning
- **Meta-Learning**
- Self-Supervised Learning

Dealing with Domain Shift

- Domain Generalisation
- Adversarial Domain Shift

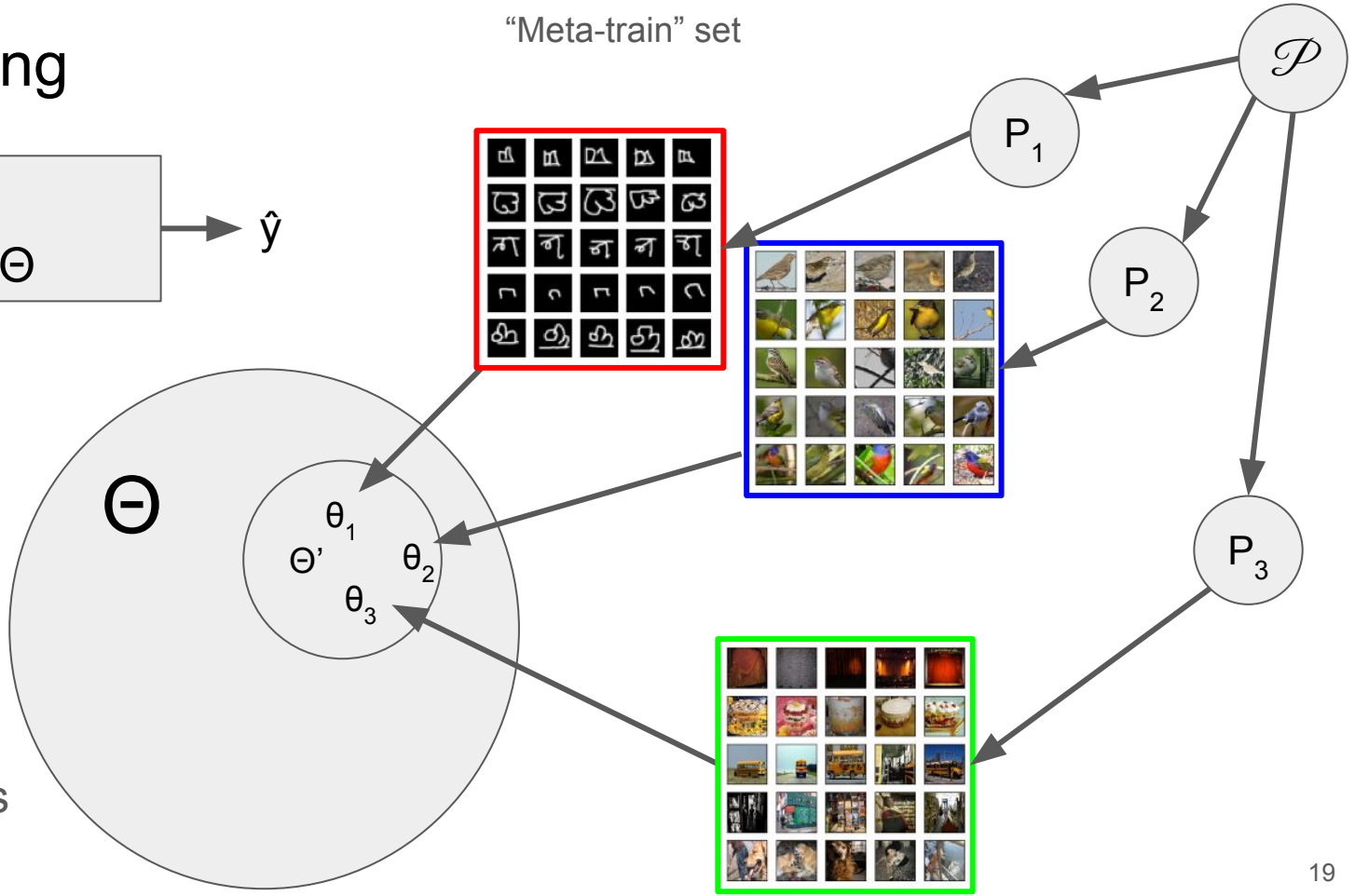
Meta-Learning



Capacity ∝ |Θ|

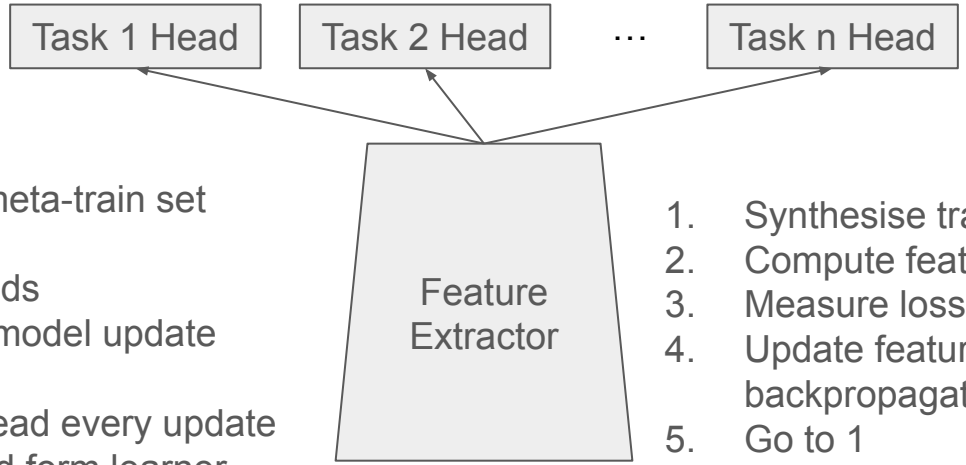
Only use relevant subset of Θ

Meta-learning finds this subset



Metric-Based Meta-Learning

Basic idea: shared feature extractor, different head for each task



Synthesise tasks from meta-train set

Problem: too many heads
Solution: one task per model update

Problem 2: train new head every update
Solution 2: use a closed form learner

1. Synthesise training task from meta-train
2. Compute feature means for each class
3. Measure loss of classifier
4. Update feature extractor with backpropagation + SGD
5. Go to 1

Nearest centroid classifier: use mean feature vector for each class

Gradient-Based Meta-Learning

Meta-knowledge

$$\phi^* = \arg\min_{\phi} \sum_i \mathcal{L}(D_i^{val}, \theta_i^*, \phi)$$ Outer problem

$$\text{s.t.} \quad \theta_i^* = \arg\min_{\theta} \mathcal{L}(D_i^{tr}, \theta, \phi)$$ Inner problem

Intuitively: find meta-knowledge that gives best performance on unseen data

How to solve?

Model Agnostic Meta-Learning: easiest method, but quite inefficient

$$\phi^* = \arg\min_{\phi} \sum_i \mathcal{L}(D_i^{val}, \text{sgd-step}(\text{sgd-step}(\phi, D_i^{tr}), D_i^{tr}))$$

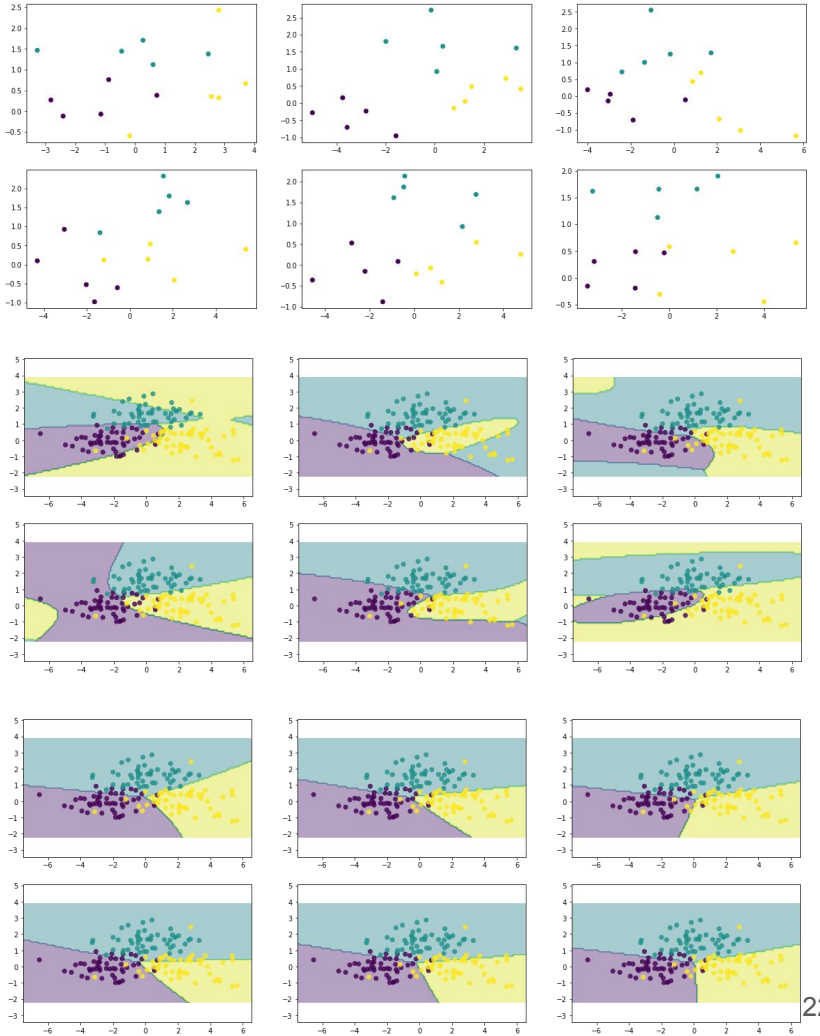
Bayesian Meta-Learning

Motivation: Bayesian probabilistic modelling enables information fusion, incorporating prior knowledge, etc

Simple probabilistic classifier: fit a Gaussian to each class with maximum likelihood (QDA)

Problem: QDA does not work well with small training datasets—we want different training sets to give similar models!

Solution: Meta-learn a prior on related tasks, compute full posterior over parameters



Bayesian Meta-Learning

Model	Backbone	1-shot	5-shot
MAML [37]	Conv-4	$58.90 \pm 1.90\%$	$71.50 \pm 1.00\%$
RELATIONNET [37]	Conv-4	$55.50 \pm 1.00\%$	$69.30 \pm 0.80\%$
PROTONET [37]	Conv-4	$55.50 \pm 0.70\%$	$72.02 \pm 0.60\%$
R2D2 [3]	Conv-4	$62.30 \pm 0.20\%$	$77.40 \pm 0.10\%$
SIMPLESHOT$^+$ [61]	Conv-4	$59.35 \pm 0.89\%$	$74.76 \pm 0.72\%$
METAQDA	Conv-4	$\mathbf{60.52 \pm 0.88\%}$	$\mathbf{77.33 \pm 0.73\%}$
PROTONET [37]	ResNet-12	$72.20 \pm 0.70\%$	$83.50 \pm 0.50\%$
METAOPT [30]	ResNet-12*	$72.00 \pm 0.70\%$	$84.20 \pm 0.50\%$
UNRAVELLING [14]	ResNet-12*	$72.30 \pm 0.40\%$	$86.30 \pm 0.20\%$
BASELINE++ [4, 37]	ResNet-18	$59.67 \pm 0.90\%$	$71.40 \pm 0.69\%$
S2M2 [37]	ResNet-18	$63.66 \pm 0.17\%$	$76.07 \pm 0.19\%$
METAOPTNET [30]	WRN	$72.00 \pm 0.70\%$	$84.20 \pm 0.50\%$
BASELINE++ [37, 4]	WRN	$67.50 \pm 0.64\%$	$80.08 \pm 0.32\%$
S2M2 [37]	WRN	$74.81 \pm 0.19\%$	$87.47 \pm 0.13\%$
METAQDA	WRN	$\mathbf{75.83 \pm 0.88\%}$	$\mathbf{88.79 \pm 0.75\%}$

Model	Backbone	ECE+TS		ECE	
		1-shot	5-shot	1-shot	5-shot
LIN.CLASSIF.	Conv-4	3.56	2.88	8.54	7.48
SIMPLESHOT	Conv-4	3.82	3.35	33.45	45.81
QDA	Conv-4	8.25	4.37	43.54	26.78
MQDA-MAP	Conv-4	2.75	0.89	8.03	5.27
MQDA-FB	Conv-4	**2.33**	**0.45**	**4.32**	**2.92**
S2M2+LIN.CLASSIF	WRN	4.93	2.31	33.23	36.84
SIMPLESHOT	WRN	4.05	1.80	39.56	55.68
QDA	WRN	4.52	1.78	35.95	18.53
MQDA-MAP	WRN	3.94	0.94	31.17	17.37
MQDA-FB	WRN	**2.71**	**0.74**	**30.68**	**15.86**

Improved calibration: expected calibration error measures quality of model uncertainty

Note: can use pre-trained feature extractor and still meta-learn prior

Further Reading

- Hospedales et al. Meta-Learning in Neural Networks: A Survey. IEEE T-PAMI 2021.
- Snell et al. Prototypical Networks for Few-Shot Learning. NeurIPS 2017.
- Finn et al. Model-Agnostic Meta-Learning. ICML 2017.
- Zhang et al. Shallow Bayesian Meta-Learning for Real-World Few-Shot Recognition. ICCV 2021.

Overview

Learning with Limited Labelled Data

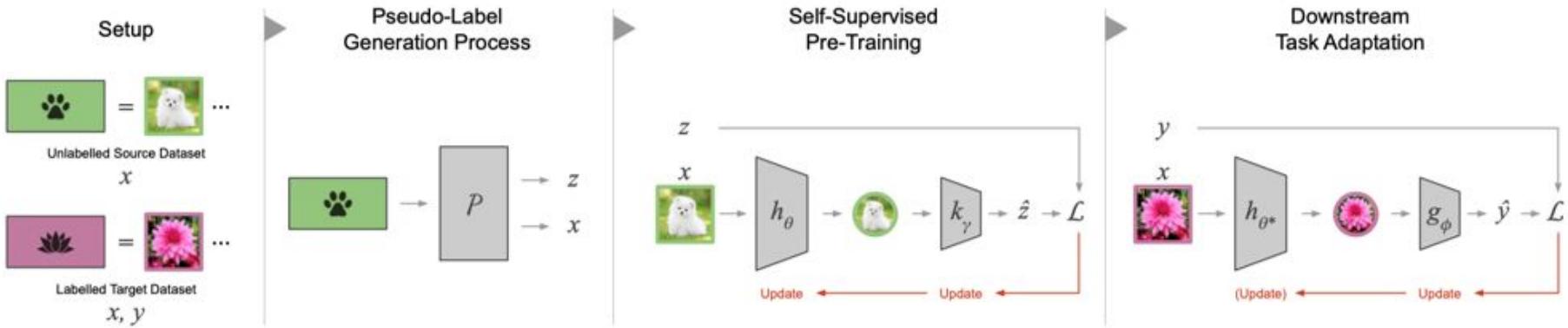
- Transfer Learning
- Meta-Learning
- **Self-Supervised Learning**

Dealing with Domain Shift

- Domain Generalisation
- Adversarial Domain Shift

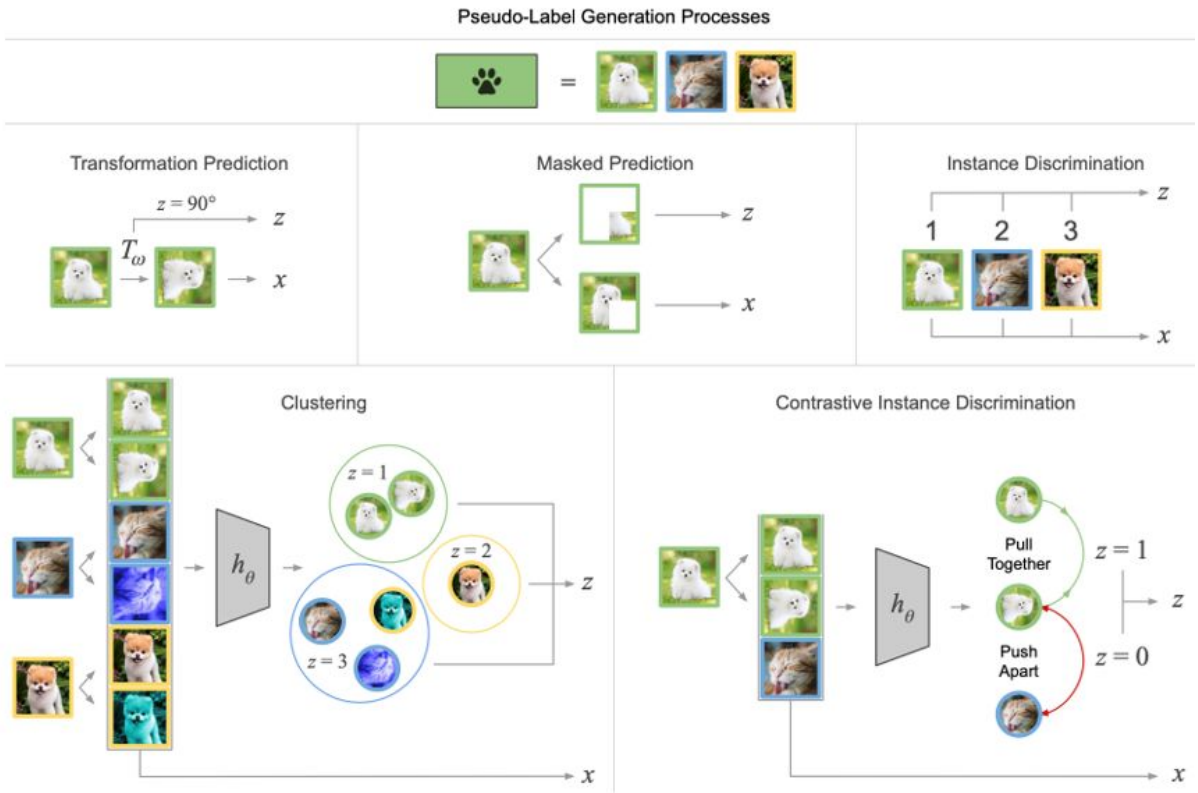
Self-Supervision

Main idea: synthesise supervised “pre-text task” out of unlabelled data

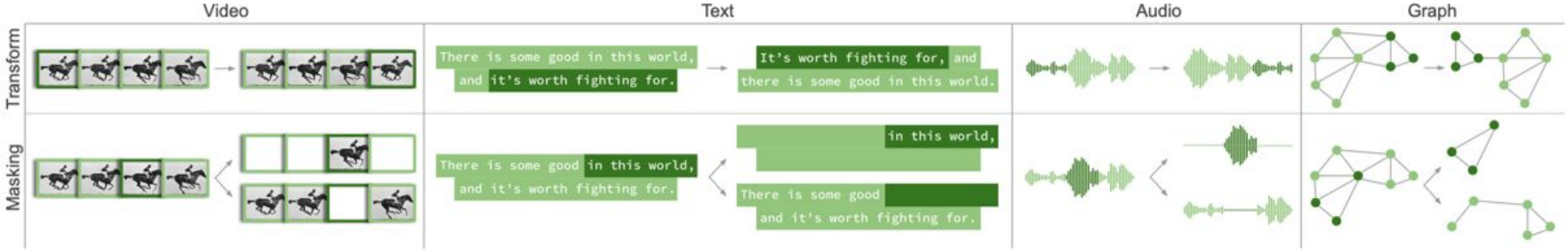


Use-case: alternative to supervised pre-training; use as initialisation for fine-tuning, linear readout, etc

Self-Supervision: Pre-text Tasks

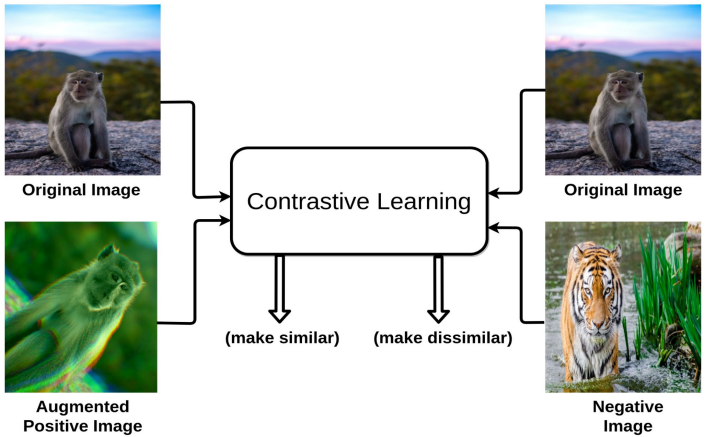


Self-Supervision: Pre-text Tasks



Contrastive instance discrimination:

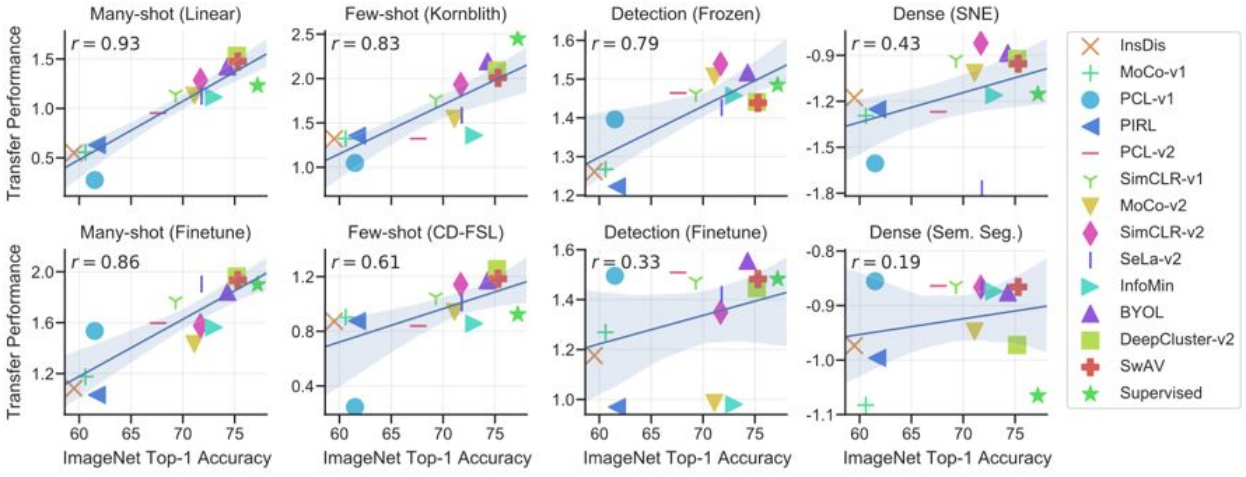
- Generate multiple views of instance with data augmentation
- Force different views to have similar representation
- Can be used to train invariances



Self-Supervision: Results

A lot of academic evaluations focus on ImageNet

- Supervised learning is still best for ImageNet



Self-supervised learning works better for transfer learning!

Further Reading

- Ericsson et al. Self-Supervised Representation Learning: Introduction, Advances, and Challenges. IEEE SPM 2022.

- Ericsson et al. How Well do Self-Supervised Models Transfer? CVPR 2021.

Overview

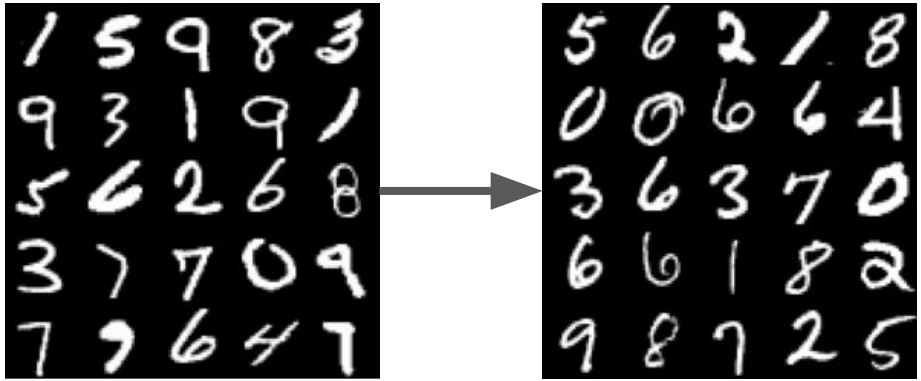
Learning with Limited Labelled Data

- Transfer Learning
- Meta-Learning
- Self-Supervised Learning

Dealing with Domain Shift

- Domain Generalisation
- Adversarial Domain Shift

Dealing with Domain Shift



Train

Test

$Z_1^{Train}, \ldots, Z_m^{Train}, Z_1^{Test}, \ldots, Z_m^{Test} \sim P$

Assumed to be unknown

Key point: assumed train and test data are independent and identically distributed

What if this isn't true?

Overview

Learning with Limited Labelled Data

- Transfer Learning
- Meta-Learning
- Self-Supervised Learning

Dealing with Domain Shift

- **Domain Generalisation**
- Adversarial Domain Shift

Domain Generalisation

Example Application:

A mobile robot needs to be able to detect whether cows are obstructing its path. We have plenty of examples of unobstructed paths, and we gather some examples of what cows look like so we can train a binary classifier.

Domain Generalisation

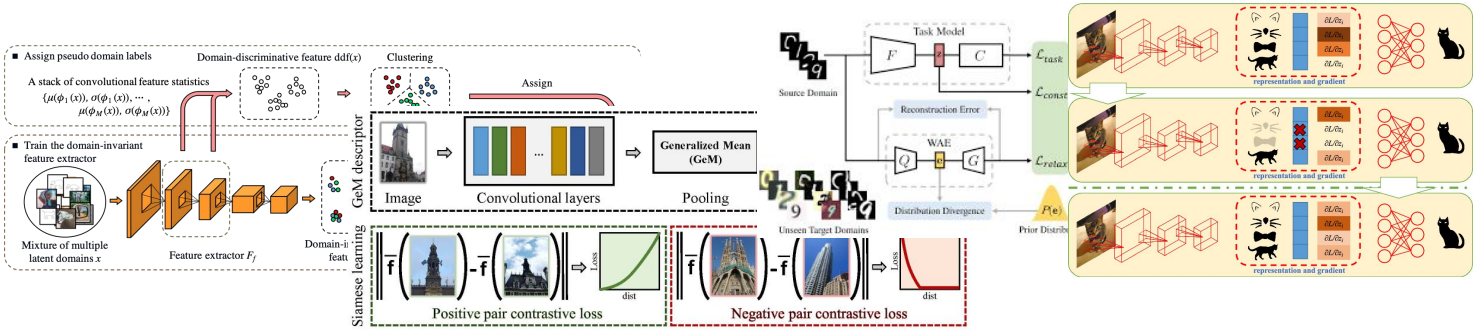
Train set examples

After model deployment

Domain Shift: from environment, different calibration, time varying concept, etc…

Another solution: causal model with appropriate invariances

Domain Generalisation: Learn Appropriate Invariances?



In Search of Lost Domain Generalization, ICLR 2021

Algorithm	CMNIST	RMNIST	VLCS	PACS	Office-Home	TerraInc	DomainNet	Avg
ERM	34.2 ± 1.2	98.0 ± 0.0	76.8 ± 1.0	83.3 ± 0.6	67.3 ± 0.3	46.2 ± 0.2	40.8 ± 0.2	63.8
IRM	36.3 ± 0.4	97.7 ± 0.1	77.2 ± 0.3	82.9 ± 0.6	66.7 ± 0.7	44.0 ± 0.7	35.3 ± 1.5	62.9
DRO	32.2 ± 3.7	97.9 ± 0.1	77.5 ± 0.1	83.1 ± 0.6	67.1 ± 0.3	42.5 ± 0.2	32.8 ± 0.2	61.8
Mixup	31.2 ± 2.1	98.1 ± 0.1	78.6 ± 0.2	83.7 ± 0.9	68.2 ± 0.3	46.1 ± 1.6	39.4 ± 0.3	63.6
MLDG	36.9 ± 0.2	98.0 ± 0.1	77.1 ± 0.6	82.4 ± 0.7	67.6 ± 0.3	45.8 ± 1.2	42.1 ± 0.1	64.2
CORAL	29.9 ± 2.5	98.1 ± 0.1	77.0 ± 0.5	83.6 ± 0.6	68.6 ± 0.2	48.1 ± 1.3	41.9 ± 0.2	63.9
MMD	42.6 ± 3.0	98.1 ± 0.1	76.7 ± 0.9	82.8 ± 0.3	67.1 ± 0.5	46.3 ± 0.5	39.3 ± 0.9	64.7
DANN	29.0 ± 7.7	89.1 ± 5.5	77.7 ± 0.3	84.0 ± 0.5	65.5 ± 0.1	45.7 ± 0.8	37.5 ± 0.2	61.2
C-DANN	31.1 ± 8.5	96.3 ± 1.0	74.0 ± 1.0	81.7 ± 1.4	64.7 ± 0.4	40.6 ± 1.8	38.7 ± 0.2	61.1

Domain Generalisation: Is there any hope?

ERM works quite well! But previous work has shown…

- Overfitting behaviour is different
- Tuning hyperparameters is more difficult

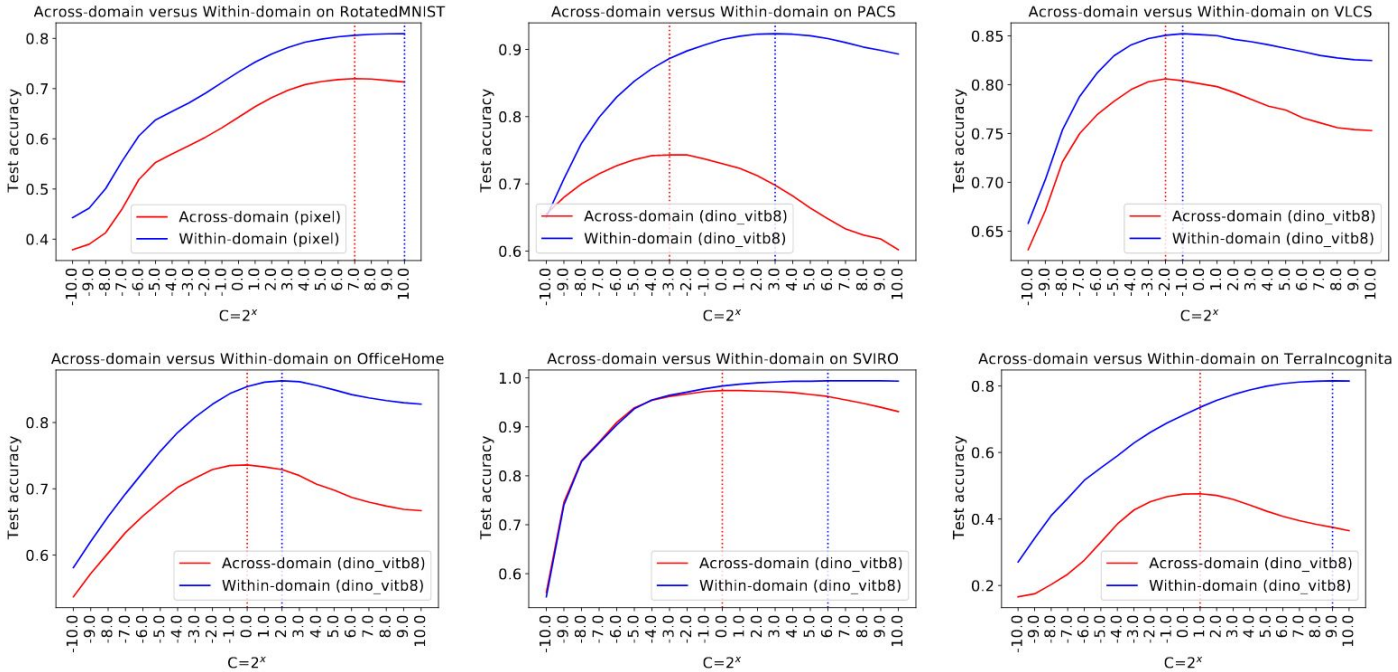
What does theory say?

$$\text{Test Error} < \text{Training Error} + \frac{\text{Model Capacity}}{\text{Train Set Size}} + \frac{\text{Model Capacity}}{\text{Train Domains}}$$

Observation: performance is more dependent on model capacity

Domain Generalisation: Smaller Modelling Capacity?

Simple Experiment: in domain vs out of domain performance of linear SVMs, multiple datasets, varying modelling capacities



Domain Generalisation: Validation Sets

Instance-wise

Train set

Domain-wise

Validation set

Domain Generalisation: Comparing Validation Sets

Which method for constructing validation sets works best in practice?

		Domain Wise		Instance Wise	
		Acc	$\log C$	Acc	$\log C$
Raw pixel	RotatedMNIST	71.6	4.97 ($\pm$0.52)	71.2	6.82 ($\pm$0.28)
DINO VIT-B8	PACS	74.3	-1.73 ($\pm$0.40)	70.6	1.56 ($\pm$0.66)
	VLCS	80.4	-1.04 ($\pm$0.69)	80.5	-0.87 ($\pm$0.35)
	OfficeHome	73.6	-0.17 ($\pm$0.35)	73.1	1.04 ($\pm$0.40)
	SVIRO	97.2	0.21 ($\pm$0.33)	95.3	4.85 ($\pm$0.46)
	Terra Incognita	45.4	-0.17 ($\pm$0.87)	38.8	5.37 ($\pm$0.35)
DINO VIT-S8	PACS	73.0	-1.91 ($\pm$1.43)	71.6	1.04 ($\pm$0.89)
	VLCS	80.4	-1.91 ($\pm$0.66)	80.6	-1.21 ($\pm$0.35)
	OfficeHome	72.2	-0.69 ($\pm$0.57)	72.3	0.35 ($\pm$0.40)
	SVIRO	94.5	-1.18 ($\pm$0.47)	91.3	5.68 ($\pm$0.97)
	Terra Incognita	37.1	0.00 ($\pm$1.50)	33.1	4.85 ($\pm$0.57)

Main Observations:

- Tuning with domain-wise validation set gives better performance, on average
- Tuning with domain-wise validation set chooses stronger regularisation

Further Reading

- Gulrajani & Lopez-Paz. In Search of Lost Domain Generalization. ICLR 2021.
- Li et al. Finding Lost DG: Explaining Domain Generalisation via Model Complexity. arXiv 2022.

Overview

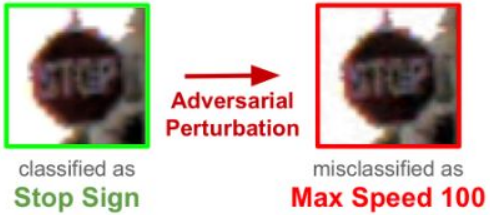
Learning with Limited Labelled Data

- Transfer Learning
- Meta-Learning
- Self-Supervised Learning

Dealing with Domain Shift

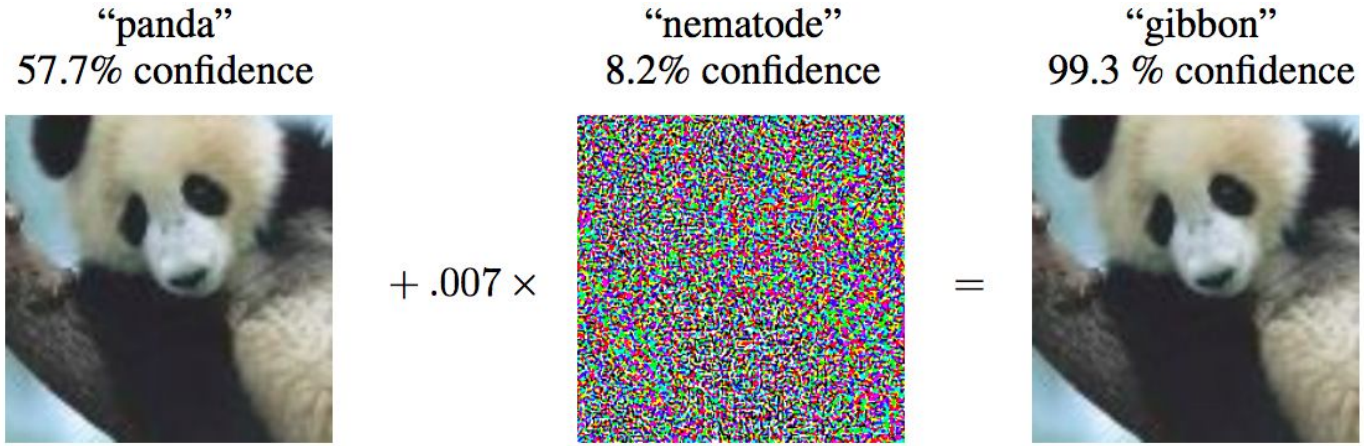
- Domain Generalisation
- **Adversarial Domain Shift**

Adversarial Examples



"Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake" – Goodfellow, 2017

In many cases the interference is imperceptible

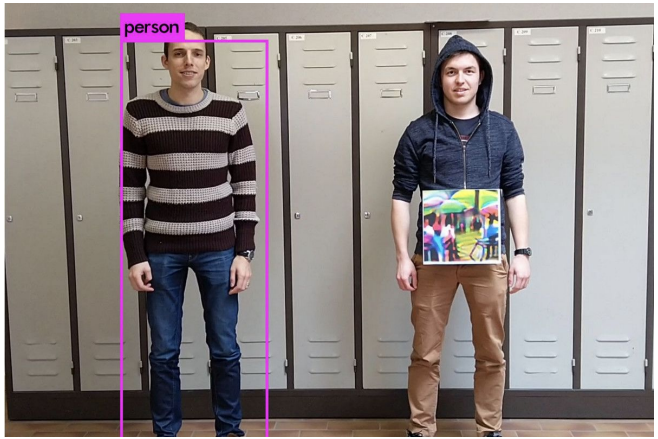


Important: not only an issue for deep learning

Adversarial Examples: Physical Perceptible Interference

Recognition

- Interference does not have to spatially coincide
- Single patch can be quite versatile



Detection

- Only minor occlusion
- Could be made to look more innocuous

Adversarial Examples: Crafting Interference

Fast Gradient Sign Method (FGSM):

$$\tilde{x} = x + \epsilon \operatorname{sign}(\nabla_x \mathcal{L}(f(x), y))$$

Projected Gradient Descent:

with:

$$\tilde{x}^0 = x$$

repeat:

$$x' = \tilde{x}^t + \epsilon \operatorname{sign}(\nabla_x \mathcal{L}(f(\tilde{x}^t), y))$$

$$\tilde{x}^{t+1} = \operatorname{proj}(x', \epsilon)$$

ensures $\|x - \tilde{x}^{t+1}\| \leq \epsilon$

Adversarial Examples: Adversarial Training

Key point: expose neural network to adversarial attacks during training

Repeat:

1. Sample minibatch, B
2. For each (x, y) in B
   a. Use FGSM or PGD to find an adversarial example, x’
   b. Put (x’, y) in B’
3. Update network parameters using B and B’

Pro: very effective

Con: very slow

Further Reading

- Szegedy et al. Intriguing Properties of Neural Networks. arXiv 2013.
- Madry et al. Towards Deep Models Resistant to Adversarial Attacks. ICLR 2018.
- Silva & Najafirad. Opportunities and Challenges in Deep Learning Adversarial Robustness: A Survey. arXiv 2020.

Fin