THE UNIVERSITY of EDINBURGH
informatics

UDRC Summer School 2021: Machine Learning: Deep Neural Networks III

Prof. Timothy Hospedales

-

UDRC Co-Investigator

-

Machine Intelligence Group - University of Edinburgh

https://homepages.inf.ed.ac.uk/thospeda/

-

Program Director, Machine Learning & Data Intelligence

Samsung AI Centre, Cambridge

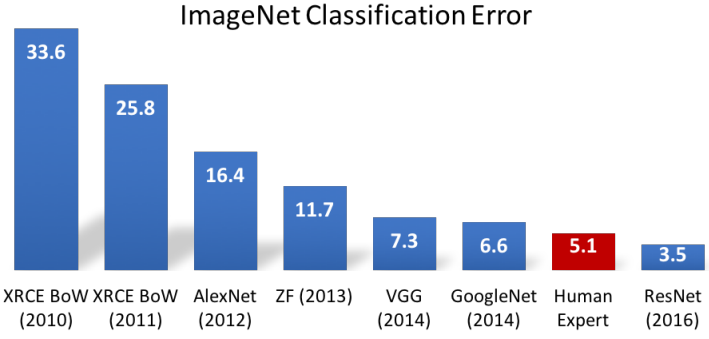
About Me: Prof. Tim Hospedales

- Background
  - BA Computer Science, University of Cambridge
  - Professor @ University of Edinburgh
    - Funded by EPSRC, DSTL, EU Horizon 2020
    - Alan Turing Institute Fellow
  - Program Director, Machine Learning @ Samsung AI Cambridge
- Research Area:
  - Deep Learning. Meta-Learning. Data-Efficient & Robust Learning.
- Track record:
  - Over 70 papers in Tier 1 venues of AI, ML, Vision.
  - => CVPR, ICCV, ECCV, ICLR, AAAI, IJCAI, ICML, NeurIPS, T-PAMI, IJCV.
  - Five best paper prizes.
  - Five patents

Outline

- Part I: Data-Efficient Deep Learning
  - Common Regularizers
  - Beyond Regularization
- Part II: Intro to Robust and Explainable Deep Learning
  - Adversarially Robust Learning
  - Uncertainty in Deep Learning
- Part III: Intro to Meta-learning
  - Concept
  - Some examples

Deep Learning Success



DeepMind

"Human-level control through deep reinforcement learning"

letter

Deep Q-Learning

5

Autonomous Vehicles

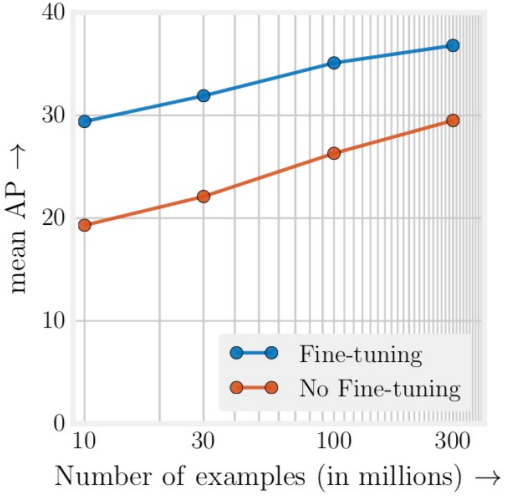
Superhuman Pictionary

Pixelor: Siggraph Asia 2020
Live Demo: http://surrey.ac:9999/

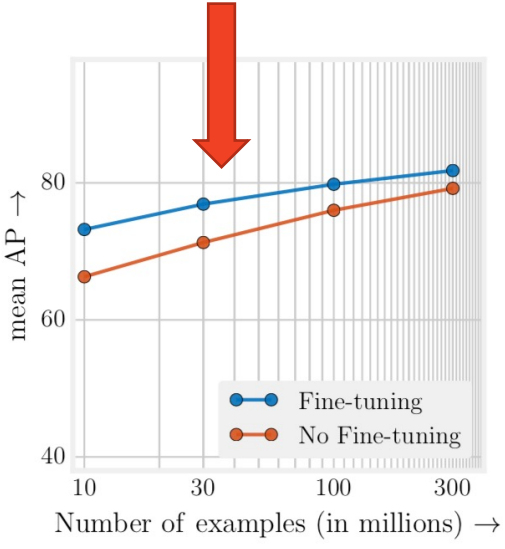
Success Story of Deep Learning Era

- Gather and annotate bigger datasets.
- Train bigger models.

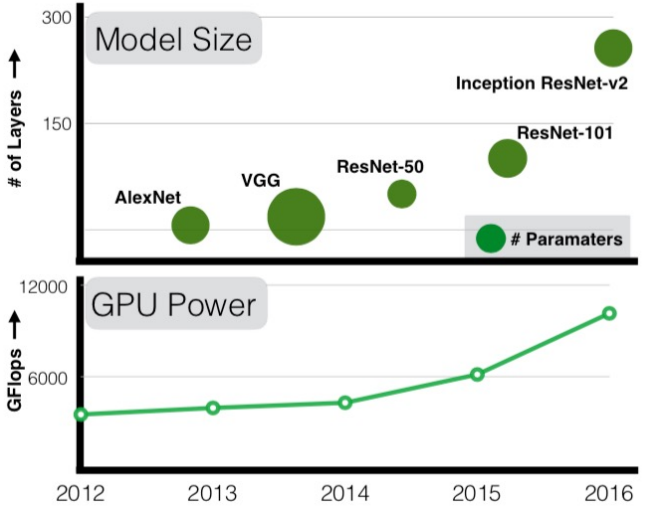
No saturation so far!
Performance grows with log of train data.



JFT-300M => MS-COCO



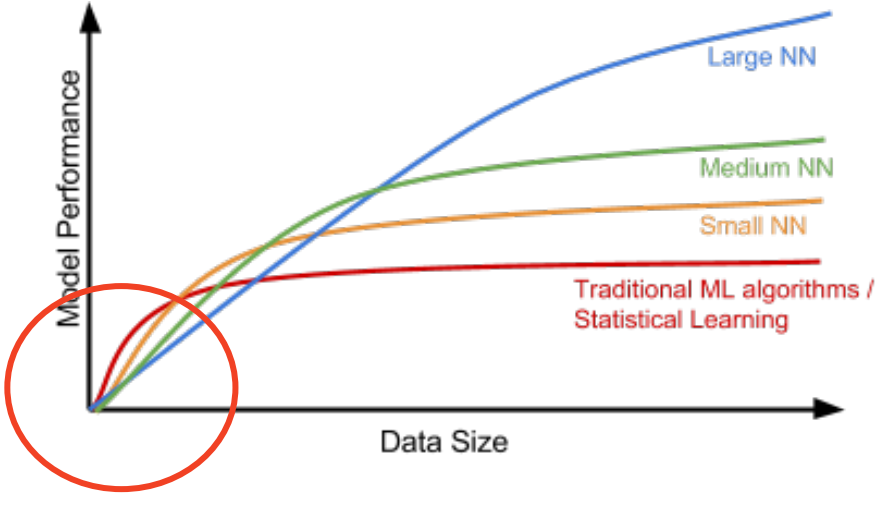
JFT-300M => PASCAL



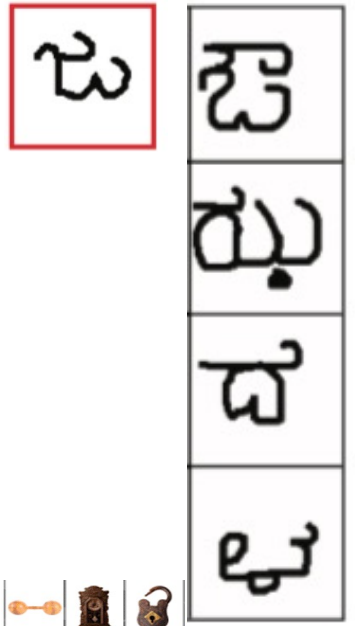
[Sun et al, Revisiting Unreasonable Effectiveness of Data in Deep Learning Era, ICCV'17]

Mechanism of Deep Learning Era Success?

- Gather and annotate bigger datasets.
- Train bigger models.



Do we need another paradigm?

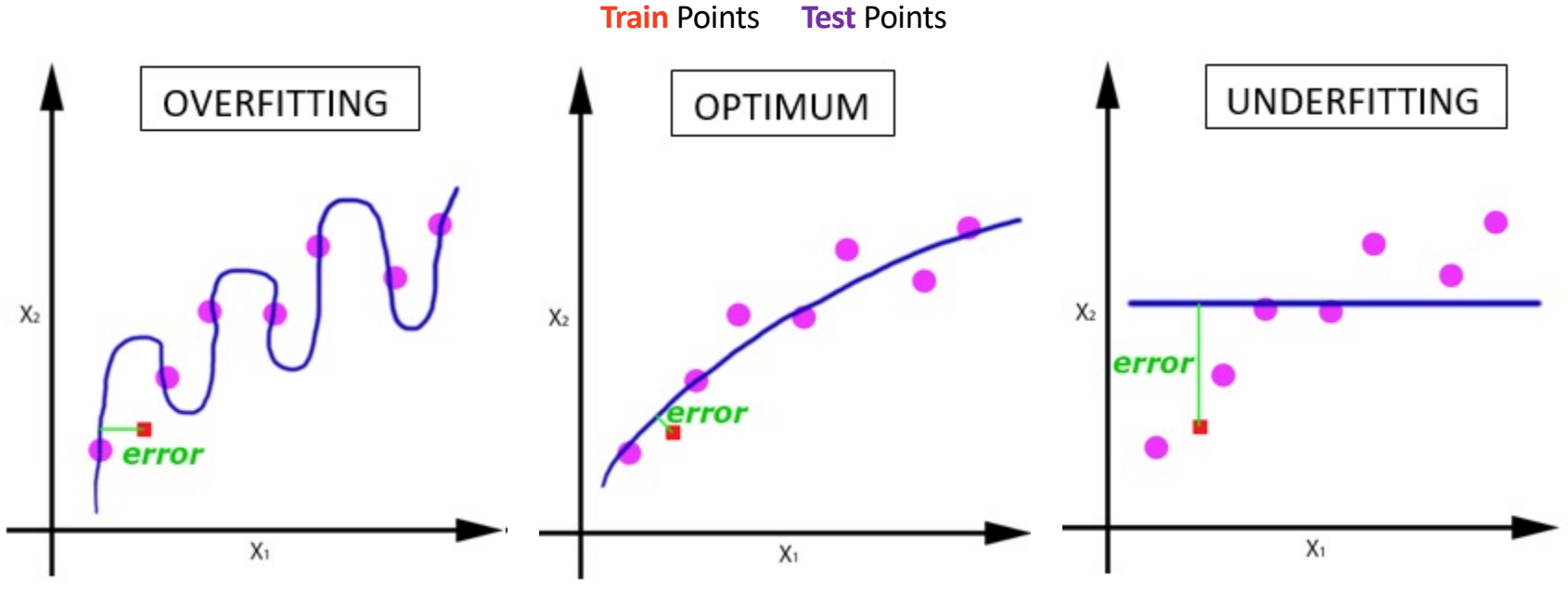


Do we need another paradigm?

- Humans have one shot learning
  - Learn 5 objects per day for first 18 years.
- Long tail of object categories?
  - Emerging categories
- Long tail of domains
  - Underwater, Radar, Sonar, LIDAR, Medical, Satellite, etc
- Defense applications often interested in detecting rarely observed objects/events..

Why is Low-data Learning Hard? Overfitting

- Underfitting vs Overfitting. Linear regression example.



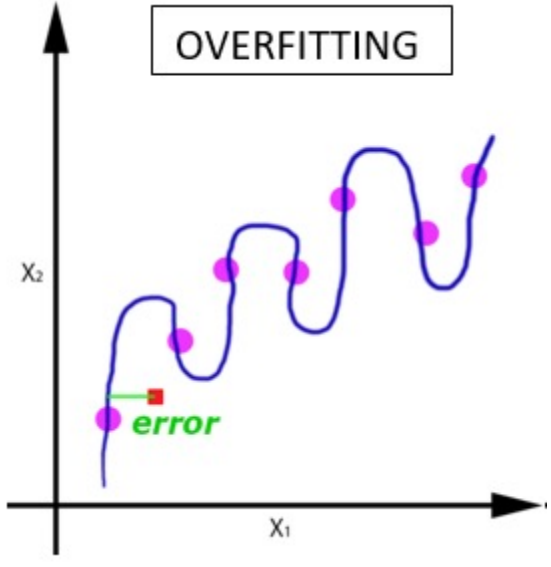
EG: 20th order polynomial.

EG: 2nd order polynomial.

EG: 0th order polynomial.

Why is Low-data Learning Hard? Overfitting

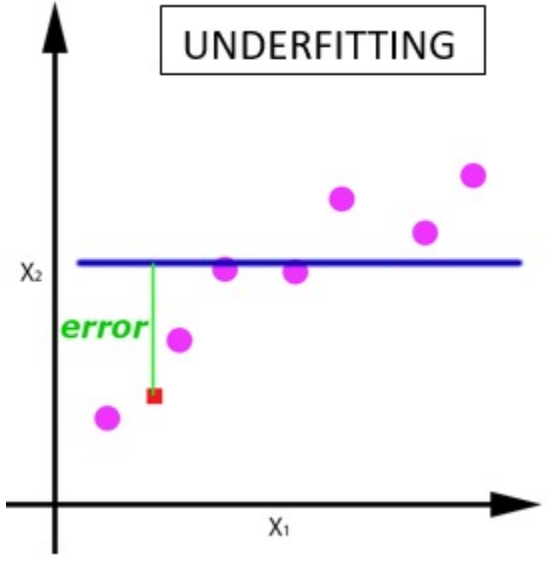
- Underfitting vs Overfitting. Linear regression example.



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

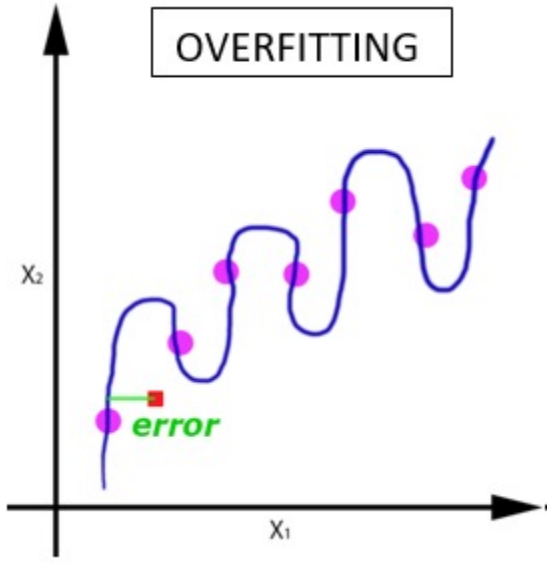
Overfitting?
- Low train error
- High test error

Underfitting?
- High train error
- High test error



Why is Low-data Learning Hard? Overfitting

- Underfitting vs Overfitting. Linear regression example.



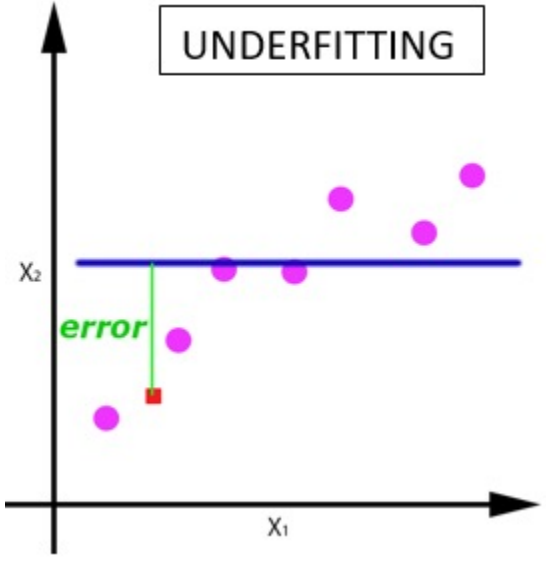
EG: 20th order polynomial.

Classic Solution?

- Try several model complexities
- Evaluate validation set performance of each
- Pick the model with best validation performance

Issue for deep learning?

- Too many complexity parameters in DL (depth, width, non-linearity, etc)
- Few-Shot: Would pick a simple model that doesn't provide deep learning level performance.



EG: 0th order polynomial.

Common Regularisation Techniques

Reducing Data Dependence

Common techniques for overfitting reduction

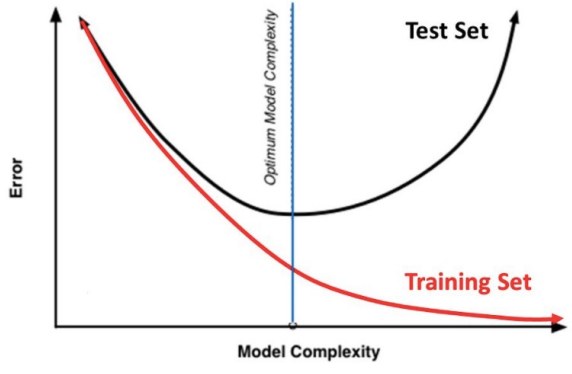
- Weight decay / L2 regularization
- Early Stopping
- Label Smoothing
- Data Augmentation
- Mixup

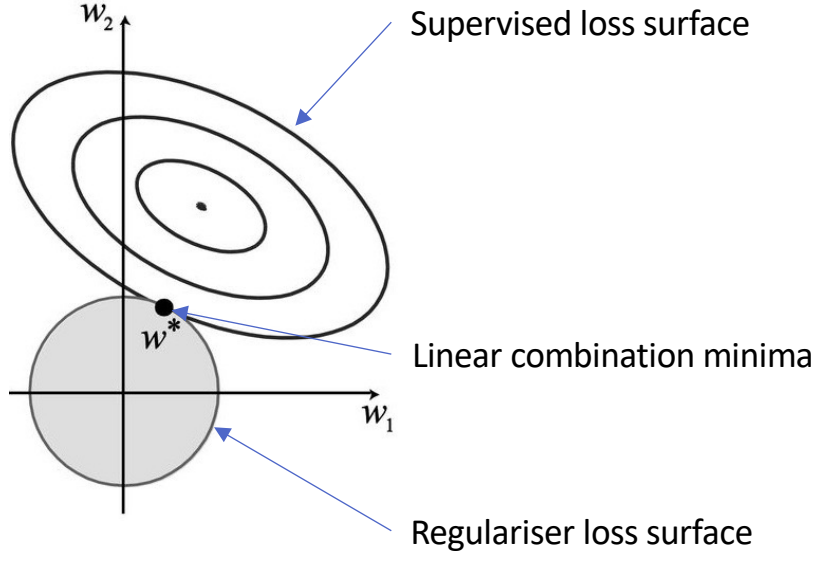
Weight Decay & L2 Regularization

Weight decay. L2 regularisation.

- Other things being equal, prefer weights near zero.

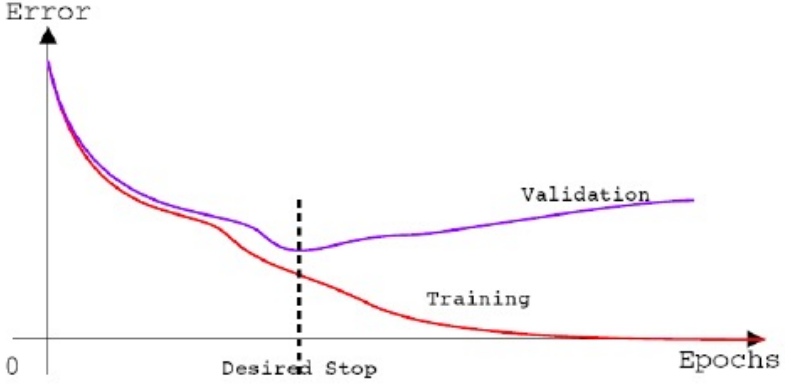
$$\min_{\boldsymbol{w}} \mathcal{L}(D; \boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_2^2$$





Early Stopping

- With gradient descent, the model gradually improves its fit to the data.
  - It takes “time” (iterations of gradient descent) for overfitting to happen.
  - => Try to stop early, before overfitting happens
- Early Stopping
  - Observe validation set error during training (proxy for testing/generalisation error).
  - Stop when validation set error starts to increase.



Label Smoothing

- Conventional Cross-Entropy Learning Objective

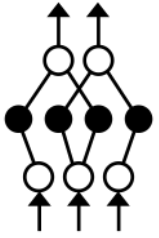
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0]

$\boldsymbol{y}_{k=y} = 1, \quad \boldsymbol{y}_{k\neq y} = 0$

[0.1, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.7, 0.0, 0.0]

$$L(\hat{\boldsymbol{p}}, \boldsymbol{y}) = -\sum_k \boldsymbol{y}_k \log \hat{\boldsymbol{p}}_k$$

$$\hat{\boldsymbol{p}} = softmax(Wh(\boldsymbol{x}))$$



- Label Smoothing

[0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.91, 0.01, 0.01]

$\boldsymbol{y}_{k=y} = 1 - \epsilon + \epsilon/K, \quad \boldsymbol{y}_{k\neq y} = \epsilon/K$

[0.1, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.7, 0.0, 0.0]

$$L(\hat{\boldsymbol{p}}, \boldsymbol{y}) = -\sum_k \boldsymbol{y}_k \log \hat{\boldsymbol{p}}_k$$

[Muller, NeurIPS'19, When does label smoothing help?]

Data Augmentation

EG: SpecAugment

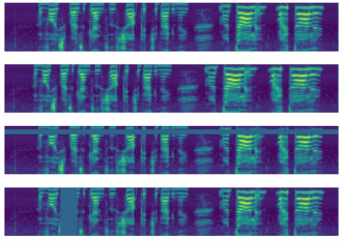
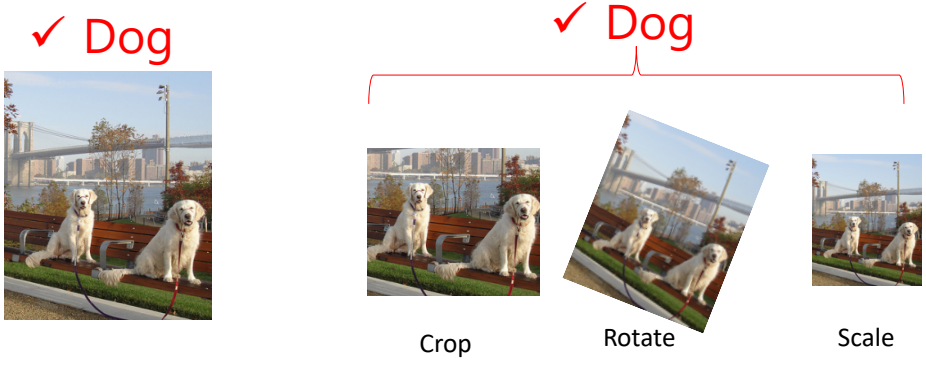


Figure 1: *Augmentations applied to the base input, given at the top. From top to bottom, the figures depict the log mel spectrogram of the base input with no augmentation, time warp, frequency masking and time masking applied.*

[Park, InterSpeech, 19]

- Data Augmentation

$$L(\boldsymbol{x}, \boldsymbol{y}) = -\sum_k \boldsymbol{y}_k \log \hat{\boldsymbol{p}}_k(x) \quad \Rightarrow \quad L(\boldsymbol{x}, \boldsymbol{y}) = -\sum_k \boldsymbol{y}_k \log \hat{\boldsymbol{p}}_k(A_\epsilon(x)))$$



- Key: Augmentation must preserve desired semantics.
- Class preserving augmentations are domain knowledge:
  - => A route to implicitly providing expert prior knowledge.

Mixup Augmentation

Y=[1.0, 0.0, 0.0]

Y=[0.0, 1.0, 0.0]

- Conventional Supervised Learning

$$\min_{\boldsymbol{w}} E_{x,y\sim D} L(y, f_w(x))$$

- Mixup

$$\min_{\boldsymbol{w}} E_{x_1,y_1,x_2,y_2\sim D,\lambda\in[0,1]} L(\lambda y_1 + (1-\lambda)y_2, f_w(\lambda x_1 + (1-\lambda)x_2))$$

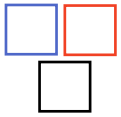
[Zhang, ICLR'18, mixup: Beyond Empirical Risk Minimization]

Beyond Regularisation: Other Data Sources

- You can only get so far using regularization and cross-validation in a closed world and tabula-rasa learning.
  - Luckily there are often diverse data sources available.

- Beyond Closed-World/Tabula-Rasa Assumption:
  - Transfer Learning,
  - Few-Shot learning,
  - Domain Adaptation,
  - Domain Generalisation,
  - Semi-Supervised Learning,
  - Self-supervised Learning,
  - Meta-Learning

Faces of Data Sparsity



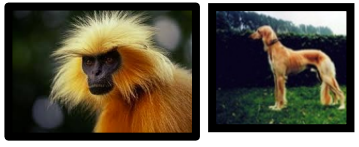
Annotated

Unannotated

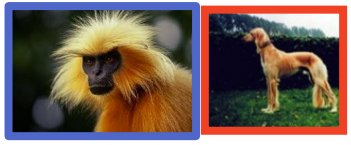
Target: Solve dog vs monkey classification

Extra Data:

No Labels



Few Labels



Domain Adaptation
Domain Generalisation

Transfer learning

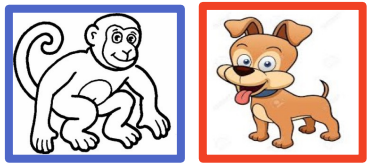
Label Noise Robust Learning

Transfer learning
(Meta learning)

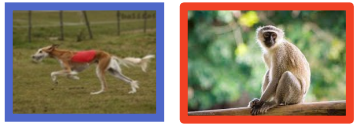
Self-supervised learning

Semi - supervised learning

Same Categories/ Domain Shift

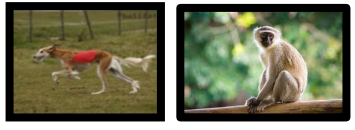


Inaccurate labels

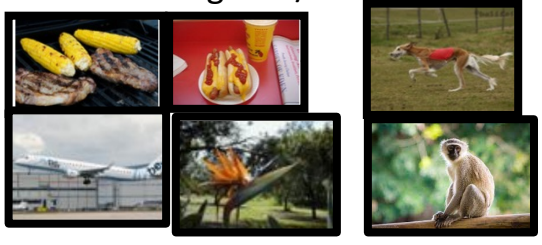


Other Categories

Same Categories/Unlabeled



Other Categories/Unlabeled



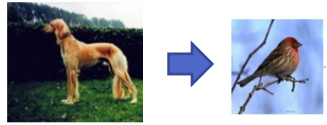
Some Typical Problem Settings/Algorithms

Situation	Target* (During training)	Auxiliary	Assumption
(Supervised) Transfer Learning	$\{X_t, Y_t\}$	$\{X_s, Y_s\}$	$p(X_s) \neq p(X_t)$ Or $p(Y_s\|X_s) \neq p(Y_t\|X_t)$ Or $\mathcal{Y}_s \neq \mathcal{Y}_t$
Semi-supervised learning	$\{X_t, Y_t\}$	$\{X_s\}$	$\mathcal{Y}_s = \mathcal{Y}_t$ $p(Y_s\|X_s) = p(Y_t\|X_t)$
Self-supervised learning	$\{X_t, Y_t\}$	$\{X_s\}$	$\mathcal{Y}_s \neq \mathcal{Y}_t$
Domain Adaptation	$\{X_t\}$	$\{X_s, Y_s\}$	$p(X_s) \neq p(X_t)$ Usually $p(Y_s\|X_s) = p(Y_t\|X_t)$, $\mathcal{Y}_s = \mathcal{Y}_t$
Domain Generalisation		$\{X_s, Y_s\}$	$p(X_s) \neq p(X_t)$ Usually $p(Y_s\|X_s) = p(Y_t\|X_t)$, $\mathcal{Y}_s = \mathcal{Y}_t$
Noisy Labels	$\{X_t, Y_t\}$	$\{X_s, \hat{Y}_s\}$	$\hat{Y}_s = Y_s + \epsilon$

Task-Shift:

$p(Y_s|X_s) \neq p(Y_t|X_t)$

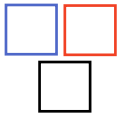
Or $\mathcal{Y}_s \neq \mathcal{Y}_t$



Domain-Shift:

$p(X_s) \neq p(X_t)$

Faces of Data Sparsity



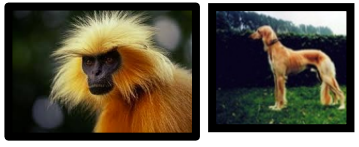
Annotated

Unannotated

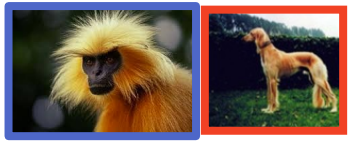
Target: Solve dog vs monkey classification

Extra Data:

No Labels



Few Labels



Domain Adaptation
Domain Generalisation

Transfer learning

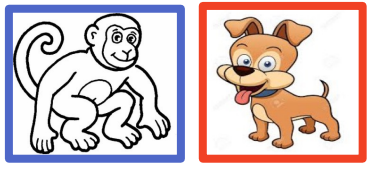
Label Noise Robust Learning

Transfer learning
(Meta learning)

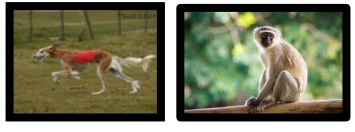
Self-supervised learning

Semi - supervised learning

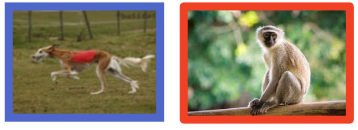
Same Categories/ Domain Shift



Same Categories/Unlabeled

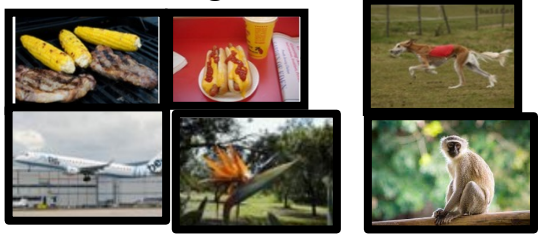


Inaccurate labels



Other Categories

Other Categories/Unlabeled



Reducing Data Dependence

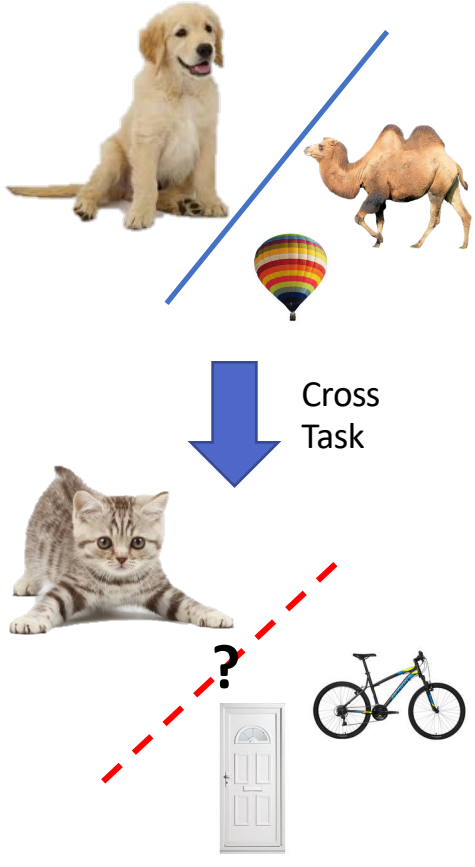
Transfer Learning

- Transfer Learning: "*The application of skills, knowledge, and/or attitudes that were learned in one situation to another learning situation*" (Perkins, 1992).
- Note:
  - Transfer Learning ≠ "Fine-Tuning"
  - Fine Tuning⊂ Transfer Learning

Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

- Application:
  - Across Task: $p(Y_s|X_s) \neq p(Y_t|X_t)$
    - Change of label-space. $Y_s \neq Y_t$
  - Across Domain
    - Change of data statistics. $p(X_s) \neq p(X_t)$

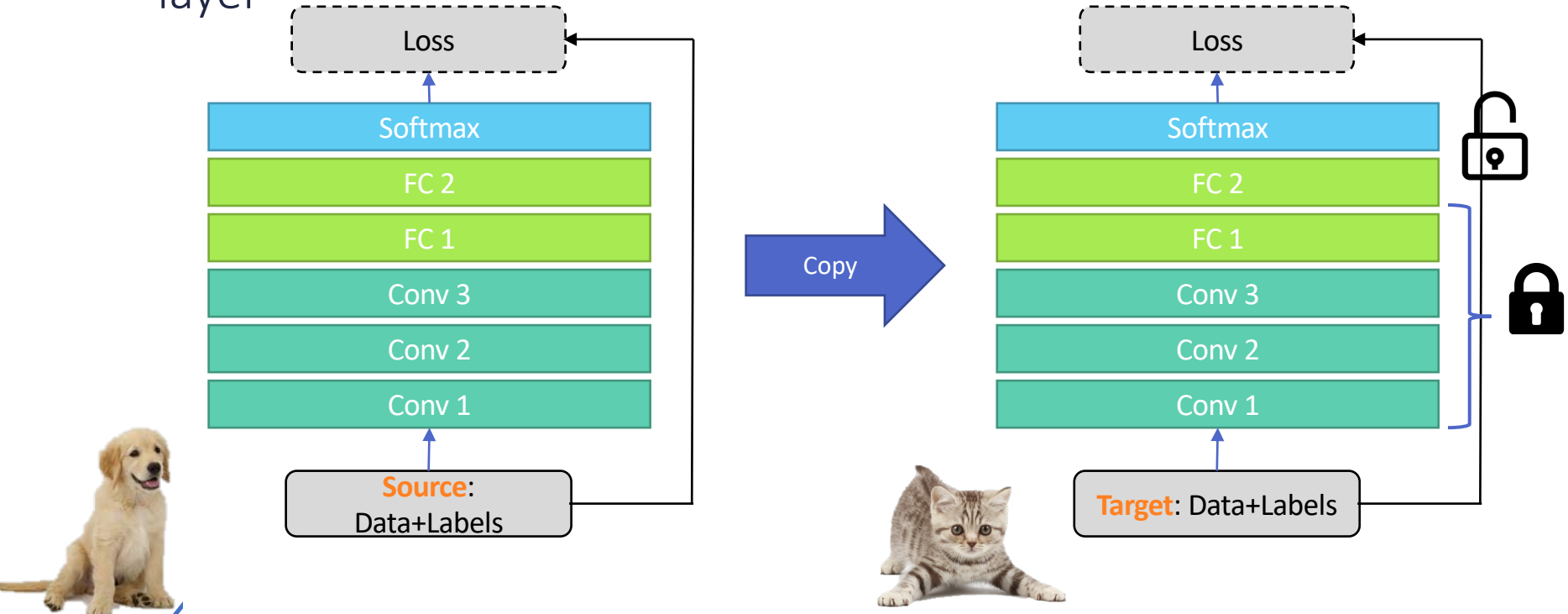


RGB => IR

Reducing Data Dependence: Transfer

Transfer Learning: Linear Readout

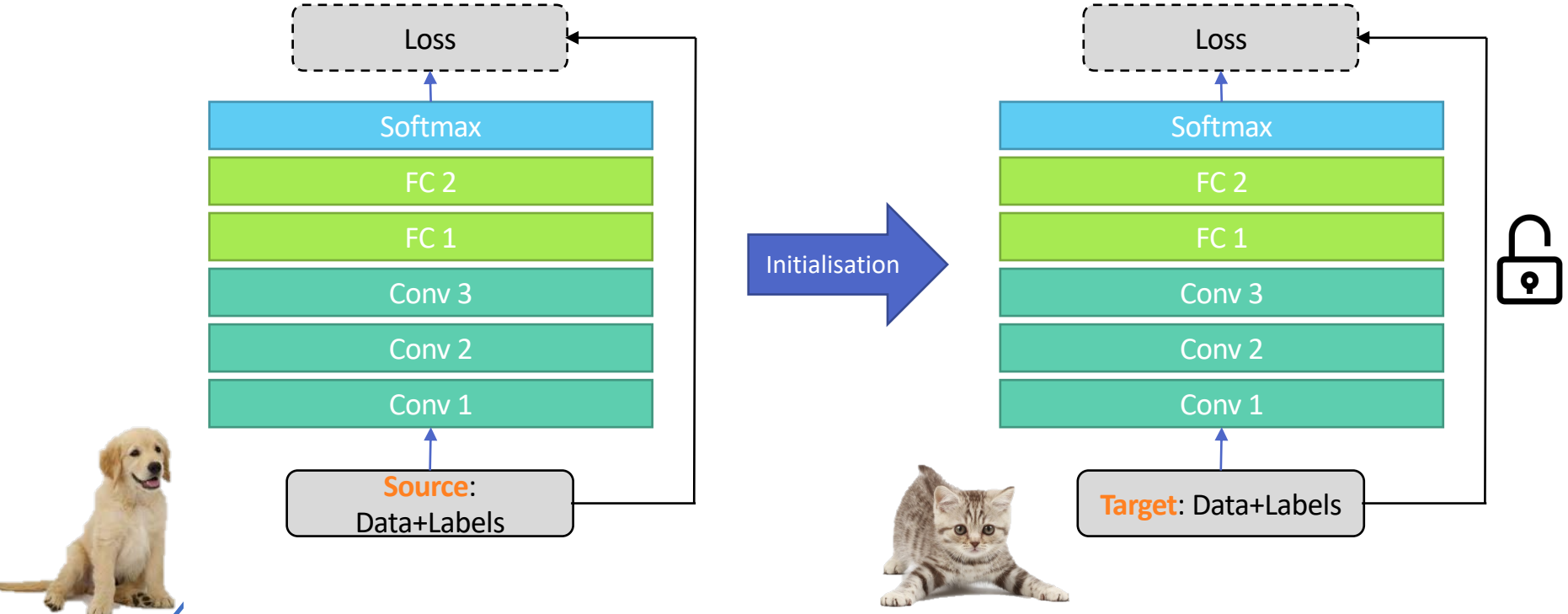
- Fix target feature extractor using source, and training classifier layer



Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

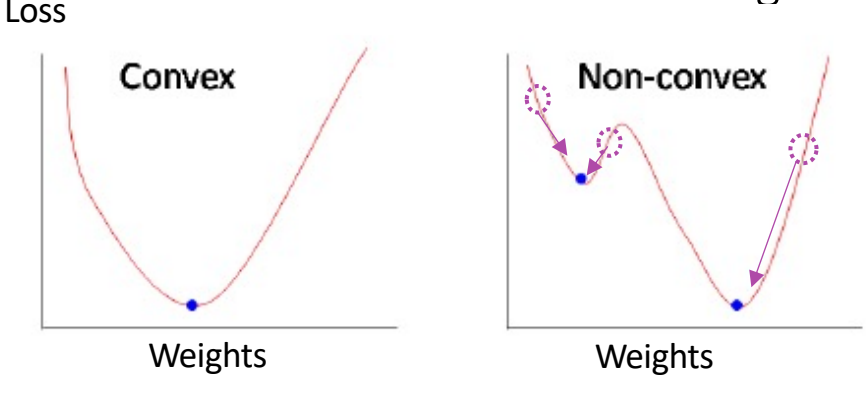
- Initialise target parameters using source, and continue training

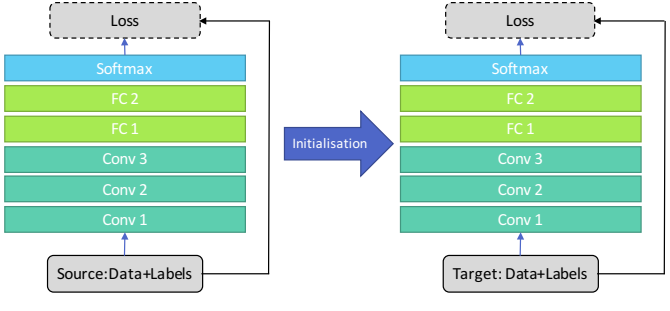


Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

- Why does fine-tuning work?
  - Neural network optimisation is non-convex.
  - With small learning rate, target task parameters do not change much.
  - Transfer initialization effectively regularizes target weights towards source weights, rather than towards zero.
    - Assume source task is relevant to target task.
      - => Better chance of good minima.





Reducing Data Dependence: Transfer

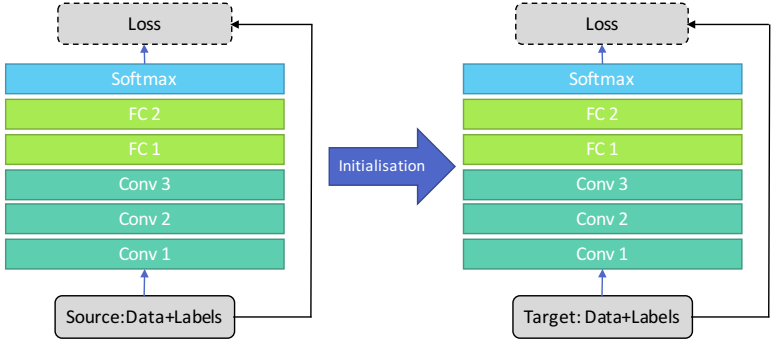
Transfer Learning: Fine-Tuning

- Very similar to explicit source→target regularisation.
  - Used in many classic learning methods(*).

$$\min_{\boldsymbol{w}_t} \mathcal{L}(D_t; \boldsymbol{w}_t) + \lambda \|\boldsymbol{w}_t - \boldsymbol{w}_s\|_2^2$$

Compare:

$$\min_{\boldsymbol{w}} \bar{\mathcal{L}} = \mathcal{L}(D; \boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_2^2$$
$$\min_{\boldsymbol{w}} \bar{\mathcal{L}} = \mathcal{L}(D; \boldsymbol{w}) + \lambda \|\boldsymbol{w} - \boldsymbol{0}\|_2^2$$



[Improving SVM Accuracy by Training on Auxiliary Data Sources, ICML'04]
[Cross-domain video concept detection using adaptive svms, ACM MM'07]

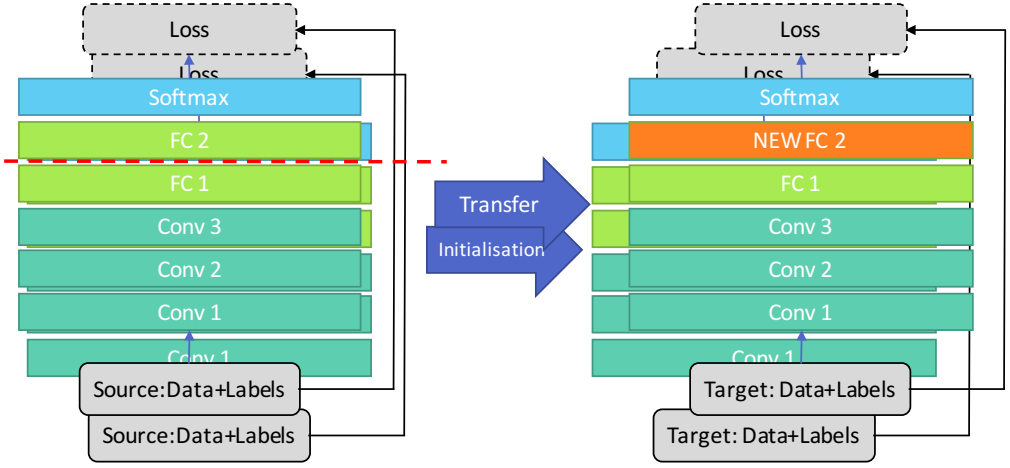
Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

- Assumption:
  - Source task relevant to target.
- Practical Considerations. Questions:
  - How to control relevance degree?
  - Which layers to transfer?
  - What learning rate to use?
- Typical:
  - Relearn top while freeze bottom.
  - Then tune all w/ LR proportional to depth

$$\min_{\boldsymbol{w}} \mathcal{L}(D_t; \boldsymbol{w}_t) + \lambda \|\boldsymbol{w}_t - \boldsymbol{w}_s\|_2^2$$

Question:
Always good practice?



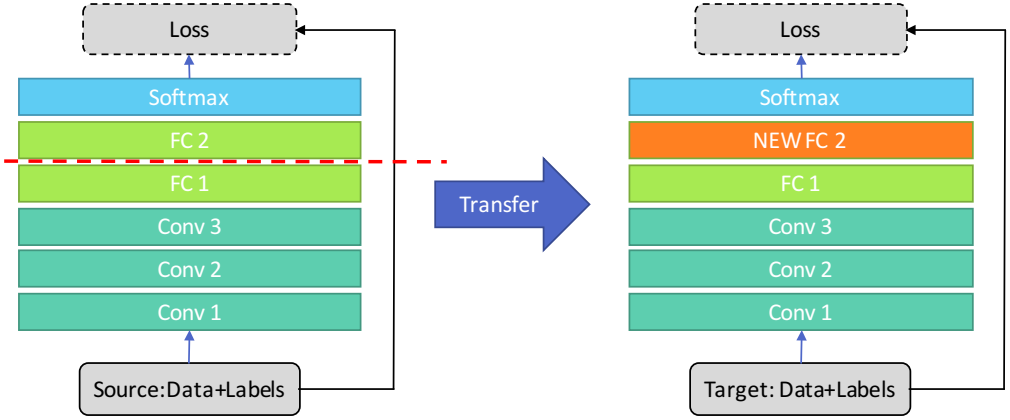
[How transferable are features in deep neural networks?, NIPS'14]

Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

- Assumption:
  - Source task relevant to target.
- Practical Considerations. Questions:
  - How to control relevance degree?
  - Which layers to transfer?
  - What learning rate to use?
- Typical:
  - Relearn top while freeze bottom.
  - Then tune all w/ LR proportional to depth

$$\min_{\boldsymbol{w}} \mathcal{L}(D_t; \boldsymbol{w}_t) + \lambda \|\boldsymbol{w}_t - \boldsymbol{w}_s\|_2^2$$



Only for cross-task.

Assumption:
- Relevance proportional to depth.

[How transferable are features in deep neural networks?, NIPS'14]

A State of the Art Solution: UDRC work

Classic Transfer Learning: Penalty Regularized

$$\min_{\boldsymbol{w}} \mathcal{L}(D_t; \boldsymbol{w}_t) + \lambda \|\boldsymbol{w}_t - \boldsymbol{w}_s\|_2^2$$

Transfer Learning: Constraint Reguarized (Euclidean)

$$\min_{\boldsymbol{w}} \mathcal{L}(D_t; \boldsymbol{w}_t) \quad \text{s.t. } \|\boldsymbol{w}_t - \boldsymbol{w}_s\|_F < \lambda$$

Transfer Learning: Constraint Reguarized (MARS)

$$\min_{\boldsymbol{w}} \mathcal{L}(D_t; \boldsymbol{w}_t) \quad \text{s.t.} \|\boldsymbol{w}_t - \boldsymbol{w}_s\|_\infty < \lambda$$

Max Distance Moved During Fine tuning

- Enables a generalization bound:

Max Distance Moved

Input dim Layers Max Weight Norm

$$\mathbb{E}_{(\vec{x},y)}[l(f(\vec{x}),y)] \leq \frac{1}{m}\sum_{i=1}^{m} l(f(\vec{x}_i),y_i) + \frac{4\sqrt{\log(2d)}c\rho C_\infty \sum_{j=1}^{L} \frac{D_j^\infty}{B_j^\infty} \prod_{j=1}^{L} 2B_j^\infty}{\sqrt{m}} + 3\sqrt{\frac{\log(2/\delta)}{2m}},$$

Test Error Train Error Bound

Num Train Examples

[Gouk et al, ICLR'2021, Distance-Based Regularisation of Deep Networks for Fine-Tuning] * UDRC invention!

Transfer: Challengens

- What if the inputs are heterogeneous
  - (EG: Task 1: RGB, Task 2: Infra Red)?

- How to know if a source task is relevant?
  - How to prevent negative transfer if source is irrelevant?

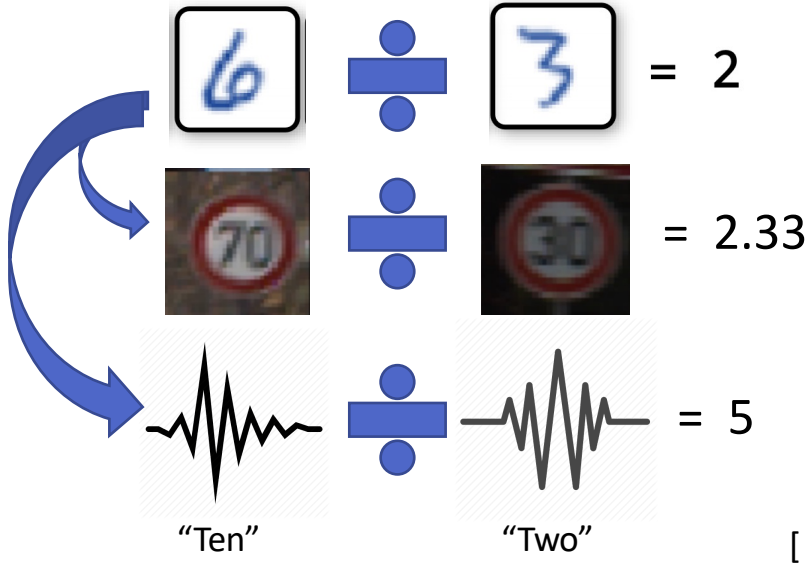
- How to select the relevant source among many?

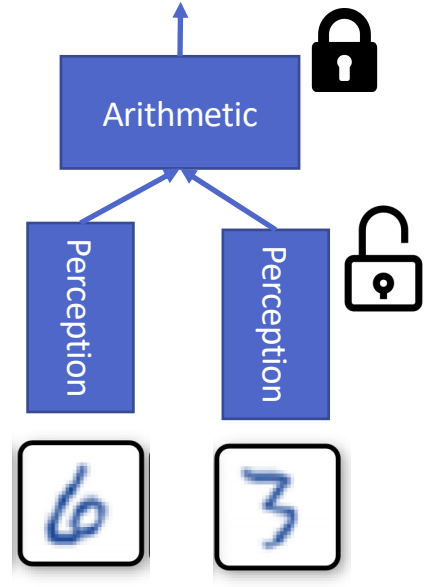
Reducing Data Dependence: Transfer

RGB => IR

Transfer Learning: Fine-Tuning

- Issues: What if the inputs are heterogeneous?
  - Should learning rate always be proportional to depth?
  - EG: Perceptual arithmetic network





[EG: HOUDINI: Lifelong Learning as Program Synthesis, NIPS'18]

Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

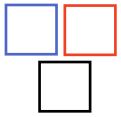
- Issues:
  - How to select the relevant source among many?
- For linear models….
  - Optimising target model $\boldsymbol{w}$, assuming a set of potentially relevant sources $\{\boldsymbol{w}_k\}$:

$$\min_{\boldsymbol{w}}[\|\boldsymbol{y} - \boldsymbol{w}X\| + \min_k \|\boldsymbol{w} - \boldsymbol{w}_k\|]$$

  - …. not so straightforward with deep models
    - => Often resort to cross-validation

E.g., Evgeniou et al, JMLR, 2005
E.g., Kang et al, ICML, 2011

Faces of Data Sparsity



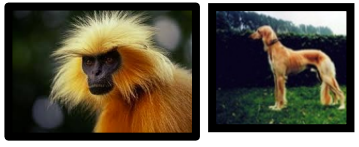
Annotated

Unannotated

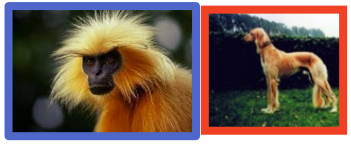
Target: Solve dog vs monkey classification

Extra Data:

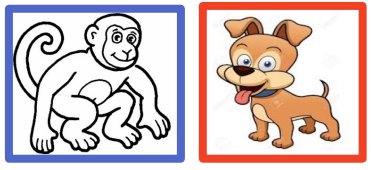
No Labels



Few Labels



Same Categories/ Domain Shift

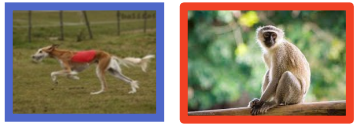


Domain Adaptation
Domain Generalisation

Transfer learning

Label Noise Robust Learning

Inaccurate labels



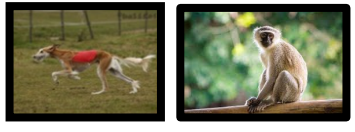
Transfer learning
(Meta learning)

Other Categories

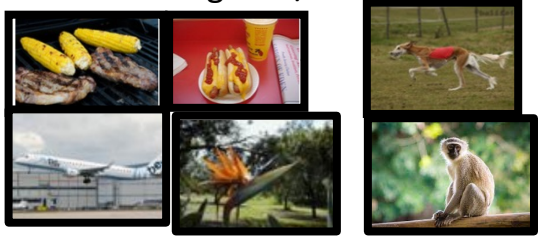
Self-supervised
learning

Semi - supervised
learning

Same Categories/Unlabeled

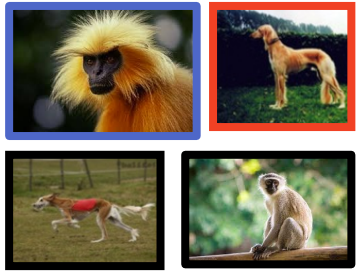


Other Categories/Unlabeled



Semi-Supervised: Problem

- Problem:
  - Learn from mix of labeled & unlabeled data (of same label-space).
  - When annotation bottleneck rather than raw data bottleneck.

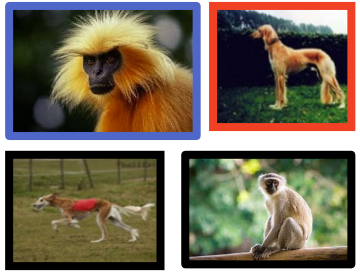


- Typical supervised learning with $\{X_t, Y_t\}$: $\min_w \sum_i L(y_t^i, p_w(y|x_t^i))$
  - What to do with $\{X_s\}$?

Semi-supervised learning	$\{X_t, Y_t\}$	$\{X_s\}$	$\mathcal{Y}_s = \mathcal{Y}_t$

Semi-Supervised: A Classic Solution

- Entropy Regularisation:



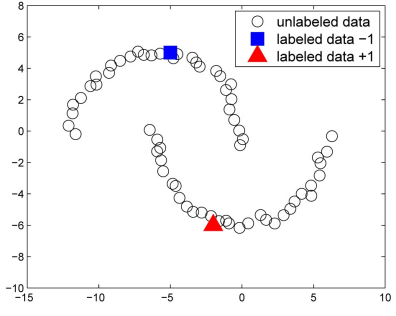
$$\min_{\boldsymbol{w}} \sum_{i} L(y_t^i, p_w(y|x_t^i))$$

Target/Labeled

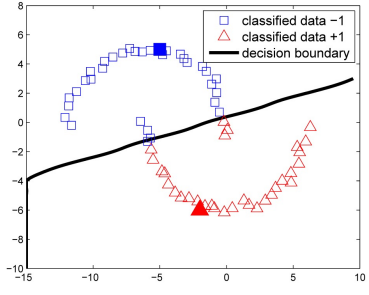
Source/Unlabeled

$$\sum_{j,k} p_w(y_k|x_s^j) log\, p_w(y_k|x_s^j)$$

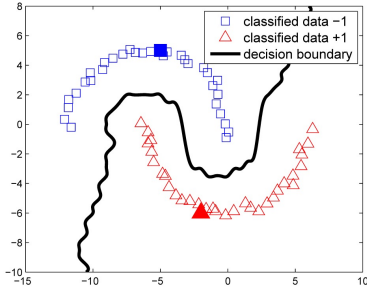
Example Problem



Supervised



Semi-Supervised

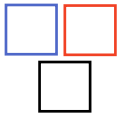


[Grandvalet, NIPS'04, Semi-supervised Learning by Entropy Minimization]

Semi-Supervised: State of the Art

- Lots of Sophisticated Solutions Recently
  - Mean Teacher - https://arxiv.org/abs/1703.01780
  - Mix Match - https://arxiv.org/abs/1905.02249
  - Noisy Student
    - Audio - https://arxiv.org/abs/2005.09629
    - Images - https://arxiv.org/abs/1911.04252

Faces of Data Sparsity



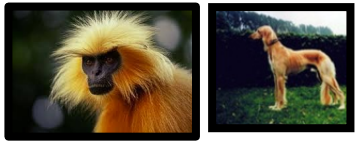
Annotated

Unannotated

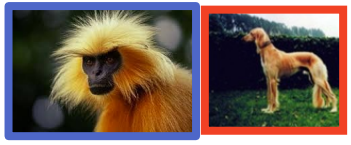
Target: Solve dog vs monkey classification

Extra Data:

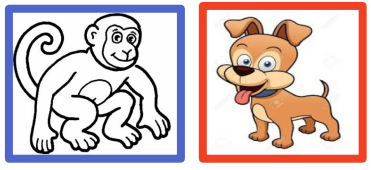
No Labels



Few Labels



Same Categories/ Domain Shift

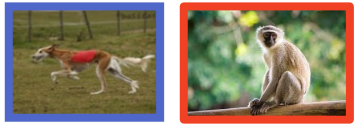


Domain Adaptation
Domain Generalisation

Transfer learning

Label Noise Robust Learning

Inaccurate labels



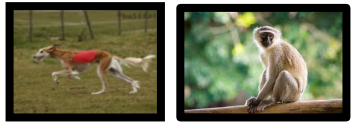
Transfer learning
(Meta learning)

Other Categories

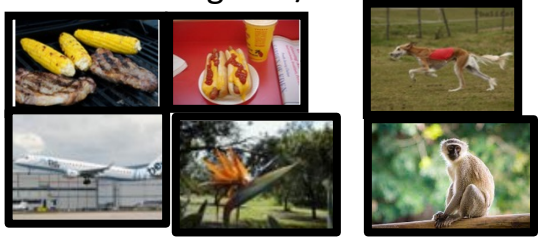
Self-supervised
learning

Semi - supervised
learning

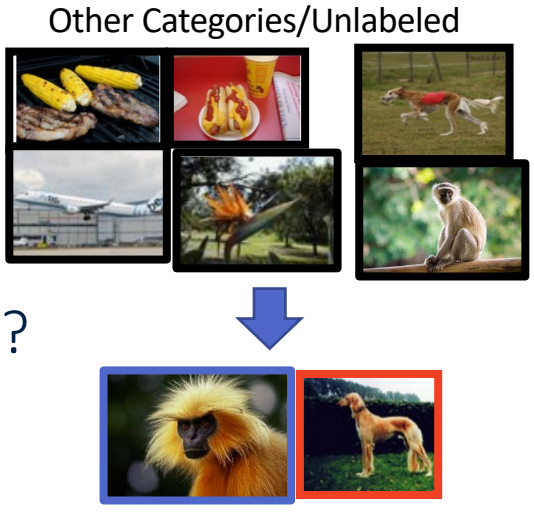
Same Categories/Unlabeled



Other Categories/Unlabeled



Self-Supervision: Problem



- Problem:
  - How to exploit unlabelled task-irrelevant data?

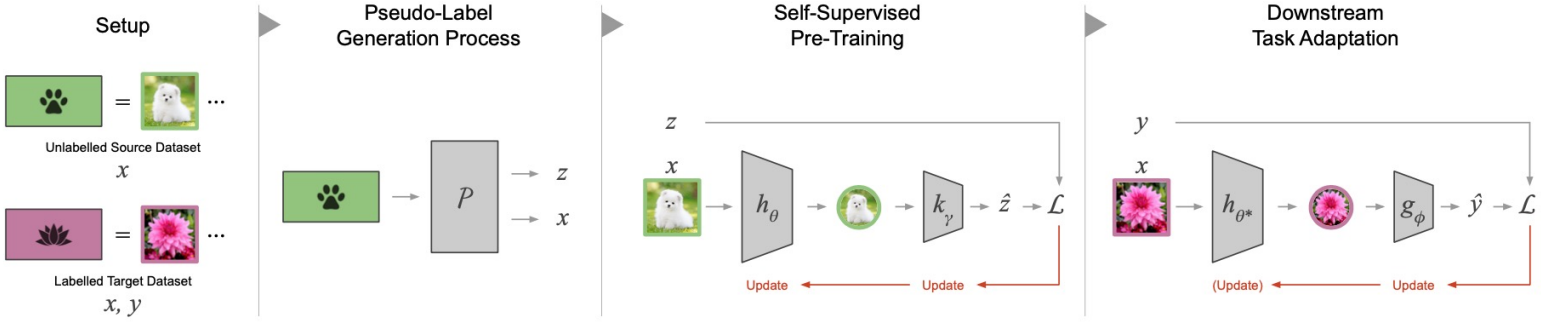
- Typical supervised learning with $\{X_t, Y_t\}$:
  - What to do with $\{X_s\}$?

$$\min_w \sum_i L(y_t^i, f_w(x_t^i))$$

Self-supervised learning	$\{X_t, Y_t\}$	$\{X_s\}$	$\mathcal{Y}_s \neq \mathcal{Y}_t$

Self-Supervision: Idea

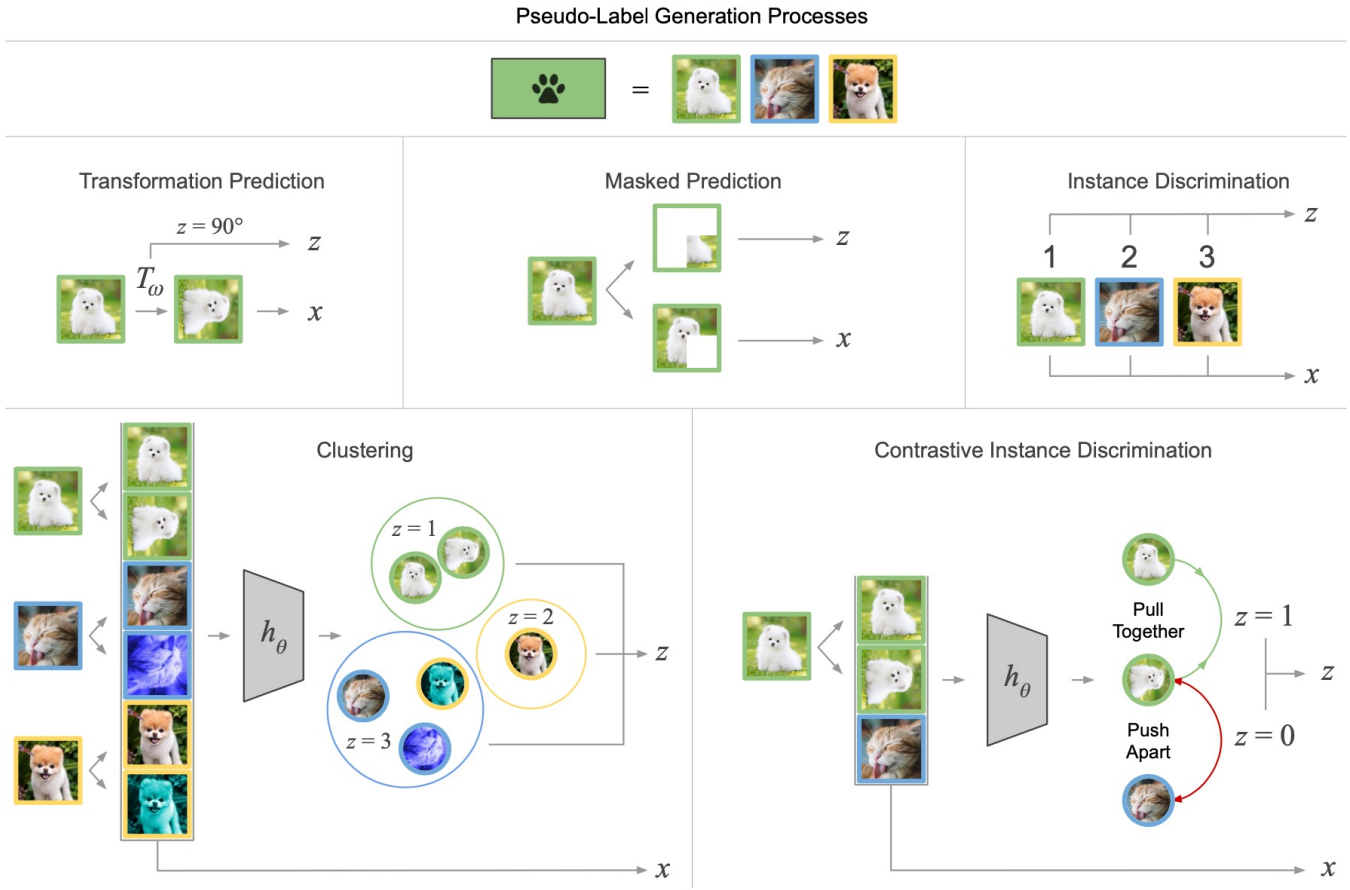
- Pretext-tasks: Programmatic generation of tasks for the unlabeled data.



- Some parameters θ shared between pretext and real task
  - Multi-task or transfer.

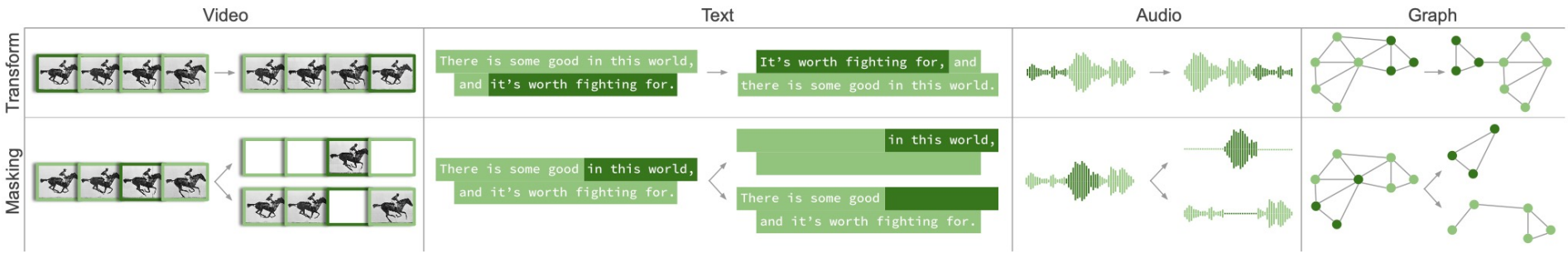
$$\min_{\theta,\gamma} E_{x_S}[L\left(P(x_s), f_{\theta,\gamma}(x_s)\right)] \quad \Rightarrow \quad \min_{\theta,\phi} E_{x_t}[L\left(y_t, f_{\theta,\phi}(x_t)\right)]$$

Self-Supervision: Pretexts



$$\min_{\theta,\gamma} E_{x_S}[L\left(P(x_s), f_{\theta,\gamma}(x_s)\right)]$$

Self-Supervision: Pretexts



$$\min_{\theta,\gamma} E_{x_S}[L\left(P(x_s), f_{\theta,\gamma}(x_s)\right)]$$

Self-Supervision: Results

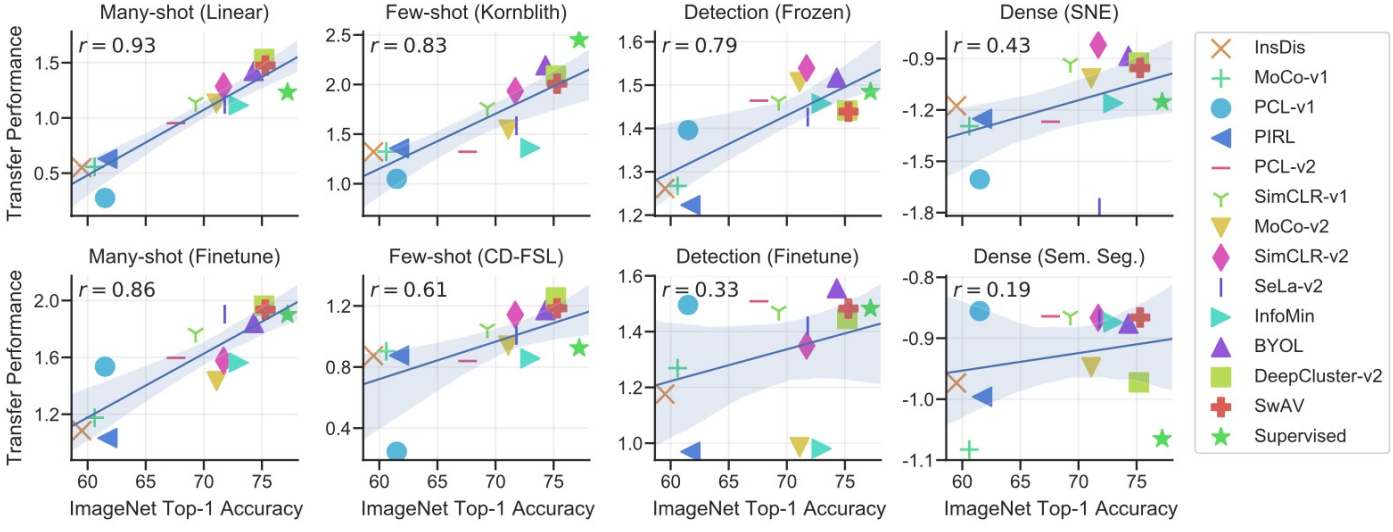


Figure 1. Transfer performance is highly correlated with ImageNet performance for many-shot recognition but increasingly less correlated for few-shot recognition, object detection and dense prediction. On the x-axes we plot ImageNet top-1 accuracy and on the y-axes the average transfer log-odds. The gradients of the regression lines describe the correlation, with confidence intervals in shaded areas. For perfect correlation, the ideal line is a positive slope diagonal. Correlation coefficients (Pearson's r) are shown in the top left of each plot.

- Self-supervision outperforms supervised transfer learning even without exploiting labels!

[Ericsson, Gouk, Hospedales, CVPR2021, How Well Do Self-Supervised Models Transfer?] *UDRC work!

Domain Adaptation: Problem

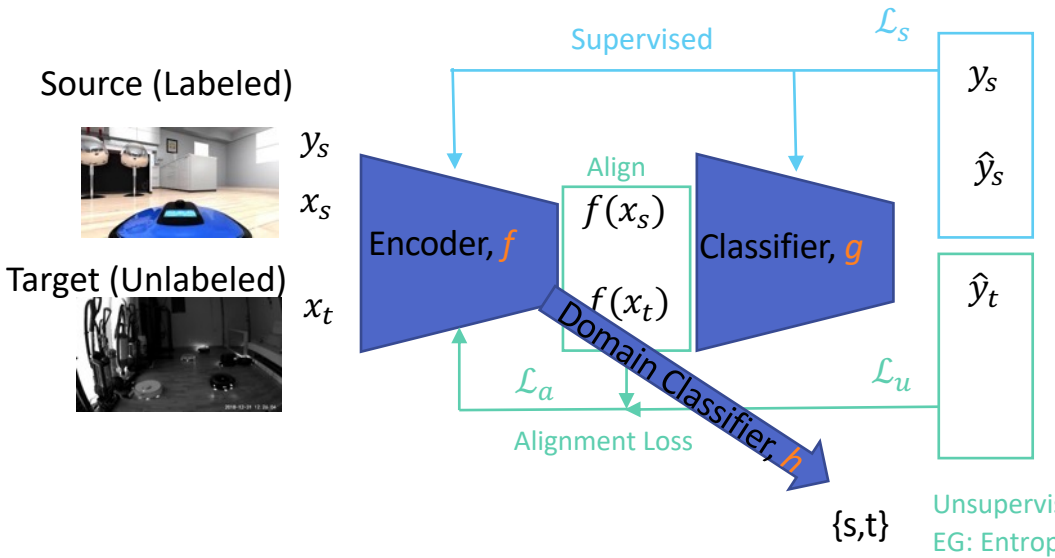
- Problem:
  - Learn from mix of labelled & unlabelled data (same label-space, different domain).
  - Often some domains are easier to collect/annotate than others.

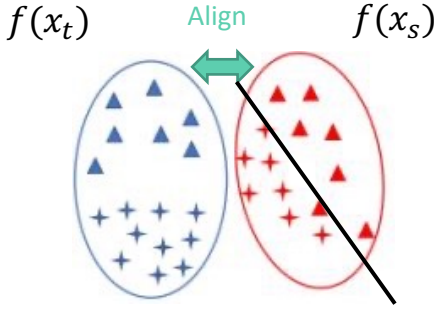
- Typical supervised learning:
  - What to do with $\{X_t\}$?

$$\min_{\boldsymbol{w}} \sum_i L(y_s^i, p_w(y|x_s^i))$$

Domain Adaptation	$\{X_t\}$	$\{X_s, Y_s\}$	$p(X_s) \neq p(X_t)$ Usually $p(Y_s\|X_s) = p(Y_t\|X_t)$, $\mathcal{Y}_s = \mathcal{Y}_t$

Domain Adaptation: Standard Solutions





$$\mathcal{L}_{total} = \mathcal{L}_s + \mathcal{L}_a + \mathcal{L}_u$$

$$\mathcal{L}_s: \text{argmin}_{f,g} E_{x_s}[L(y_s, g(f(x_s)))]$$

$$\mathcal{L}_a: \text{argmin}_{f,h} E_{x_d,d}\left[L(\frac{1}{2}, \bar{h}(f(x_d)) + L(d, h(\bar{f}(x_d)))\right]$$

$$\mathcal{L}_u: \text{argmin}_{f,g} E_{x_t}[H(g(f(x_t)))]$$

[EG: Tzeng, CVPR'17, Adversarial Discriminative Domain Adaptation]

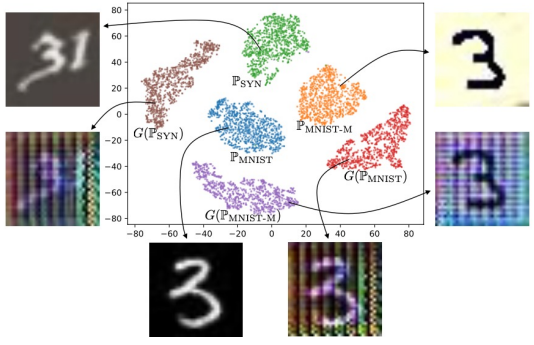
Domain Generalisation: Problem

- Problem:
  - Deployment/target domain is shifted wrt train domain(s).
  - No target data available for training (e.g., future).
  - => Need to build a robust model.

Domain Generalisation	**None**	$\{X_s, Y_s\}$	$p(X_s) \neq p(X_t)$ **Usually** $p(Y_s\|X_s) = p(Y_t\|X_t)$, $Y_s = Y_t$

Domain Generalisation: A Solution

- Many complicated and controversial solutions.
  - Simple => Extend Mixup.



$$\min_{\boldsymbol{w}} E_{x,y\sim D} L(y, f_w(x))$$

$$\hookrightarrow \min_{\boldsymbol{w}} E_{x_1,y_1,x_2,y_2\sim D,\lambda\in[0,1]} L(\lambda y_1 + (1-\lambda)y_2, f_w(\lambda x_1 + (1-\lambda)x_2))$$

  - Domain Mixup.

$$\min_{\boldsymbol{w}} E_D E_{x,y\sim D} L(y, f_w(x))$$

$$\hookrightarrow \min_{\boldsymbol{w}} E_{D_1 D_2\sim D} E_{\substack{x_1,y_1\sim D_1 \\ x_2,y_2\sim D_2}} L(\lambda y_1 + (1-\lambda)y_2, f_w(\lambda x_1 + (1-\lambda)x_2))$$

[Wang et al, ICASSP'20, Heterogeneous domain generalization via domain mixup]

Label Noise: Problem

Inaccurate labels

- Situation:
  - Some subset of your data has flawed annotation.
  - Context: When annotation is automated, or hard to verify.
- Issue:
  - With this additional noise source, overfitting issues can be much worse. (Overfit to label errors => V.Poor test performance)

Label Noise: Some Solutions

- Simple Solutions:
  - Cross-entropy loss is highly vulnerable to label-noise.
  - Simple more robust alternatives: Absolute error, focal loss, symmetric cross entropy.

$$L_{CE} = \sum_k y_k \log p_w(y_k|x) \Rightarrow$$

$$L_{MAE} = \|y - p_w(y_k|x)\|_1$$

$$L_{SCE} = L_{CE}\big(y, p(y|x)\big) + L_{CE}(p(y|x), y)$$

[Wang, CVPR'19, Symmetric cross entropy for robust learning with noisy labels;
Ghosh, AAAI'17, Robust Loss Functions under Label Noise for Deep Neural Networks]

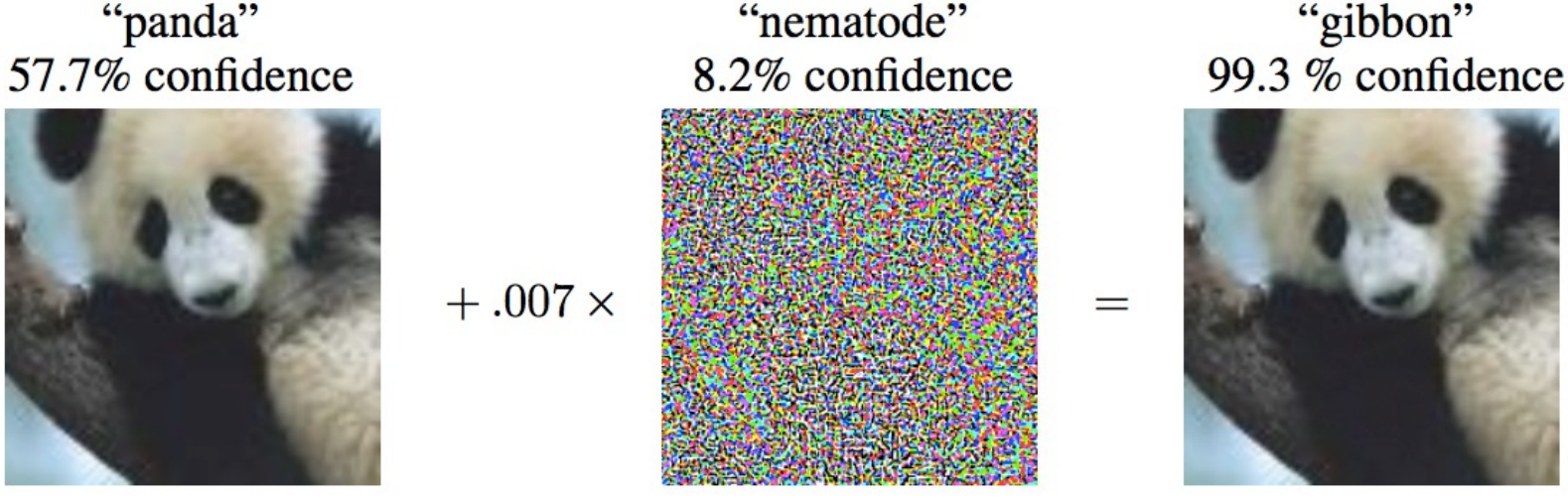
Outline

- Part I: Data-Efficient Deep Learning
  - Common Regularizers
  - Beyond Regularization
- Part II: Intro to Robust and Explainable Deep Learning
  - Adversarially Robust Learning
  - Uncertainty in Deep Learning
- Part III: Intro to Meta-learning
  - Concept
  - Some examples

Adversarial Attacks and Defense

Adversarial Examples

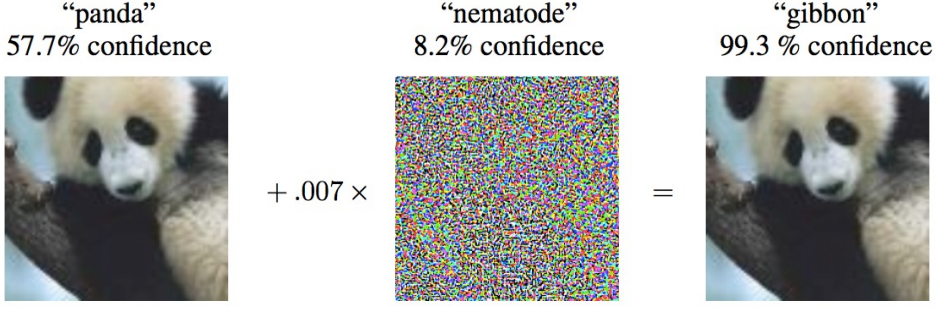
- "Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake" – Goodfellow, 2017
  - Interesting: In many cases one can perturb an input in a way that fools an AI system but would not fool (or indeed by invisible to) a human!

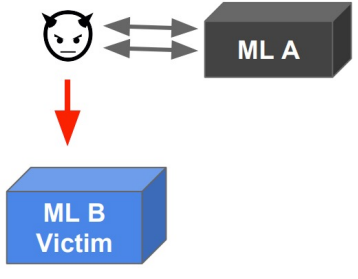


[Szegedy, ICLR'14, Intriguing properties of neural networks]

Finding Adversarial Examples:

- Given a loss function $L(x, y, \theta)$ of image, label, parameters.
- Contrast:
  - Conventional Training: $\theta' = \theta - \alpha \nabla_\theta L(x, y, \theta)$
  - Adversarial Example Search: $x' = x + \alpha \nabla_x L(x, y, \theta)$
    - Imperceptible: Add constraint: $\delta = |x' - x| < \epsilon$
  - Untargeted: Ask for any mistake. Targeted: Ask for a specific mistake.
- Here: `White-box' assumes attacker has access to your network θ
  - If you are attacking a web API, you probably don't have access to $\nabla_x L(x, y, \theta)$
  - But `Black-box' transfer attacks also work….



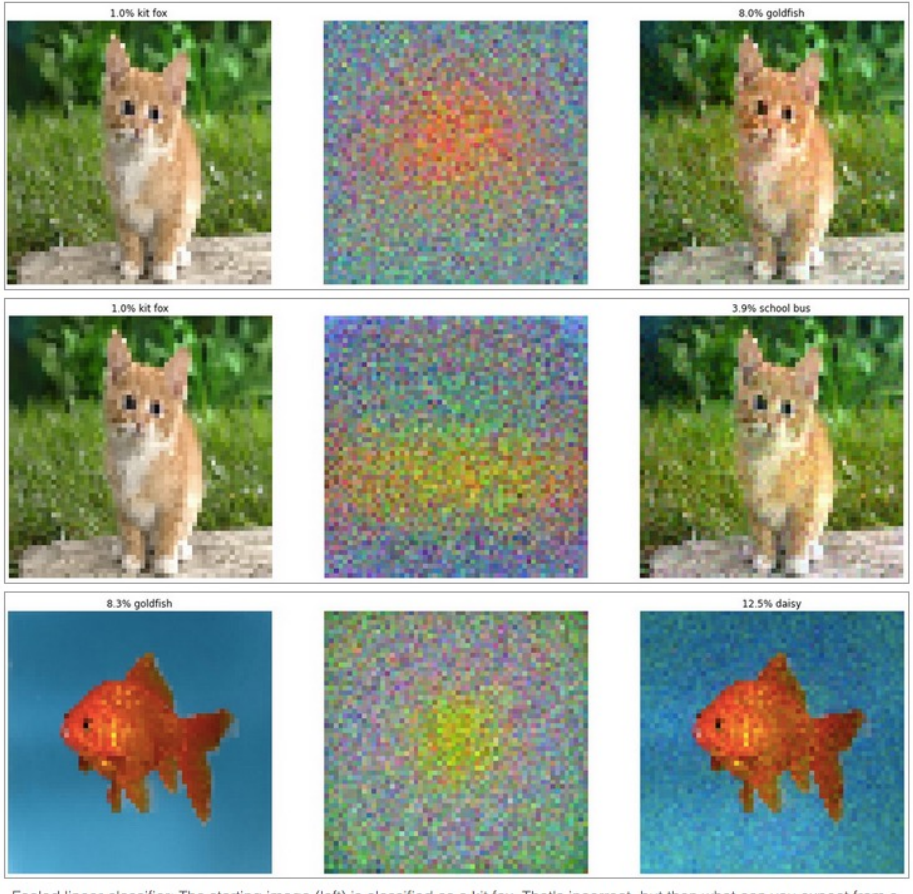


Maximizing p(airplane|x)

[Goodfellow, 2017]

Not only a deep learning phenomenon

Linear classifiers and many others can also be fooled.



Fooled linear classifier: The starting image (left) is classified as a kit fox. That's incorrect, but then what can you expect from a linear classifier? However, if we add a small amount "goldfish" weights to the image (top row, middle), suddenly the classifier is convinced that it's looking at one with high confidence. We can distort it with the school bus template instead if we wanted to.

http://karpathy.github.io/2015/03/30/breaking-convnets/

Adversarial (Question) Examples for VQA

Visual Question Answering

Strongest "Yes" Beliefs

[Liu, CVPR'18; Liu TPAMI'18 - iVQA: Inverse Visual Question Answering]

Defending Against Adversarial Examples

- A Standard Solution: Adversarial Training

$$\underset{\theta}{\text{minimize}} \frac{1}{|S|} \sum_{x,y \in S} \max_{\|\delta\| \leq \epsilon} \ell(h_\theta(x+\delta), y).$$

Repeat:

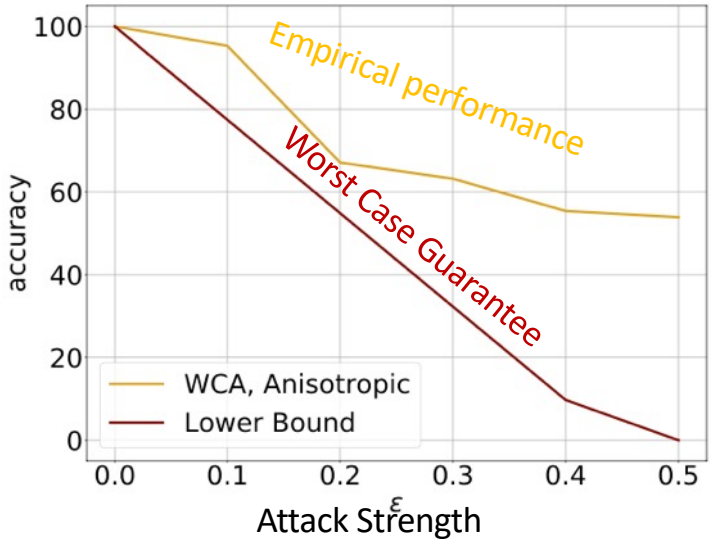
1. Select minibatch B, initialize gradient vector $g := 0$
2. For each (x, y) in B:
   a. Find an attack perturbation $\delta^\star$ by (approximately) optimizing
   $$\delta^\star = \underset{\|\delta\| \leq \epsilon}{\text{argmax}}\, \ell(h_\theta(x+\delta), y)$$
   b. Add gradient at $\delta^\star$
   $$g := g + \nabla_\theta \ell(h_\theta(x+\delta^\star), y)$$
3. Update parameters θ
   $$\theta := \theta - \frac{\alpha}{|B|} g$$

Reasonably effective
But slow! => Attack model at each training Iteration.

[Madry et al, ICLR'18, Towards Deep Learning Models Resistant to Adversarial Attacks]

State of the Art: UDRC's WCA-Net

Encoder $p(z|x)$ → $z \sim p(z|x)$ → Classifier $p(y|z)$ → Monkey



Key Contribution:

1. A bound for adversarial robustness of stochastic NNs:
2. A algorithm based on this bound => SotA performance.

Model:

$$h(\vec{x}) = \vec{w}^T \vec{z}, + b,$$
$$\vec{z} = f(\vec{x}) + \vec{\epsilon}, \qquad \vec{\epsilon} \sim \mathcal{N}(0, \Sigma)$$

$$p(z|x) \sim \mathcal{N}(f(\vec{x}), \Sigma)$$

Bound:

Adversarial vs Clean Gap

Is Less Than…

$$G^h_{p,\epsilon}(\vec{x}, y) = \max_{\vec{\delta}:\|\vec{\delta}\|_p \leq \epsilon} P(m_h(\vec{x}+\delta, y) \leq 0) - P(m_h(\vec{x}, y) \leq 0) \leq \frac{\Delta^{\tilde{h}}_p(\vec{x}, \epsilon)}{\sqrt{2\pi \vec{w}^T \Sigma \vec{w}}},$$

"Weight Covariance Alignment"

$$\mathcal{L}_{\text{WCA}} = \sum_{i=1}^{C} \ln(\vec{w}_i^T \Sigma \vec{w}_i),$$

⇒ Aligning decoder weights to noise minimises the bound.

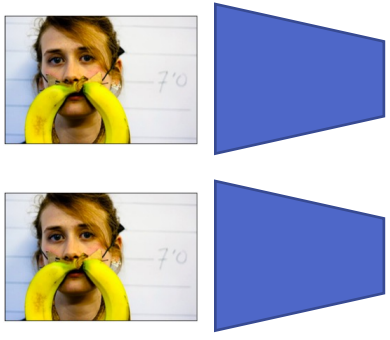
[Eustratiadis, ICML'21, Weight Covariance Alignment] UDRC invention!

Explainable Deep Learning

Uncertainty in Deep Learning

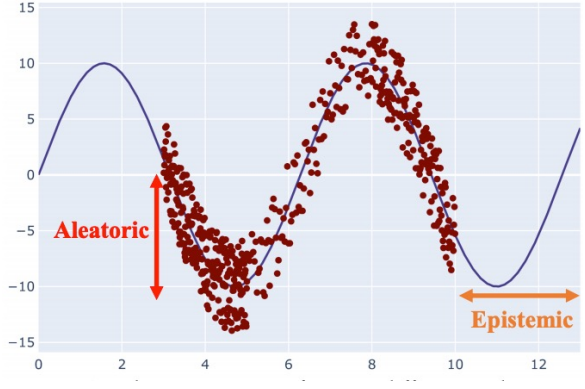
Uncertainty in Deep Learning

- Decision Functions

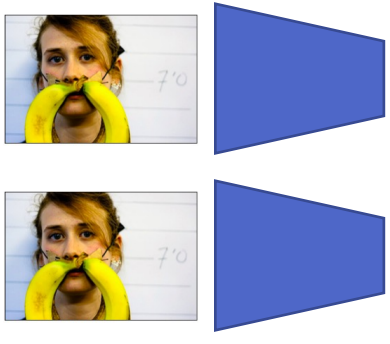


Banana

30 yrs

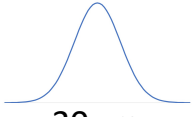


- Uncertainty Estimates



(Banana, 40%), (Moustache, 30%), (Person, 30%)

Start by using a softmax output.

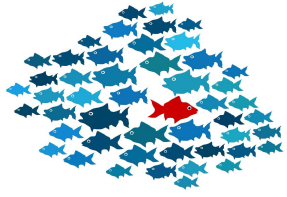


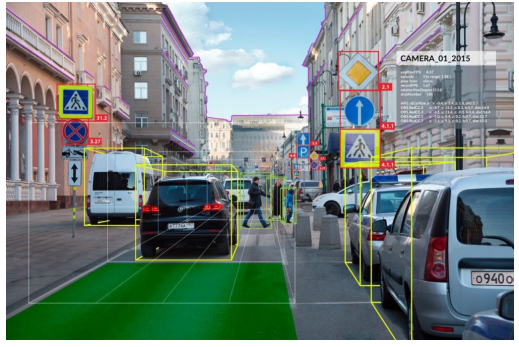
30 yrs

Output a mean and variance

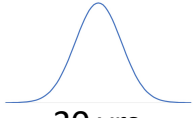
Why Is Uncertainty Important?

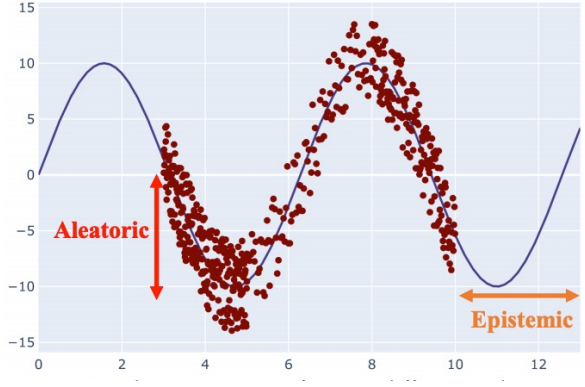
- Abstaining / Pass Decision to Human
- Outlier Detection
- Information Fusion
- Decision Theory
- Active Learning (Data Collection, Model Improvement)





(Banana, 40%), (Moustache, 30%), (Person, 30%)

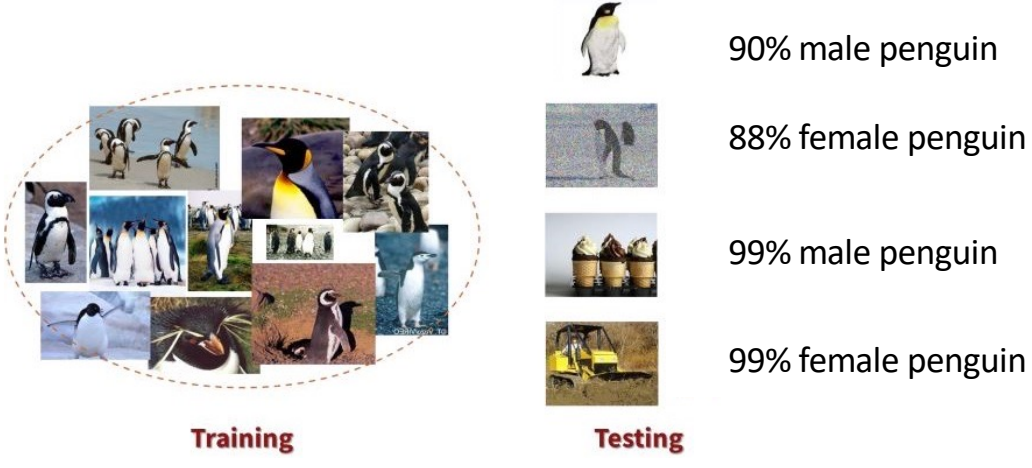




Typical Uncertainty Failure Cases

- Overconfidence
  - EG: Model gets 95% accuracy, but also has 100% confidence on the 5% it gets wrong.

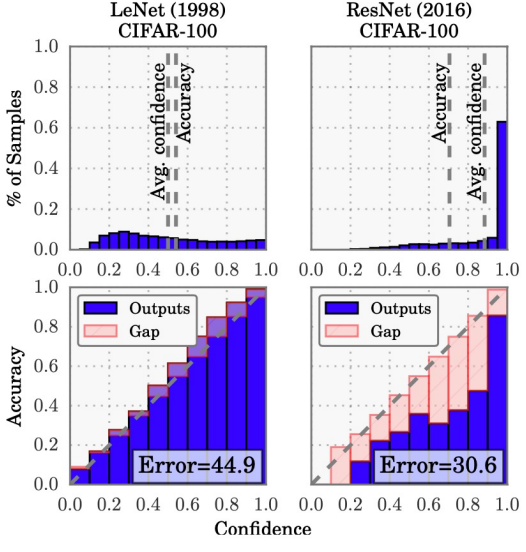
- Out-of-distribution:



How to Measure Uncertainty Calibration?

- For all inputs to which your model assigns x% confidence, does it get x% of them correct?
- ECE:
  - Bin by confidence, then compare confidence and accuracy bin-wise.

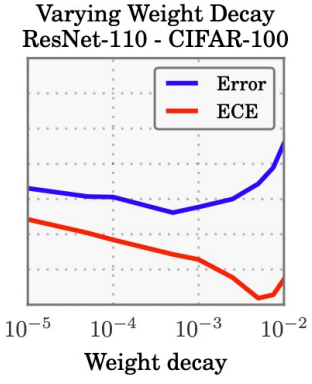
$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \Big| \mathrm{acc}(B_m) - \mathrm{conf}(B_m) \Big|,$$



[Guo, ICML'17, On Calibration of Modern Neural Networks]

How to Improve Uncertainty Calibration

- Regularization:
  - L2 regularization, weight-decay.
  - Label-smoothing, network size.
- Change of Loss Function
  - Cross-entropy => Focal Loss, Brier



$$L_{CE} = -\log p_w(x) \qquad L_{CE} = -\left(1 - p_w(x)\right)^{\gamma} \log p_w(x) \qquad L_{MSE} = \|y - p_w(x)\|_2^2$$

- Ensembles:
  - Train a whole set of models: $\{p_{w_k}(y|x)\}$
  - Average their prediction: $p(y|x) = \left(\frac{1}{K}\right) \sum_k p_{w_k}(y|x)$

[Lakshminarayanan, NIPS'17, Simple and Scalable Uncertainty Estimation; Mukhoti, NIPS'20 Calibrating Deep Networks using focal Loss]

How to Improve Uncertainty Calibration

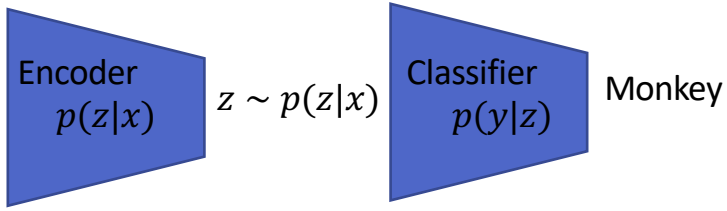
- Industry Standard: Temperature Scaling:
  - After training, recalibrate on validation set.
    - Note: Doesn't change any decisions.
    - Note: Usually complementary to other methods
    - Limitation: Requires a representative validation set.

$$p_w(y_k|x) \propto \frac{\exp(f_w(y_k|x)/T)}{\sum_l \exp(f_w(y_l|x)/T)}$$

$$T = argmin\ ECE(D_{val}, f_w)$$

How to Improve Uncertainty Calibration: Stochastic Neural Nets

- Stochastic Neural Networks:



$$h(\vec{x}) = \vec{w}^T \vec{z}, +b,$$
$$\vec{z} = f(\vec{x}) + \vec{\epsilon}, \qquad \vec{\epsilon} \sim \mathcal{N}(0, \Sigma)$$

$$p(z|x) \sim \mathcal{N}(f(\vec{x}), \Sigma)$$

Training

$$\text{argmin}_{f,w,\Sigma} L(h_w(f(x)), y)$$

- How to avoid noise collapse? Train with an entropy-boosting regularizer:
  - $\Omega(\Sigma) = \mathrm{H}(p(z|x))$

$$\text{argmin}_{f,w,\Sigma} L(h_w(f(x)), y) - \Omega(\Sigma)$$

[Yu et al, AAAI'21, Simple and Effective Neural Networks]

How to Improve Uncertainty Calibration: Bayesian Neural Nets

- Bayesian Neural Networks:

Training neural network: $g_\theta(\cdot)$ with parameter θ. Given dataset $D = \{x, y\}$

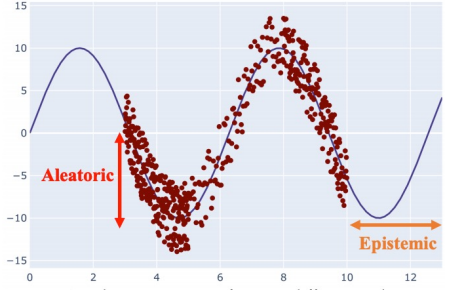
Training

Learning by Bayesian Inference

Training Set

Prior over parameters

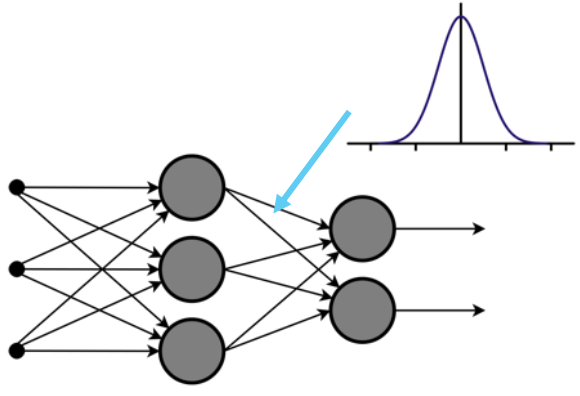
$$p(\theta|D, \phi) \propto p(D|\theta)p(\theta|\phi)$$



Testing

Recognize test set by integrating out parameters.

$$p(y|x, \phi) = \int p(y|x, \theta)p(\theta|D, \phi)d\theta$$

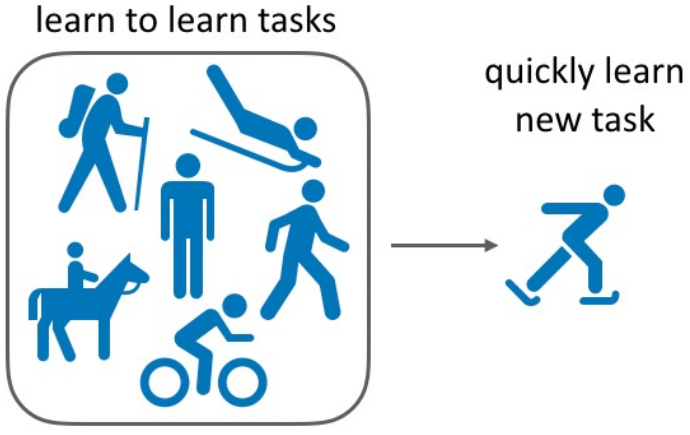


[Zhang et al, arXiv, Shallow Bayesian Meta-Learning]
AAAI'21, Few-Shot Learning Competition: 3rd Prize Winner

Outline

- Part I: Data-Efficient Deep Learning
  - Common Regularizers
  - Beyond Regularization
- Part II: Intro to Robust and Explainable Deep Learning
  - Adversarially Robust Learning
  - Uncertainty in Deep Learning
- Part III: Intro to Meta-learning
  - Concept
  - Some examples

Meta Learning and Learning-to-Learn



	Past: Shallow Learning	Current: Deep Learning	Future: Deep Meta Learning
Classifier	Learned	Learned	Learned
Feature	Hand-Crafted	Learned	Learned
Learning Algorithm Incl. Architecture, Hyperparams, etc	Hand-crafted	Hand-crafted	Learned

[Graphic from Yu et al, CORL'19]

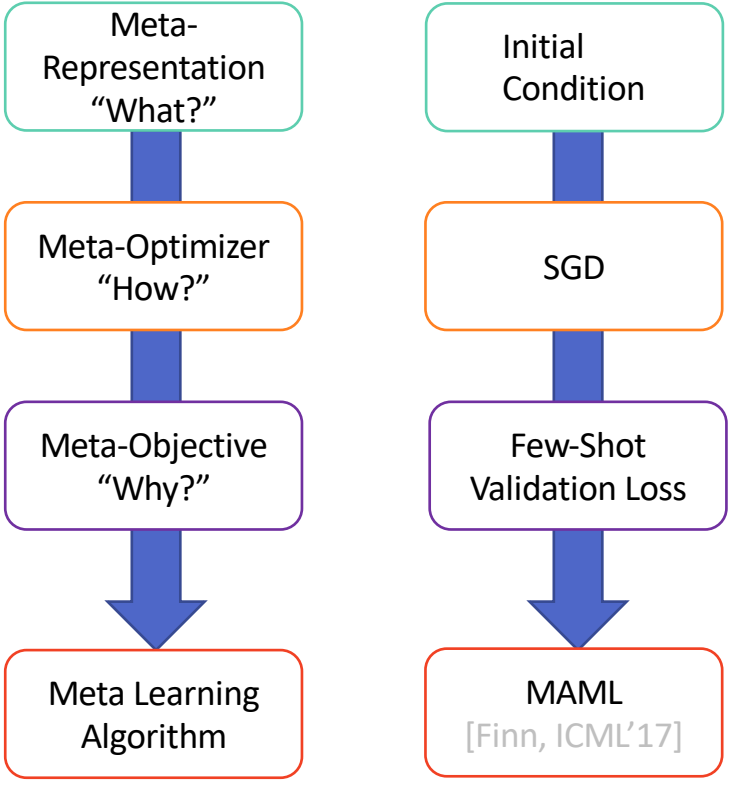
Defining Learning-to-Learn

- Machine Learning Definition [Mitchell, 1993]
  - Given: Task *T*, experience *E~T*, performance measure *P*.
  - A program learns if performance at *T* wrt *P* improves with amount of experience *E*.
- Learning to Learn Definition [Thrun, 1998]
  - Given: Tasks *T* from a task distribution *T~D*, experience of each task *E~T*, performance measure *P*.
  - A program learns-to-learn if performance at tasks *T* wrt *P* improves with amount of experience *E* and with number of tasks *T*.

➢ Propose modification: Task *episodes* (single task meta-learning).

Meta-Learning: Building Blocks

- Formalizing meta-learning



Meta-Learning: Minimise loss over a task distribution wrt meta-representation ω.

$$\min_{\omega} \mathbb{E}_{D\sim p(D)} L(D;\omega)$$

Meta-Training, Bi-level optimization view:
Outer: Train the algorithm ω
Inner: Train the model θ conditional on algorithm

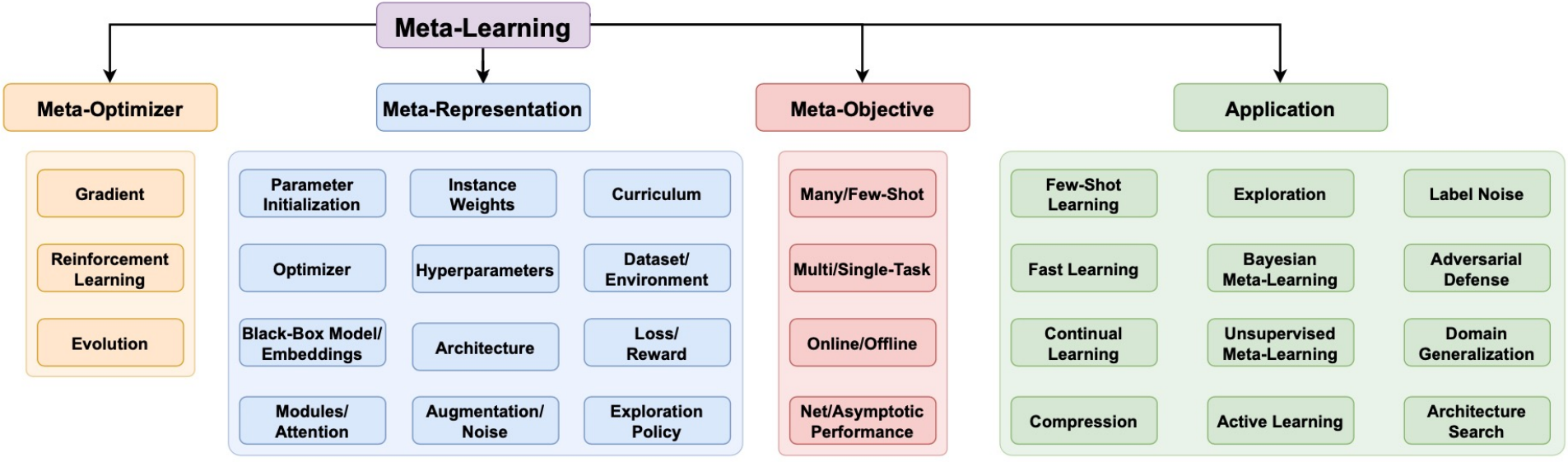
$$\omega^* = \arg\min_{\omega} \sum_t L\left(D_t^{val}; \theta_t^*, \omega\right)$$

$$\text{s.t. } \theta_t^* = \arg\min_{\theta} L(D_{t,}^{trn}; \theta_t, \omega)$$

Meta-Testing: Deploy on a new task

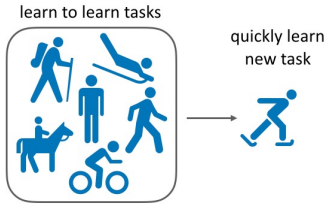
$$\theta^* = \arg\min_{\theta} L(D_{novel}; \theta, \omega)$$

Meta-Learning Opportunities

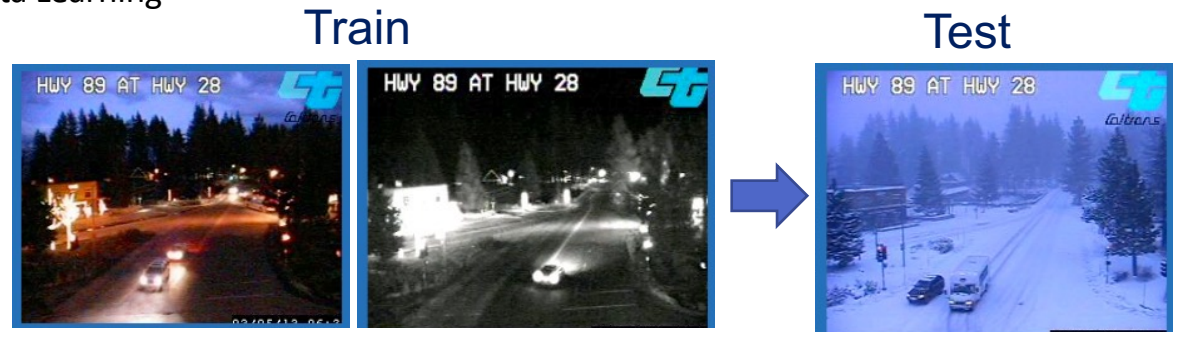


Meta Domain Generalisation

Multi-Task
Few-Shot Meta Learning



Multi-Domain
Meta Learning

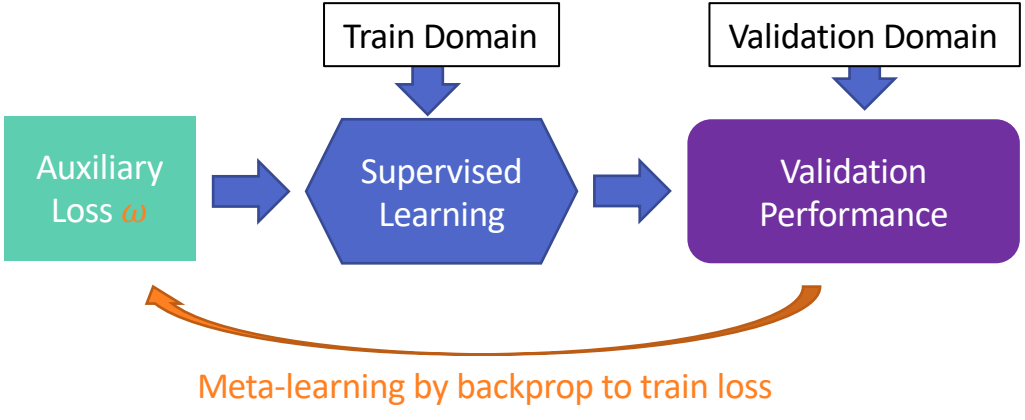


Auxiliary Loss Function
(Sub-symbolic)

SGD

Domain-Shifted Validation Loss
(Single task, Online)

Feature Critic
[Li, ICML'19]

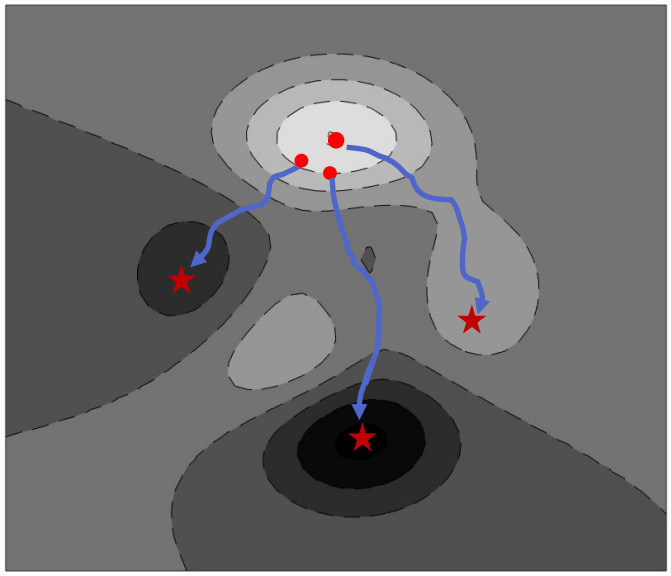


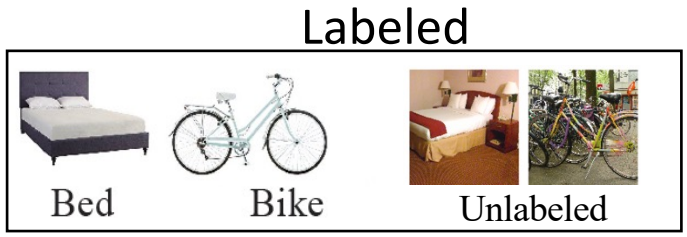
[Li et al, Feature Critic Networks, ICML'19]

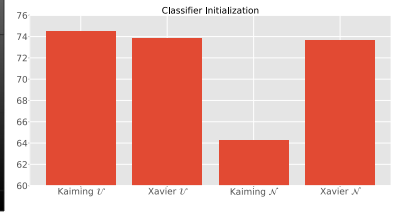
Meta Domain Adaptation

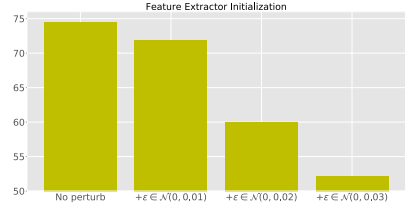
- Many popular DA algorithms are initialization dependent.
  - => Can we meta-learn a good initialization?

DA Algorithm Loss Surface









MSDA [ICCV-19]
MCD [CVPR-18]
DANN [JMLR-16]
MME [ICCV-19]
CCSA [ICCV-17]
JiGen [CVPR-19]

DA Learning ● Initialization ★ Solution

[Li & Hospedales, Meta Learning for Domain Adaptation, ECCV-20]

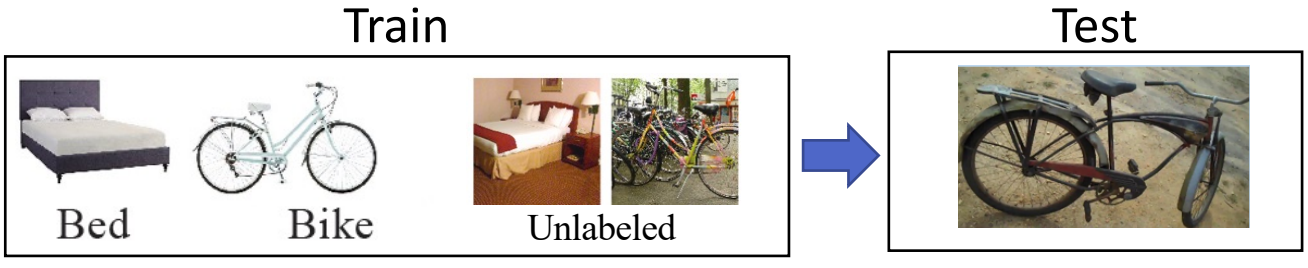
Meta Domain Adaptation

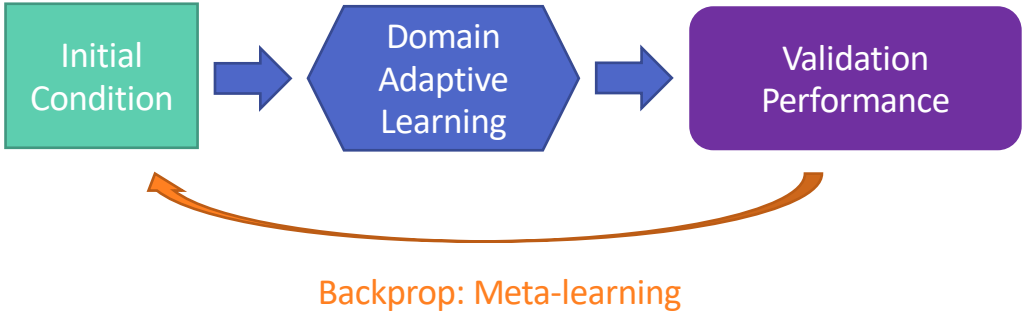
Initialization

SGD

Loss After Domain Adaptation (Single task, Online)

MetaDA [Li, ECCV'20]





CNN	Dataset	Base Method	MetaDA Benefit
ResNet-18	PACS	MCD	+2.5%
ResNet-18	PACS	DANN	+2.0%
ResNet-18	PACS	JiGen	+3.4%

[Li et al, Meta-Learning for Domain Adaptation, ECCV'20]

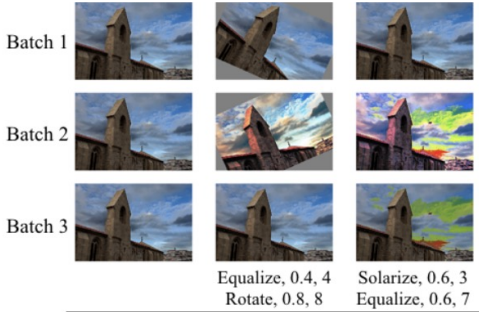
Meta Learning Data Augmentation

Data Augmentation Policy

SGD

Validation Loss (Single task, Online)

DADA [Li, ECCV'20]



Augmentation Policy

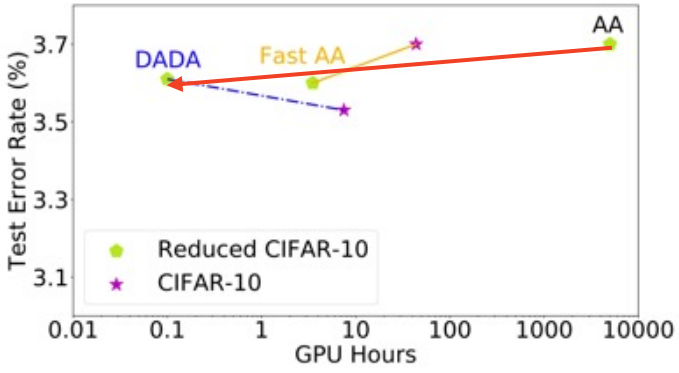
Supervised Learning

Validation Performance

Backprop: Meta-learning

~~Offline~~ => Online

5 orders of magnitude!



[Li et al, Differentiable Automatic Data Augmentation, ECCV'20]

Meta Learning Label Noise Robustness

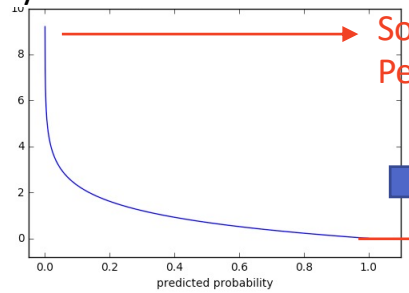
Robust Loss
(Symbolic Taylor Polynomial)

Evolutionary Search

Validation Loss
(Multitask, Offline)

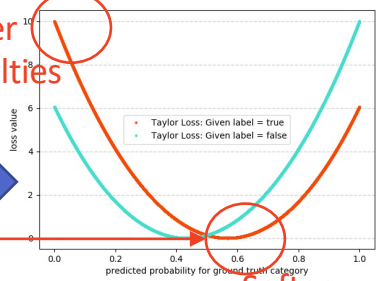
Searching for Robustness
[Gao, arXiv'20]

Symbolic Loss: Cross Ent



Softer Penalties

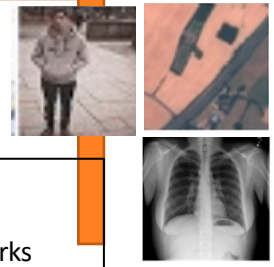
Learned



Softer Minima

Noisy Label Dataset

Multi-task Offline ES Meta-Learning



Symbolic Loss:
=> Meta-Learn once.
Then Transferable to Novel Benchmarks

Supervised Learning

Validation Performance

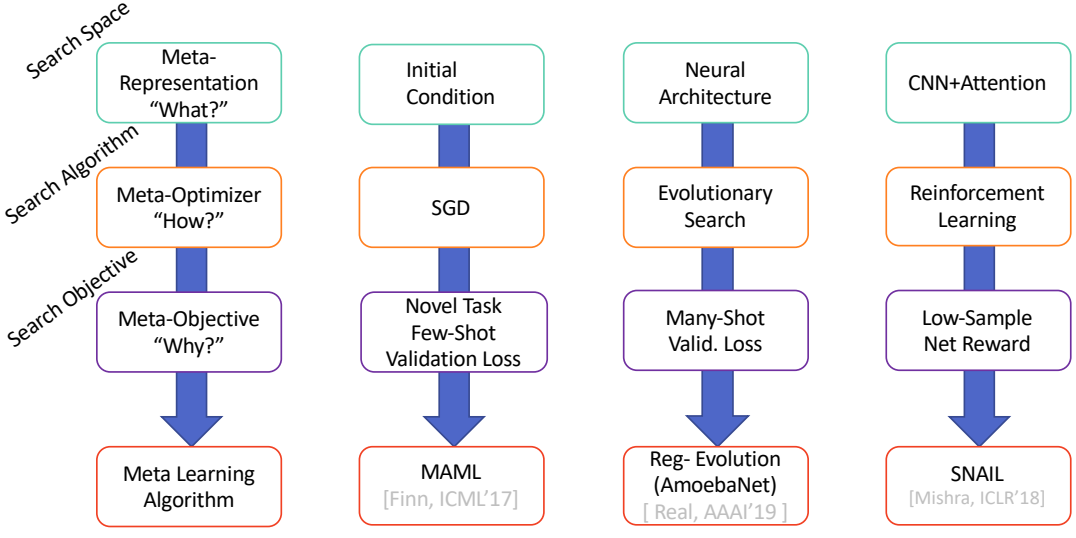
Conventional Online Transductive Meta-Learning

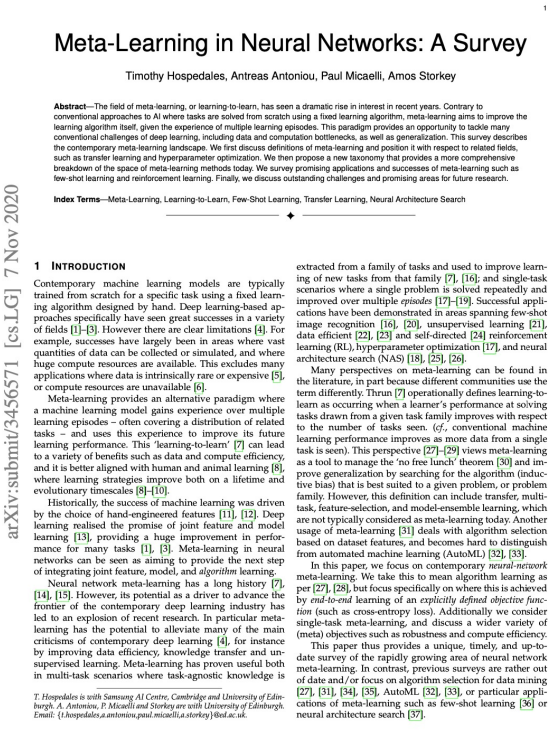
[Gao et al, Searching for Robustness, arXiv'20….]

Meta-Learning

- For more details:
  - Hospedales et al, Meta-Learning in Neural Networks: A Survey, IEEE Trans PAMI, 2021.

    - Components of a meta-learning algorithm:





arXiv:submit/3456571 [cs.LG] 7 Nov 2020

Meta-Learning in Neural Networks: A Survey

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, Amos Storkey

Abstract—The field of meta-learning, or learning-to-learn, has seen a dramatic rise in interest in recent years. Contrary to conventional approaches to AI where tasks are solved from scratch using a fixed learning algorithm, meta-learning aims to improve the learning algorithm itself, given the experience of multiple learning episodes. This paradigm provides an opportunity to tackle many conventional challenges of deep learning, including data and computation bottlenecks, as well as generalization. This survey describes the contemporary meta-learning landscape. We first discuss definitions of meta-learning and position it with respect to related fields, such as transfer learning and hyperparameter optimization. We then propose a new taxonomy that provides a more comprehensive breakdown of the space of meta-learning methods today. We survey promising applications and successes of meta-learning such as few-shot learning and reinforcement learning. Finally, we discuss outstanding challenges and promising areas for future research.

Index Terms—Meta-Learning, Learning-to-Learn, Few-Shot Learning, Transfer Learning, Neural Architecture Search

1 Introduction

Contemporary machine learning models are typically trained from scratch for a specific task using a fixed learning algorithm designed by hand. Deep learning-based approaches specifically have seen great successes in a variety of fields [1]–[3]. However there are clear limitations [4]. For example, successes have largely been in areas where vast quantities of data can be collected or simulated, and where huge compute resources are available. This excludes many applications where data is intrinsically rare or expensive [5], or compute resources are unavailable [6].

Meta-learning provides an alternative paradigm where a machine learning model gains experience over multiple learning episodes – often covering a distribution of related tasks – and uses this experience to improve its future learning performance. This 'learning-to-learn' [7] can lead to a variety of benefits such as data and compute efficiency, and it is better aligned with human and animal learning [8], where learning strategies improve both on a lifetime and evolutionary timescales [8]–[10].

Historically, the success of machine learning was driven by the choice of hand-engineered features [11], [12]. Deep learning realised the promise of joint feature and model learning [13], providing a huge improvement in performance for many tasks [1], [3]. Meta-learning in neural networks can be seen as aiming to provide the next step of integrating joint feature, model, and *algorithm* learning.

Neural network meta-learning has a long history [7], [14], [15]. However, its potential as a driver to advance the frontier of the contemporary deep learning industry has led to an explosion of recent research. In particular meta-learning has the potential to alleviate many of the main criticisms of contemporary deep learning [4], for instance by improving data efficiency, knowledge transfer and unsupervised learning. Meta-learning has proven useful both in multi-task scenarios where task-agnostic knowledge is extracted from a family of tasks and used to improve learning of new tasks from that family [7], [16]; and single-task scenarios where a single problem is solved repeatedly and improved over multiple *episodes* [17]–[19]. Successful applications have been demonstrated in areas spanning few-shot image recognition [16], [20], unsupervised learning [21], data efficient [22], [23] and self-directed [24] reinforcement learning (RL), hyperparameter optimization [17], and neural architecture search (NAS) [18], [25], [26].

Many perspectives on meta-learning can be found in the literature, in part because different communities use the term differently. Thrun [7] operationally defines learning-to-learn as occurring when a learner's performance at solving tasks drawn from a given task family improves with respect to the number of tasks seen. (*cf.*, conventional machine learning performance improves as more data from a single task is seen). This perspective [27]–[29] views meta-learning as a tool to manage the 'no free lunch' theorem [30] and improve generalization by searching for the algorithm (inductive bias) that is best suited to a given problem, or problem family. However, this definition can include transfer, multi-task, feature-selection, and model-ensemble learning, which are not typically considered as meta-learning today. Another usage of meta-learning [31] deals with algorithm selection based on dataset features, and becomes hard to distinguish from automated machine learning (AutoML) [32], [33].

In this paper, we focus on contemporary *neural-network* meta-learning. We take this to mean algorithm learning as per [27], [28], but focus specifically on where this is achieved by *end-to-end* learning of an *explicitly defined objective function* (such as cross-entropy loss). Additionally we consider single-task meta-learning, and discuss a wider variety of (meta) objectives such as robustness and compute efficiency.

This paper thus provides a unique, timely, and up-to-date survey of the rapidly growing area of neural network meta-learning. In contrast, previous surveys are rather out of date and/or focus on algorithm selection for data mining [27], [31], [34], [35], AutoML [32], [33], or particular applications of meta-learning such as few-shot learning [36] or neural architecture search [37].

T. Hospedales is with Samsung AI Centre, Cambridge and University of Edinburgh. A. Antoniou, P. Micaelli and Storkey are with University of Edinburgh. Email: {t.hospedales,a.antoniou,paul.micaelli,a.storkey}@ed.ac.uk.