Dall-E

Building an ML Application and Transfer Learning

Applied Machine Learning
Derek Hoiem

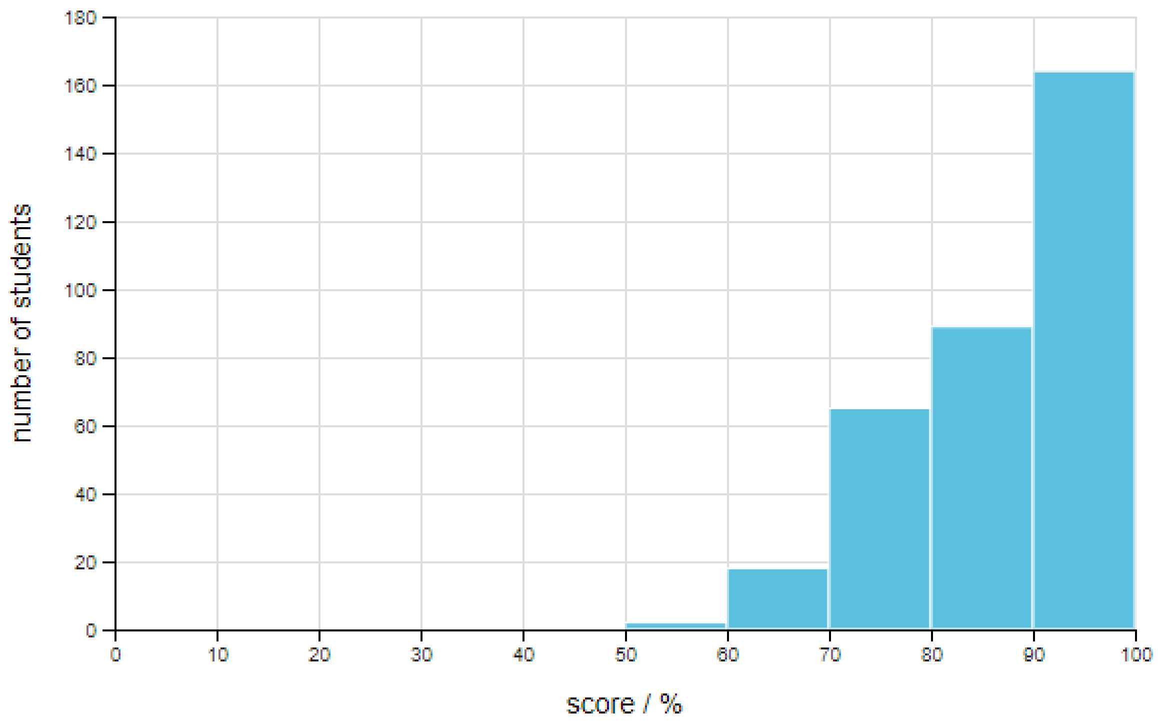
Today's lecture

- Review a few exam questions
- Example of building an ML application
- Transfer learning

Exam

- Well done!

Exam 1: Score statistics



Number of students	338
Mean score	87%
Standard deviation	10%
Median score	90%
Minimum score	56%
Maximum score	100%
Number of 0%	0 (0% of class)

Training test split - True or False

While the training error might slightly under-estimate the expected error of a random sample from the same distribution, the training error is still a useful way to decide if one model is better than another

- (a) False
- (b) True

Save & Grade | Save only | New variant

False: It's possible (and common) for a method to achieve low/zero training error, but still perform badly in testing, especially if the training examples are few compared to the model size

Training test split - Multiple Choice

Why is it important to evaluate with a test set of examples that are different from the examples used for training the model?

- (a) The expected error of the training set is lower than the expected error of a random sample from the same distribution
- (b) Expected errors of the train set and test set are both good indicators of expected performance for future examples, but the test set gives a more conservative estimate
- (c) The expected error of the training set is higher than the expected error of a random sample from the same distribution

(a): The parameters optimize the objective for the training data, so evaluation on the training data is a strongly biased optimistic estimate of performance, and is not a good indicator of expected performance for future examples

Emsembles - Multiple Choice

Which statement about random forests is **false**?

- (a) A subset of features is randomly selected to train each tree
- (b) Typically, each tree is grown to full depth, or with a high depth
- (c) The trees are trained on weighted samples, so that each tree focuses on errors of previously trained trees
- (d) The predictions of the trees are averaged to obtain the final prediction
- (e) Thus, random forests achieve low bias and low variance by averaging predictions of many complex trees

Emsembles - Multiple Choice

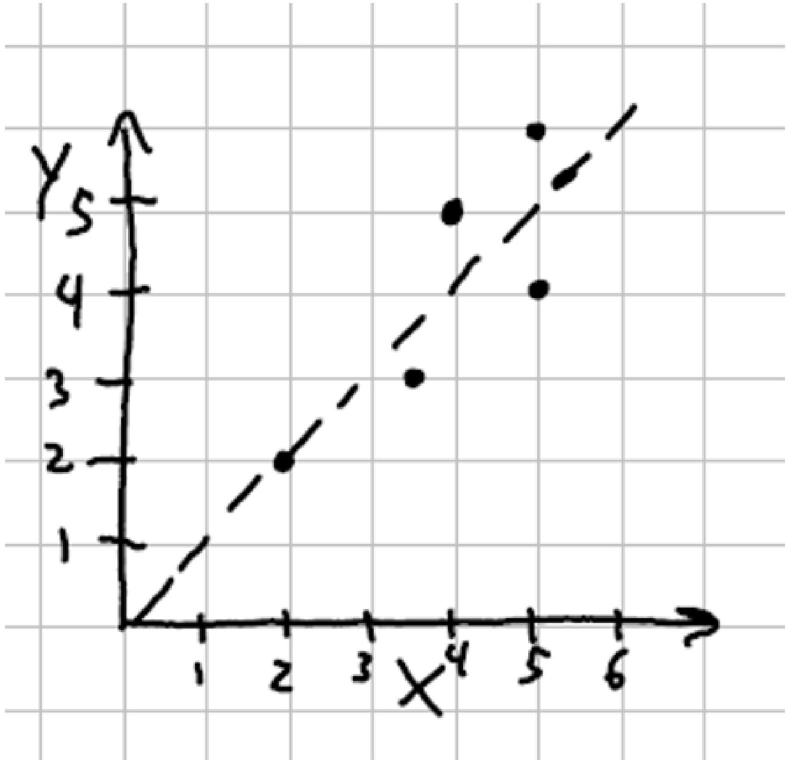
Which statement about boosted decision trees is **false**?

- (a) A subset of features is randomly selected to train each tree
- (b) Typically, each tree is grown to a short depth, sometimes as short as 2 leaf nodes
- (c) The trees are trained on weighted samples, so that each tree focuses on errors of previously trained trees
- (d) The predictions of the trees are combined to obtain the final prediction
- (e) Thus, boosted decision trees can achieve low bias and low variance by incrementally refining predictions using many simple trees

(c) The trees are independently trained

(a) All features are used to train each tree

KNN and Linear Regression - Multiple Choice



The plot above shows a linear regression from x to y based on five data points. For which of these values of x would the 1-nearest neighbor prediction be closest to the linear regression prediction? [choose one]

- (a) 1
- (b) 3
- (c) 4
- (d) 5
- (e) Cannot be determined from the plot

(b) x=3→y~=3 for regression and nearest neighbor

Stochastic gradient descent - True or False

True or False: Stochastic gradient descent operates by randomly sampling a weight update from a uniform distribution, taking a step, and evaluating whether the loss has decreased.

- (a) True
- (b) False

False: The weight update is not sampled randomly from a uniform distribution, but computed from a random sample of data. Also, SGD does not proceed by checking whether an update decreases the loss -- it just takes a step according to the loss gradient for that mini-batch.

Deep Learning - True or False

True or false: Eventually, machine learning researchers realized that sigmoid activation layers are *not sufficiently non-linear*, so they were replaced by ReLU activations.

- ○ (a) True
- ○ (b) False

False: Sigmoid activations are very non-linear. The problem is that the gradient is always less than 1 and often very small, so with many layers, the gradient becomes negligible.

We’ve covered a lot of ground in deep networks

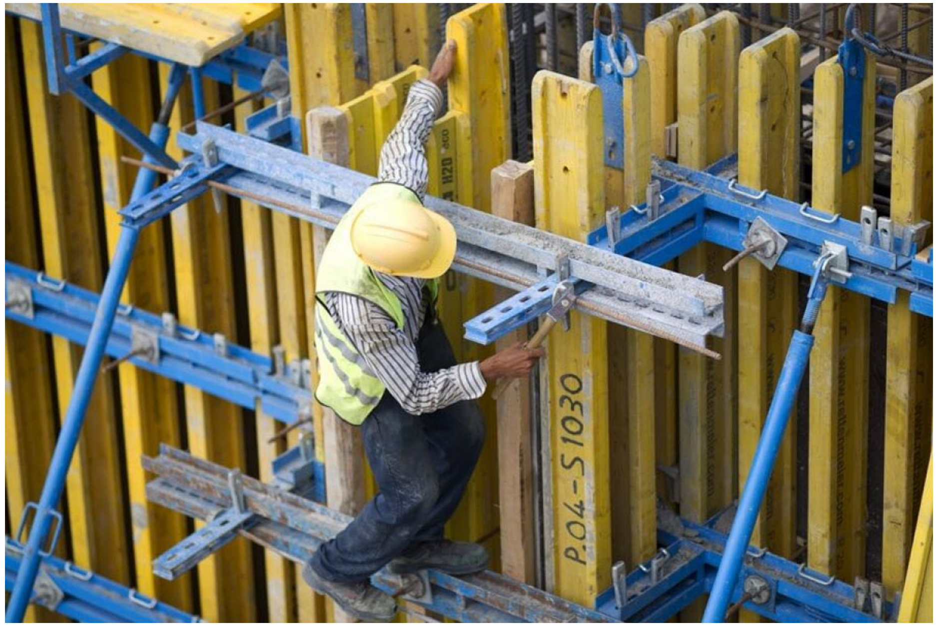
- ReLU activations, residual connections, and improved optimization techniques enabled training arbitrarily large and deep models 🙂

- Transformers provide a general and scalable way to process many kinds of data 😁

- Training on large annotated datasets or even larger unannotated datasets yields impressive models that are useful for many applications 🤯

How do you make your own ML application?

Example: Safety inspector wants to know what fraction of workers are wearing helmets, gloves, and boots on each job site



- PPE use is low (e.g., 60% use in a study in Egypt; frequent lack of use in US and other countries too)
- 1,008 fatal and 174,100 non-fatal injuries in US construction in 2020
- Consistently using PPE would significantly reduce injury and sometimes death

Image src

Step 1: Propose a solution in more technical terms

Proposed solution: Process images from the job site to detect the workers and count what fraction of detected workers are wearing each item

Left Glove: No
Right Glove: No
Hard hat: Yes
Vest: Yes
Boots: Yes

Step 1: Propose a solution in more technical terms

Main ML problem: Given an image, detect each worker and whether each detected worker is wearing: (a) glove on left hand; (b) glove on right hand; (c) boots; (d) hard hat; (e) vest

Note: There are lots of other aspects to the problem that we won't consider in this example

- How to get images onto a server where we can process them
- How to avoid duplicate counts when the same person is in more than one image on the same day
- How to summarize results and report them to the safety inspector

src
src

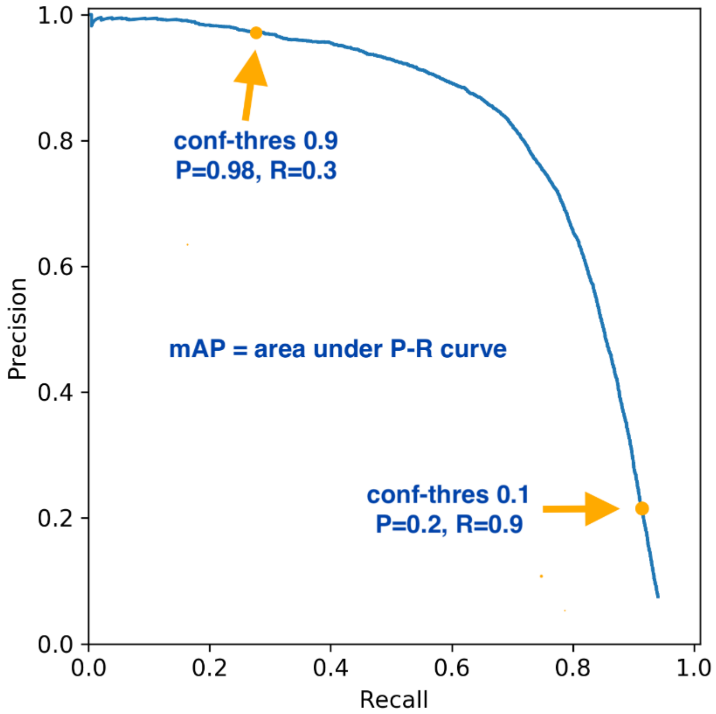
Step 2: Decide how to measure success

- What matters?
  - We want the overall estimate of fraction of workers wearing each item to be accurate
  - We want to report specific instances of workers not wearing an item, so that they can be checked as problematic or not

Step 2: Decide how to measure success

- Key aspects of performance
  - Human detection performance
    - Do we care about "small" or heavily occluded workers?
    - What counts as correct? (maybe high overlap in bounding boxes)
    - Measure precision (fraction of detections that are correct) and recall (fraction of workers that are detected)
    - Can measure Precision and Recall for each level of confidence and generate a P-R curve
    - Common overall performance measure is average precision
    - We may care about recall at a high precision value because we don't care about counting the number of workers, just knowing how likely a worker is to wear PPE

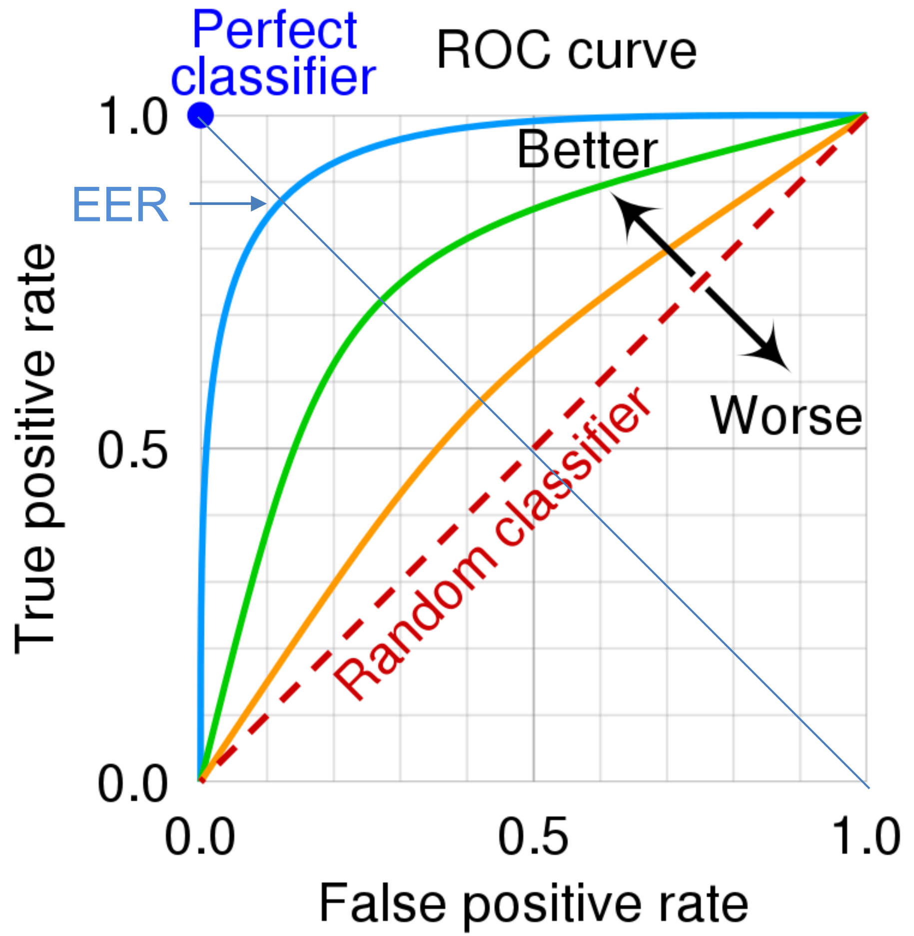
src



src

Step 2: Decide how to measure success

- Key aspects of performance
  - Human detection performance
  - Apparel classification performance, for correctly detected humans and each item:
    - TP rate: fraction of actual items that are detected
    - FP rate: fraction of item detections that are false
    - Summarize with equal error rate, accuracy when confidence is set so that FP rate = (1- detection rate)



Step 2: Decide how to measure success

- Key aspects of performance
  - Human detection performance
  - Apparel detection performance, for correctly detected humans and each item
  - Overall: Deviance between the estimated fraction of workers wearing equipment from the true fraction over a set of images
    - Difference in fractions
    - Bias: tends to overcount or undercount
    - Variance: how much could the difference be expected to vary, given a particular number of images

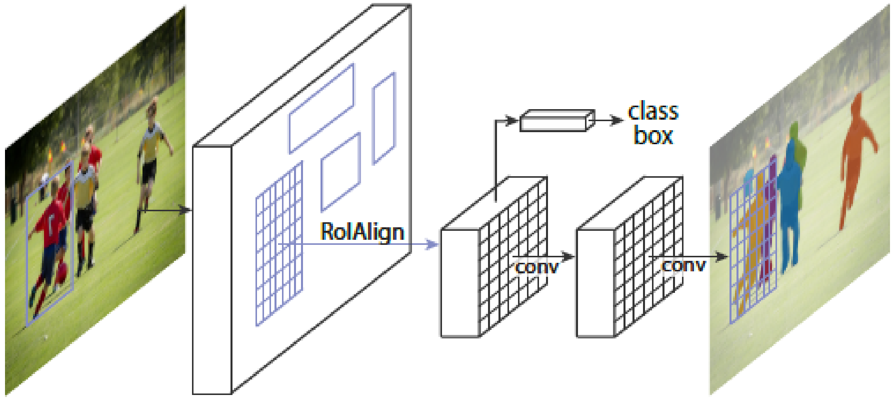
Step 3: collect and annotate validation/test images

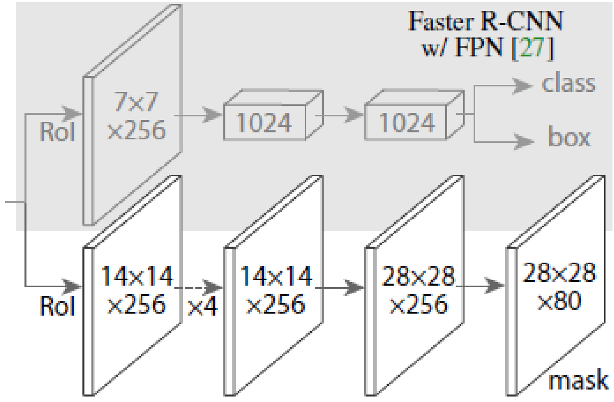
1. Collect images
   - Should be the same kind of images that will be processed in deployment
   - Collect from a variety of sites and different dates. Try to get representative diversity
2. Annotate
   - Draw boxes around each worker, even very small and hard to detect ones
   - For each PPE item, label “present”, “absent”, or “not visible”
   - How to get annotations
     - In house:
       - Use open source tool, such as VGG image annotator, or commercial tool like LabelBox
       - Develop custom tool (e.g. to process 360 images or fully integrate into existing application)
     - Outsource:
       - Amazon Mechanical Turk or other crowdsourcing tool
       - Commercial service
     - In this case, creating a small initial development validation set in-house and larger set by outsourcing could make sense
3. Split into a validation set and test set

Step 4: Determine technical details of approach

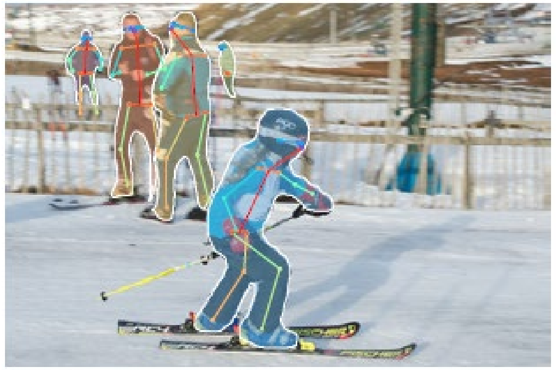
- For this example, we'll base the approach on Mask-RCNN

Detects objects and person keypoints





Includes additional branch to detect person keypoints



Modifications

- Remove bounding box detections and masks for non-person objects
- Add classification layer to keypoint branch to classify
  - Wearing left glove
  - Wearing right glove
  - Wearing hard hat
  - Wearing boots
  - Wearing safety vest

Step 5: Collect training data

- Consider combination of existing data (with applicable licenses) and new data
- Existing
  - Papers with code
  - Google for existing papers/datasets, e.g.
- Collect own data
  - similar to collecting test/validation, but not quite as much concern about being representative or reflecting actual use cases
  - E.g., could ask job sites to send photos of workers wearing and not wearing PPE (on purpose, briefly) while in natural poses

Step 6: Develop model

- Whenever possible, start with a pretrained model

- Alternatively, you could use unsupervised pretraining to initialize your model (e.g. Masked Autoencoder)

(from Chat GPT)

There are several places where you can find pre-trained Mask R-CNN models, depending on your specific needs. Here are a few options:

1. Matterport Mask R-CNN: Matterport provides a number of pre-trained models for various datasets, including COCO, Kitti, and Cityscapes. You can find the models on their GitHub page: https://github.com/matterport/Mask_RCNN.
2. Detectron2: Detectron2 is an open-source object detection library developed by Facebook AI Research. They provide pre-trained models for several datasets, including COCO, LVIS, and Cityscapes. You can find the models on their GitHub page: https://github.com/facebookresearch/detectron2.
3. TensorFlow Object Detection API: TensorFlow also provides pre-trained Mask R-CNN models for several datasets, including COCO and Kitti. You can find the models on their GitHub page: https://github.com/tensorflow/models/tree/master/research/object_detection.
4. Hugging Face Transformers: Hugging Face offers a collection of pre-trained models for various tasks, including object detection. You can find pre-trained Mask R-CNN models on their model hub: https://huggingface.co/models?pipeline_tag=object-detection&task_mask=1.

Note that these are just a few options, and there may be other sources of pre-trained Mask R-CNN models available online as

Regenerate response

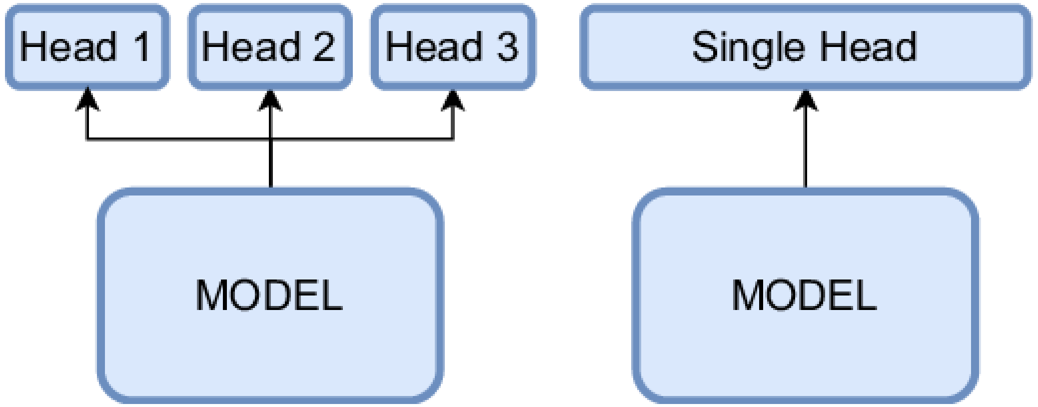
https://huggingface.co/models

Step 6a: Develop model: establish baselines

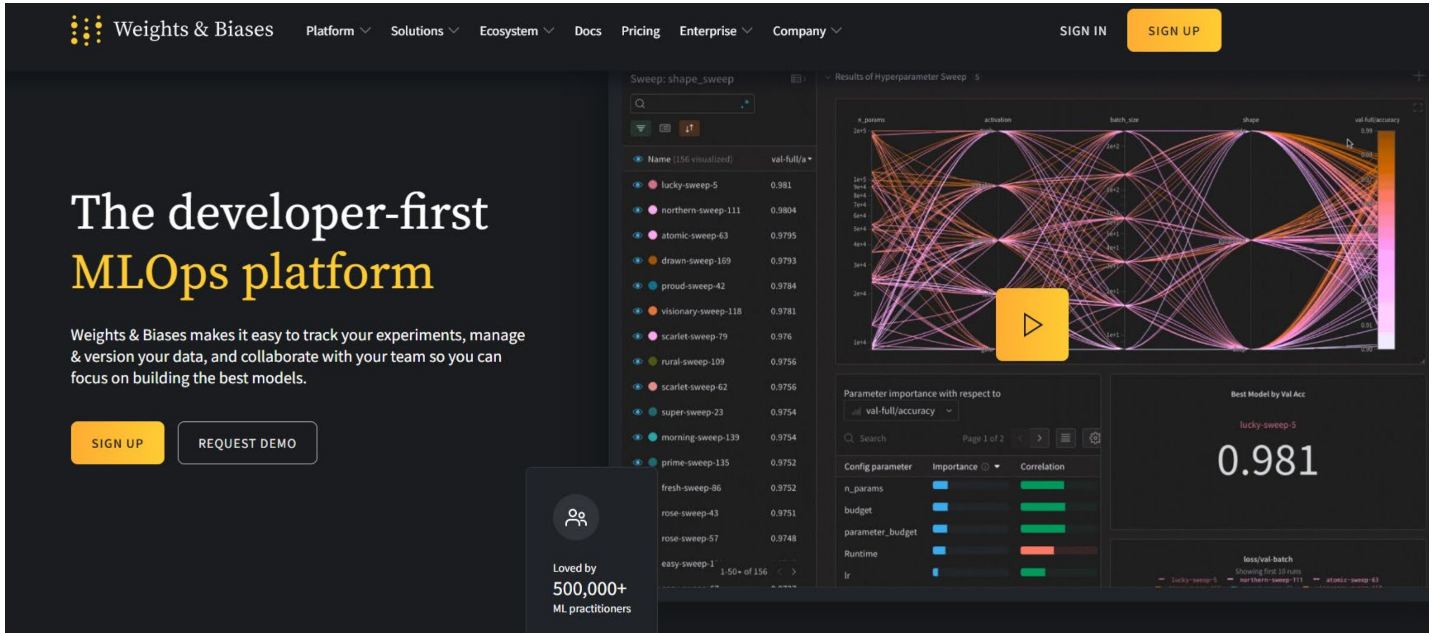
- Run the model as-is on your validation data and measure human detection performance
- Train a linear probe for classifying PPE item presence and measure all performance metrics
- Manually validate your evaluation code by displaying images and detections and checking against metrics

Step 6b: Develop model: refine model

- Fine-tune the model on your data
- Train using mix of existing and application-specific data
  - Apply only the losses that are applicable (e.g. detection or pose only for some datasets)
- Use tools like TensorBoard or Weights and Biases to monitor training and compare results
  - Always plot validation and training loss, and measure validation performance at training milestones



src



https://huggingface.co/autotrain

Step 7: Evaluate on test set

- Measure performance metrics and characterize when it works and doesn’t
  - As function of occlusion, person size, camera viewpoint, etc

Step 8: Integrate into application

- Beta test in complete workflows
- Write guides for when it works and doesn’t
- Improve efficiency, refine approach

Summary of how to build a new ML application

1. Identify problem and general approach to solution
   – This also involves thinking ahead to metrics, available models, data, and more, to ensure viability
2. Specify success metrics
   – Check with product managers and/or users to ensure these metrics reflect important performance characteristics
   – Often, the metrics can't be optimized directly
3. Create evaluation sets
   – Achieving targets for success metrics on these sets should indicate high likelihood of application success
4. Select model, objectives, and other design details
   – Usually this involves finding an analogous approach that has been successful
5. Collect data for training
   – Custom data and labeling is expensive and time-consuming, so exploit available data sources where available, and as allowed by license terms
6. Develop model, starting with baselines and simple approaches
   – Starting simple is critical so that it is easier to debug and validate changes
7. Evaluate on your test set
   – It's not just about the performance number, but about predictability and effectiveness within the application
8. Integrate into the application
   – This requires a lot of work and testing

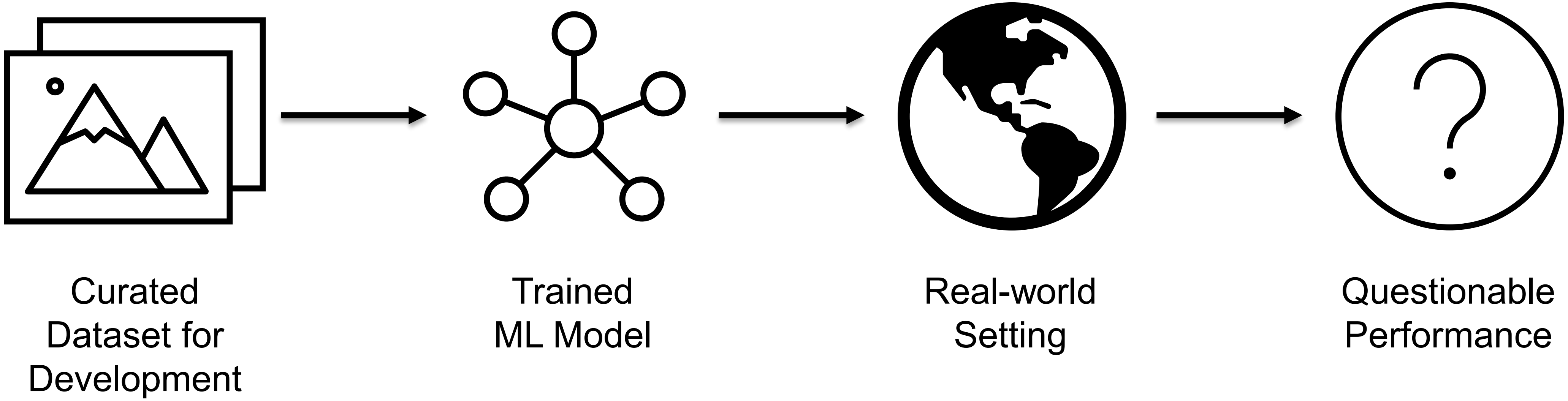
2 minute break

Thank you to Yuxiong Wang for following slides on domain adaptation and transfer learning!

Challenge for Machine Learning Models

- Development and real-world application may face different scenarios

- Limiting model performance and reliability



Slide credit: Yuxiong Wang

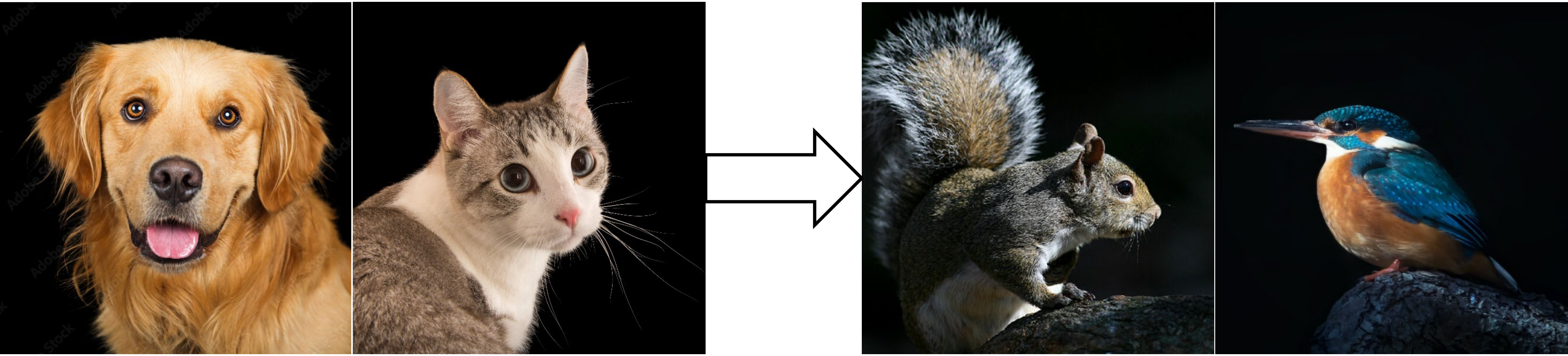
Types of Shifts

- Mainly two types of shifts from one scenario to another:

Task shift **Domain** shift

Slide credit: Yuxiong Wang

Task Shift: Changed Model Objectives



Classifying **dogs and cats**
Source (Old) Task

Classifying **squirrels and birds**
Target (New) Task

Slide credit: Yuxiong Wang

Domain Shift: Changed Input Data Distributions

Classifying dogs and cats **in studio**
Source (Old) Domain

Classifying dogs and cats **on grass**
Target (New) Domain

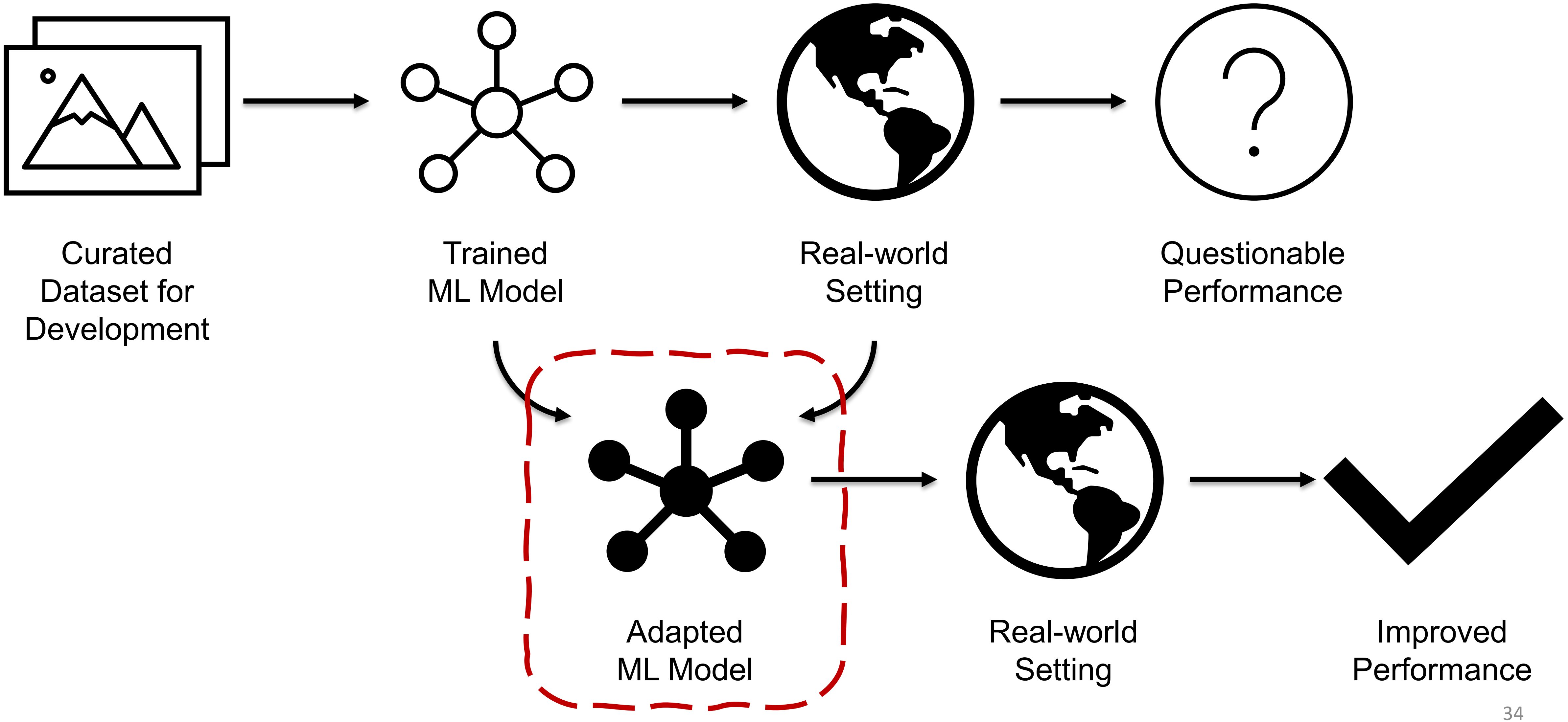
Slide credit: Yuxiong Wang

Types of Shifts: Task or Domain?

- Task shift
  - **Objective** of model is changed
  - But data distributions are usually assumed similar or related
- Domain shift
  - Input data come from changed **distributions**
  - But model task usually remains the same

Slide credit: Yuxiong Wang

Overcoming Task/Domain Shift



Slide credit: Yuxiong Wang

Overcoming Task/Domain Shift

- Task shift
  – Changed task objective
- Domain shift
  – Changed data distribution

- Task adaptation
  – Transfer learning
  – Meta-learning
- Domain adaptation
  – Instance translation
  – Domain adversarial training

- Some adaptation ideas may be applicable for both (e.g., Meta-learning)

Slide credit: Yuxiong Wang

Application: Autonomous Driving

- Adapt to different weather conditions, lighting conditions, or driving environments

Normal Weather Condition

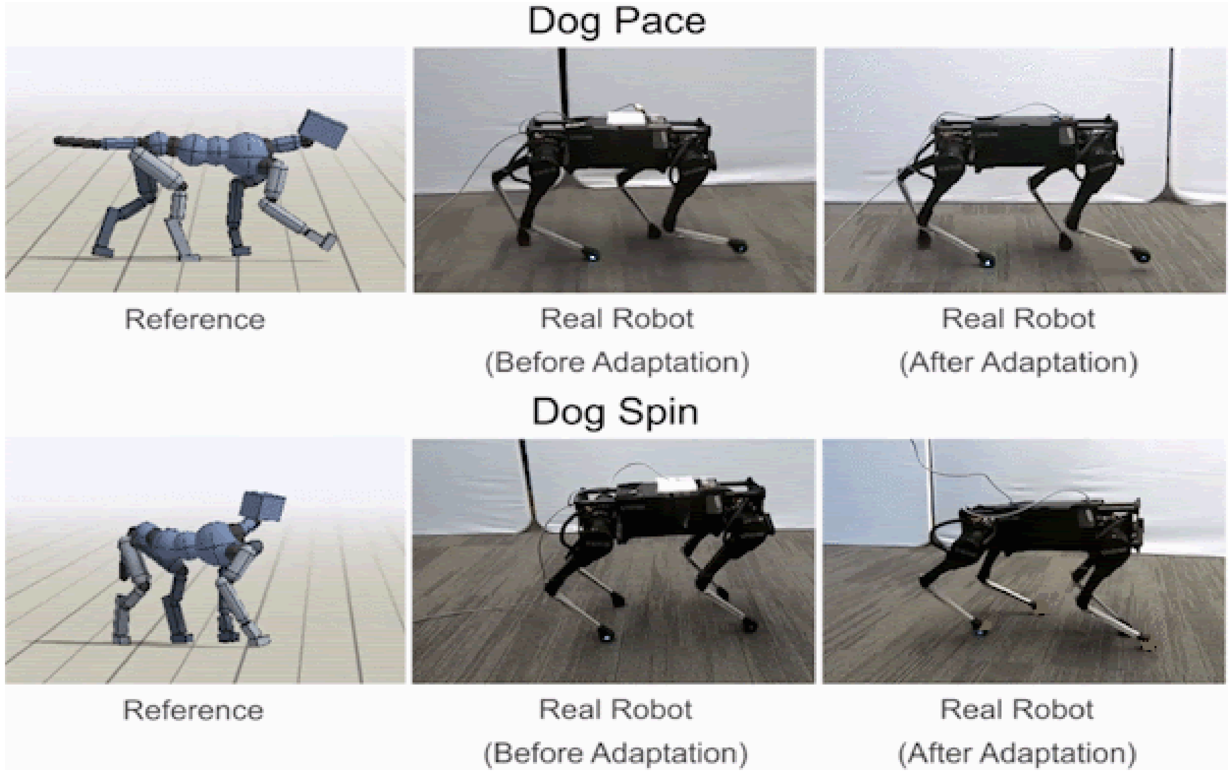
Foggy Weather Condition

Images from Sakaridis et al. IJCV '18

Slide credit: Yuxiong Wang

Application: Robotics

- Adapt from simulated environment to real-world robotic systems, or adapt from one learned task to another



Images from Google Research, 2020

Slide credit: Yuxiong Wang

Application: Speech recognition

- Adapt to different accents, speaking styles, or environmental conditions

- Example: Model trained with American English could be adapted to British English by fine-tuning on new domain

Methods for Task Adaptation

- Transfer learning: Pre-training and fine-tuning
- Meta-learning: Model-Agnostic Meta-Learning (MAML) and variants

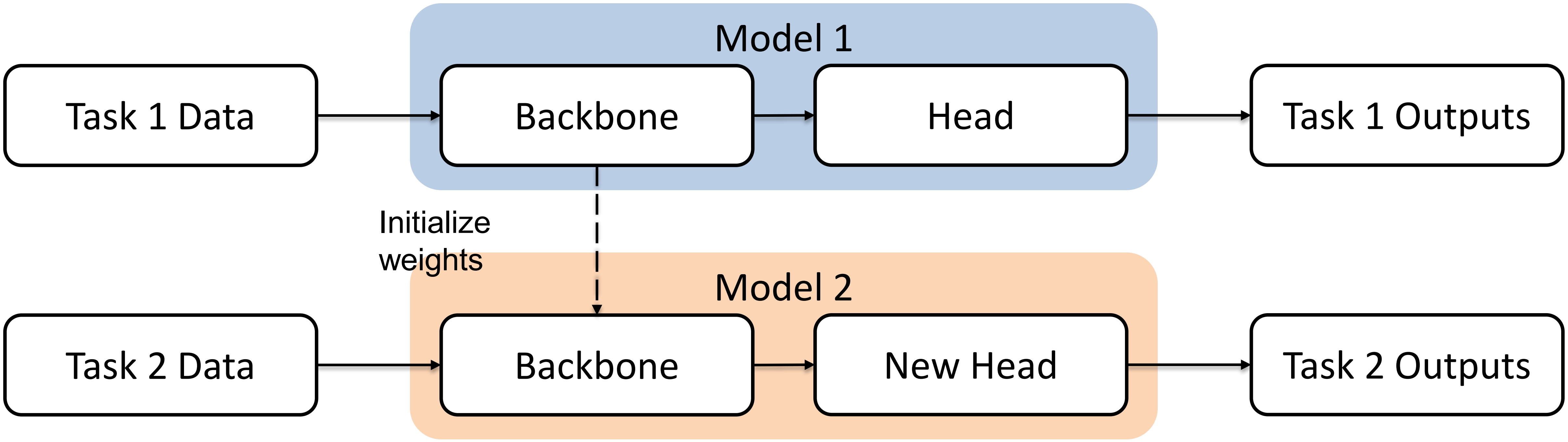
Transfer Learning

- Goal: To **reuse knowledge** learned from one task (which usually has abundant supervisory information), to another related task

- Implementation is simple
  - "Pre-train" model on source task
  - Copy learned weights from learned model
  - "Fine-tune" new model on target task

Slide credit: Yuxiong Wang

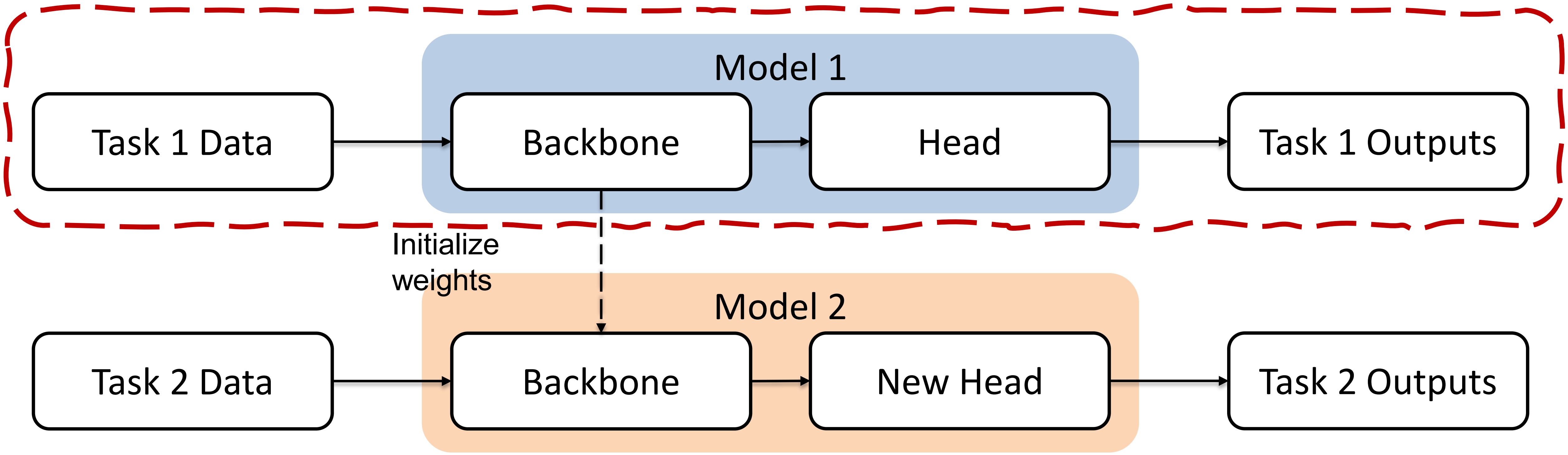
Transfer Learning



Slide credit: Yuxiong Wang

Transfer Learning

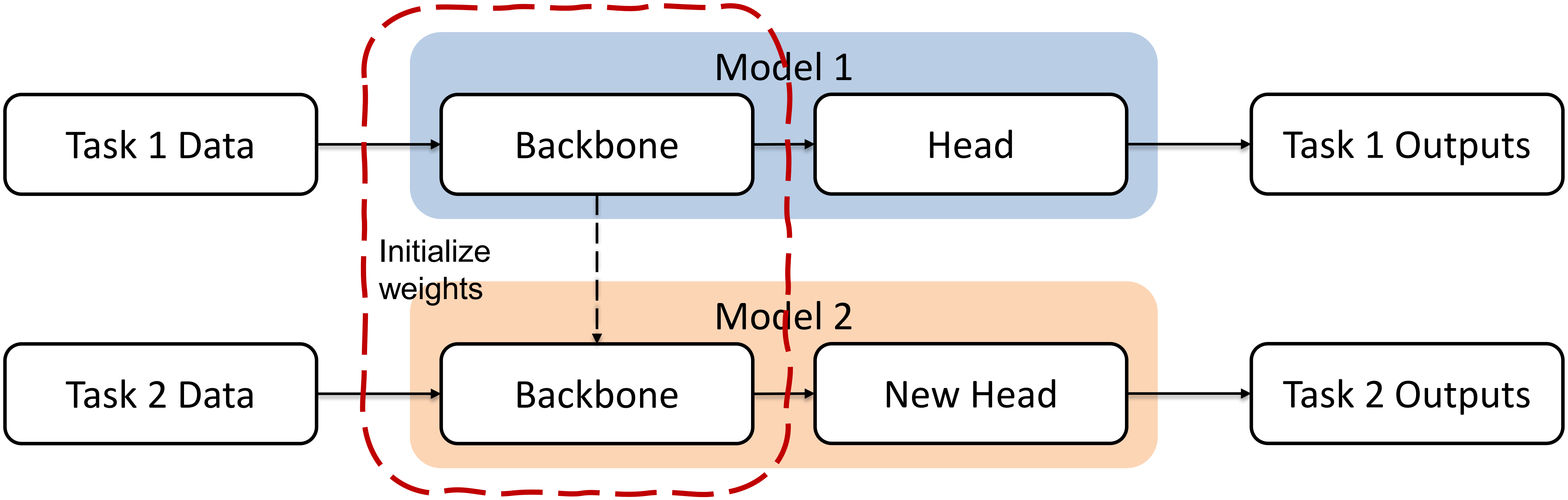
- Step 1: Pre-train Model 1 on Task 1



Slide credit: Yuxiong Wang

Transfer Learning

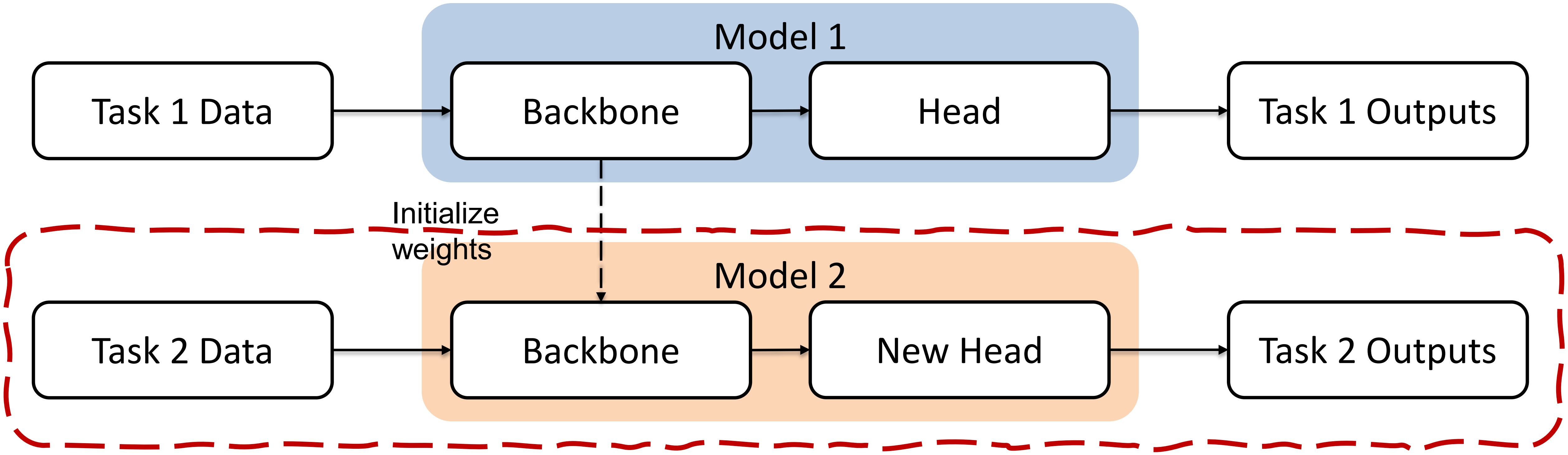
- Step 2: Initialize weights using learned Model 1



Slide credit: Yuxiong Wang

Transfer Learning

- Step 3: Fine-tune Model 2 on Task 2
  - Backbone may use a smaller learning rate or even be "frozen"



Slide credit: Yuxiong Wang

Model-Agnostic Meta-Learning (MAML)

- Proposed by Finn et al. ICML '17

- Goal: To learn a good **parameter initialization** that can be quickly adapted to new tasks

- Model-agnostic: Can be applied to any differentiable model
  – Flexible, can be used in a wide range of applications
  – Including computer vision, natural language processing, and robotics

Slide credit: Yuxiong Wang

Model-Agnostic Meta-Learning (MAML)

- Assumption and setting
  - Have a pool of various tasks
  - Each task contains a set of training/validation samples

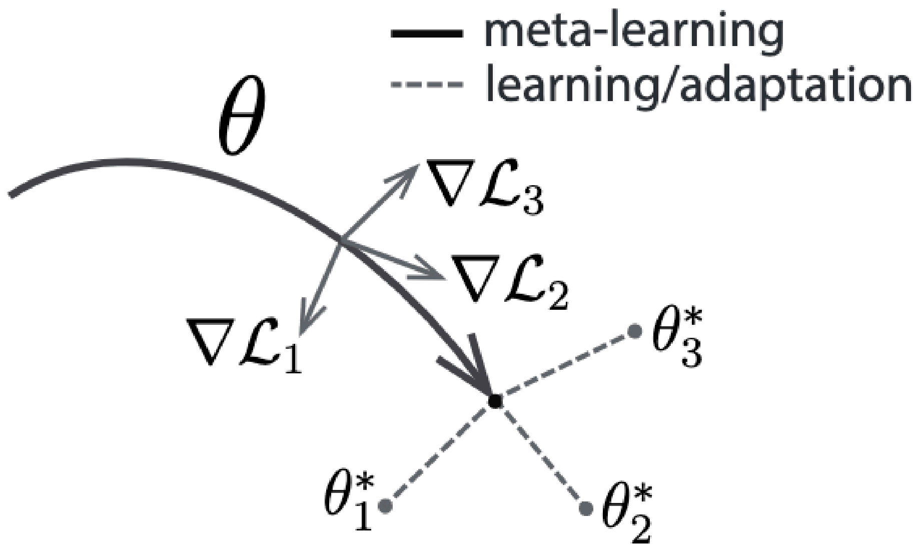
- An example of task pool
  - Classify Dogs into Shepherd, Labrador, Golden, Husky ...
  - Classify Cat into Siamese, Maine, Persian, Shorthair ...
  - Classify Bird into Canary, Parrot, Dove, Sparrow ...

Slide credit: Yuxiong Wang

Model-Agnostic Meta-Learning (MAML)

- Meta-learning phase
  - Use pool of tasks to obtain a good parameter initialization
  - Learn from the "experience of learning"

- Adaptation phase
  - Use few samples and optimization steps tc adapt to new task
  - New task can be outside the task pool used in meta-learning



Slide credit: Yuxiong Wang

Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
    Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    Sample datapoints $\mathcal{D}'_i = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$ using each $\mathcal{D}'_i$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
end while

Find gradient step(s) to improve parameters for each few-shot task

Update parameters so that those update steps reduce the loss as much as possible for all tasks

MAML is “learning to learn” – it learns parameters that are close to good parameters for many classification tasks, so that new tasks can be learned from a few examples and optimization steps

Methods for Domain Adaptation

- Instance translation
  - Transform target-domain data into source-domain

- Domain adversarial training
  - Align source-domain and target-domain feature spaces

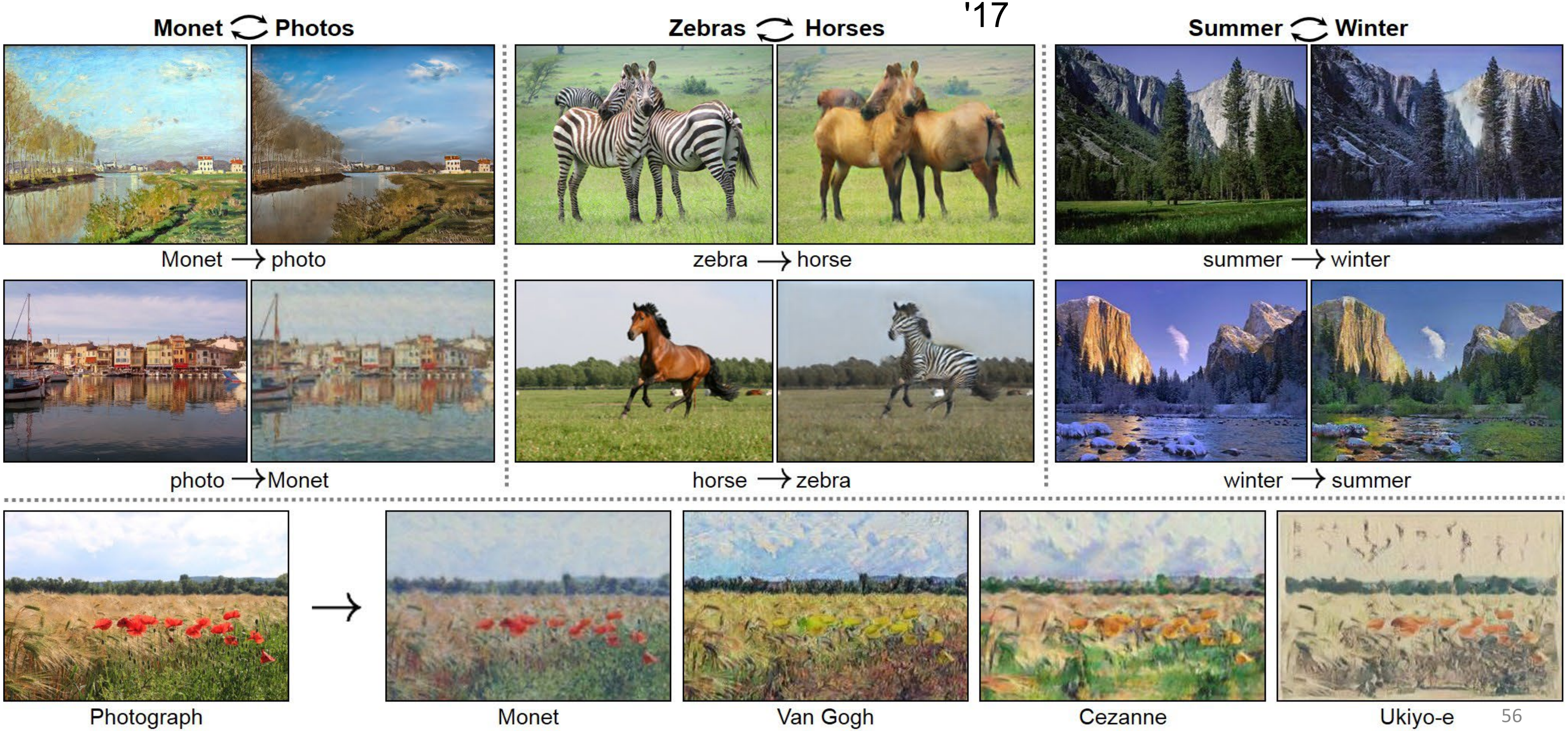
Instance Translation

- Use generative models (e.g., CycleGAN by Zhu et al. ICCV '17) to create instances
- Look like source domain but preserve same target domain content
- Then feed source-like instances into source-domain model ✅

Slide credit: Yuxiong Wang

Instance Translation

CycleGAN by Zhu et al. ICCV '17



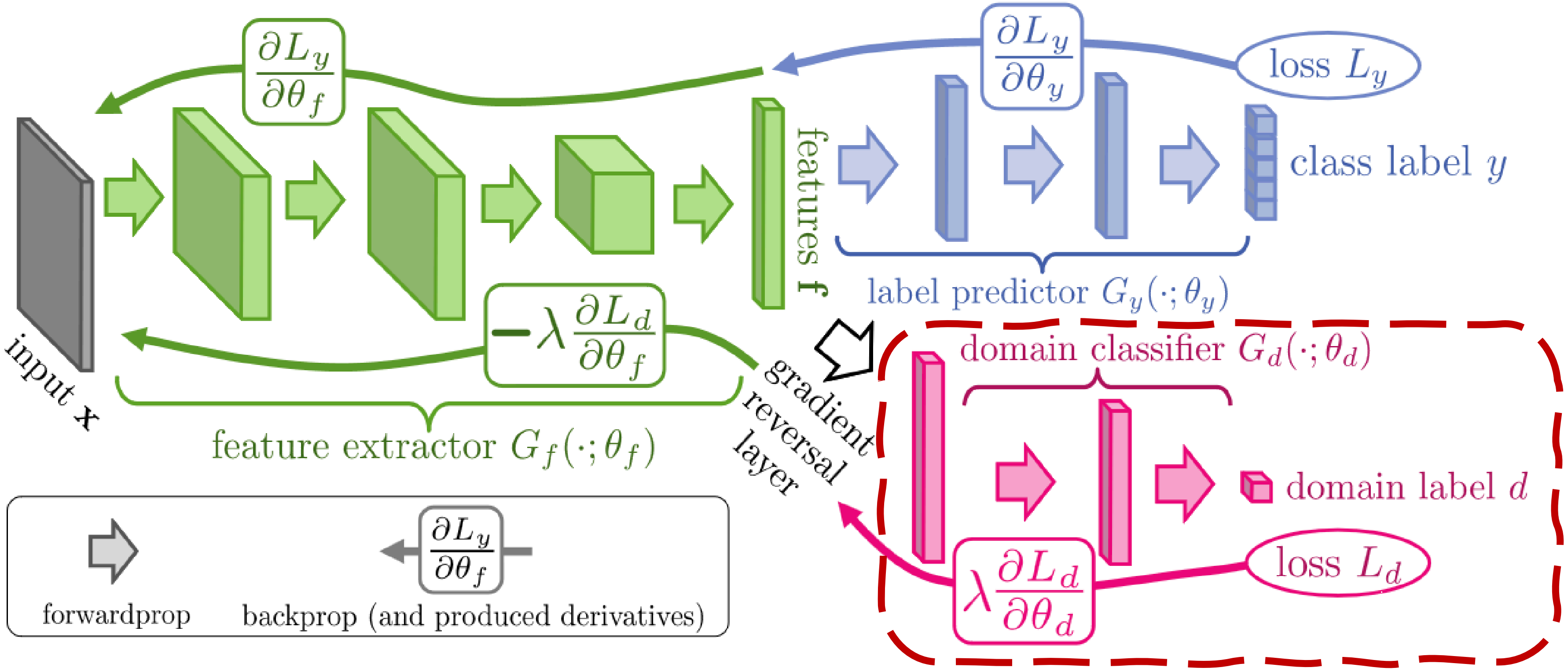
Slide credit: Yuxiong Wang

Domain Adversarial Training

- Proposed by Ganin et al. JMLR '17
- Goal: Learn a **domain-invariant** model
  - Model produces features that do not change with domain shift
  - Only reflect contents about labels, but not domain characteristics

Domain Adversarial Training

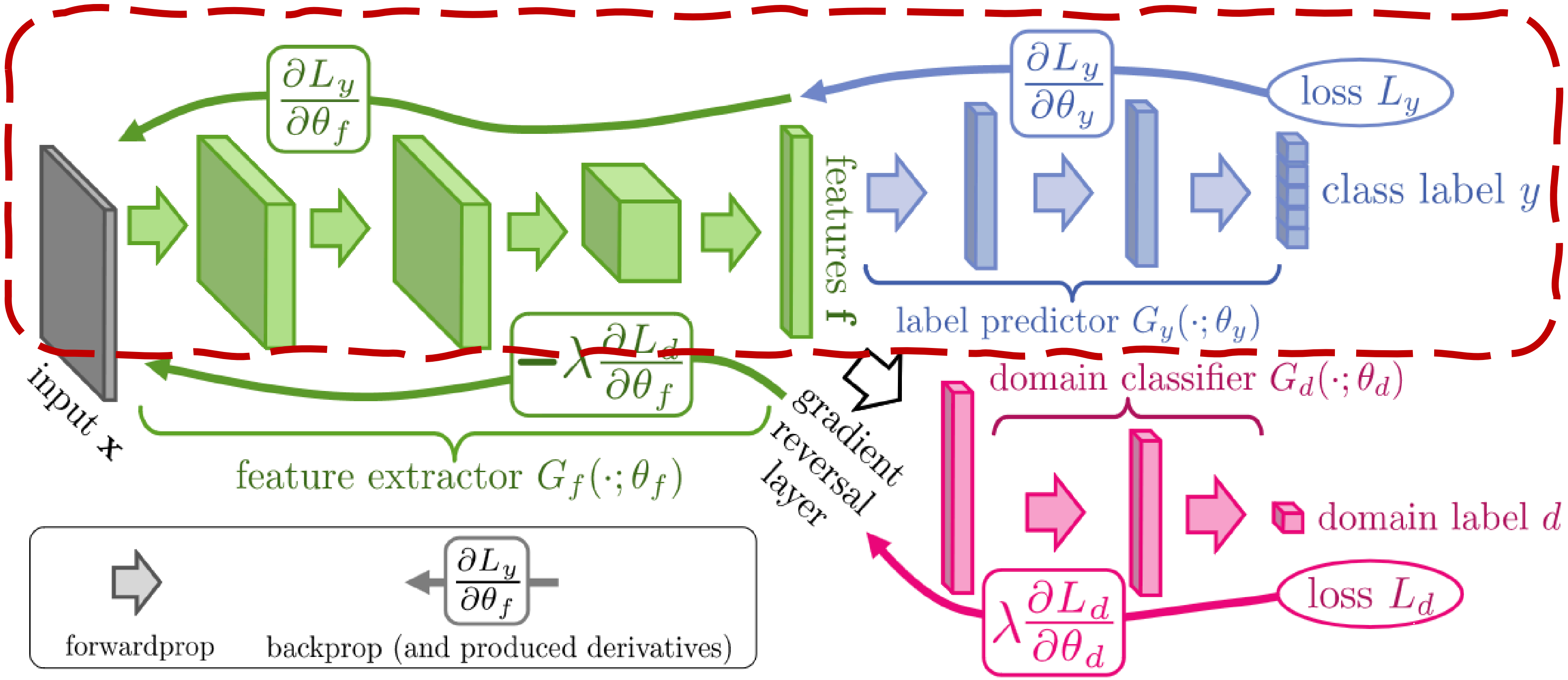
- Attach a domain classifier network and apply **adversarial** training
- Aim of domain classifier: To distinguish source vs. target domains



Slide credit: Yuxiong Wang

Domain Adversarial Training

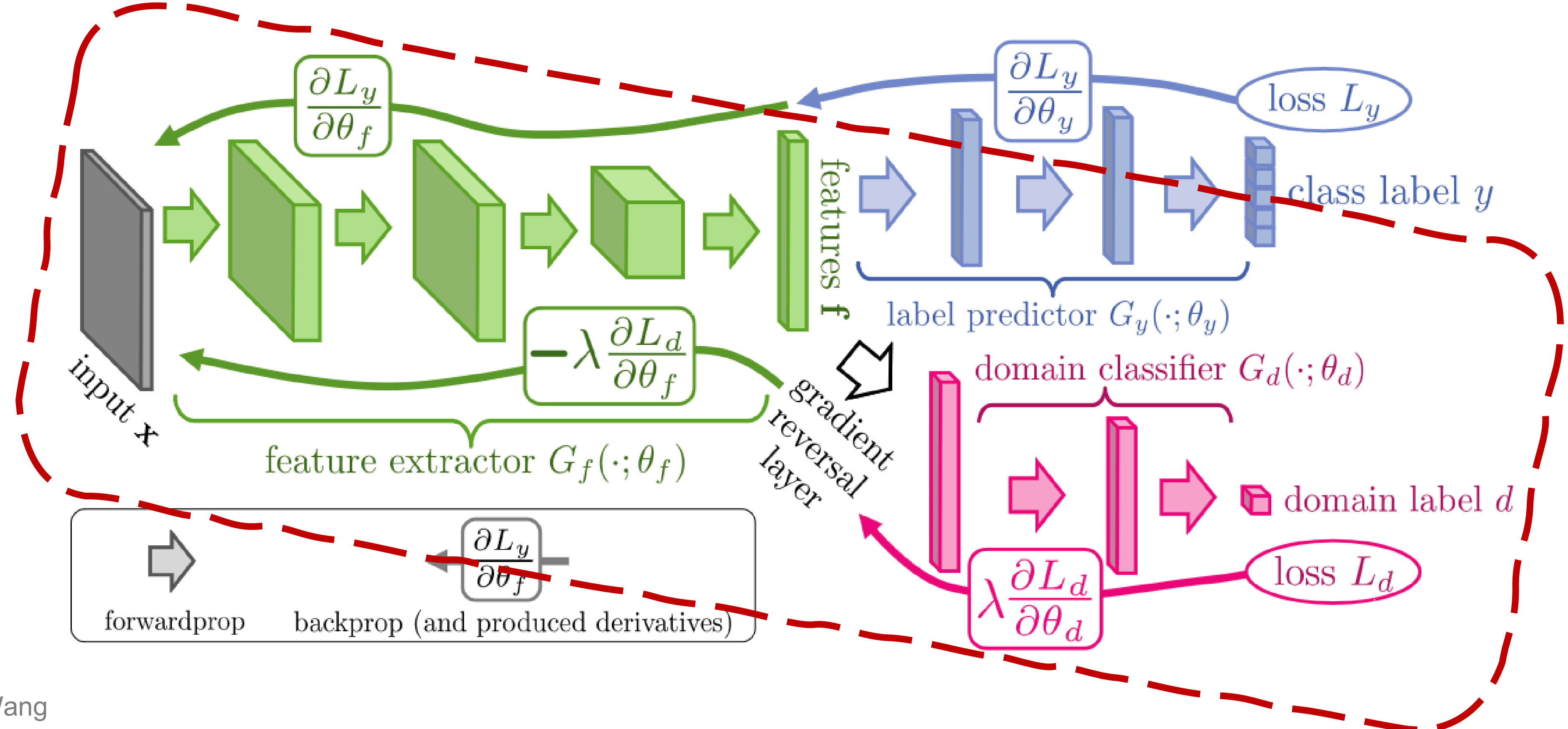
- Aim of main network: 1) Correctly predict label of source-domain data;



Slide credit: Yuxiong Wang

Domain Adversarial Training

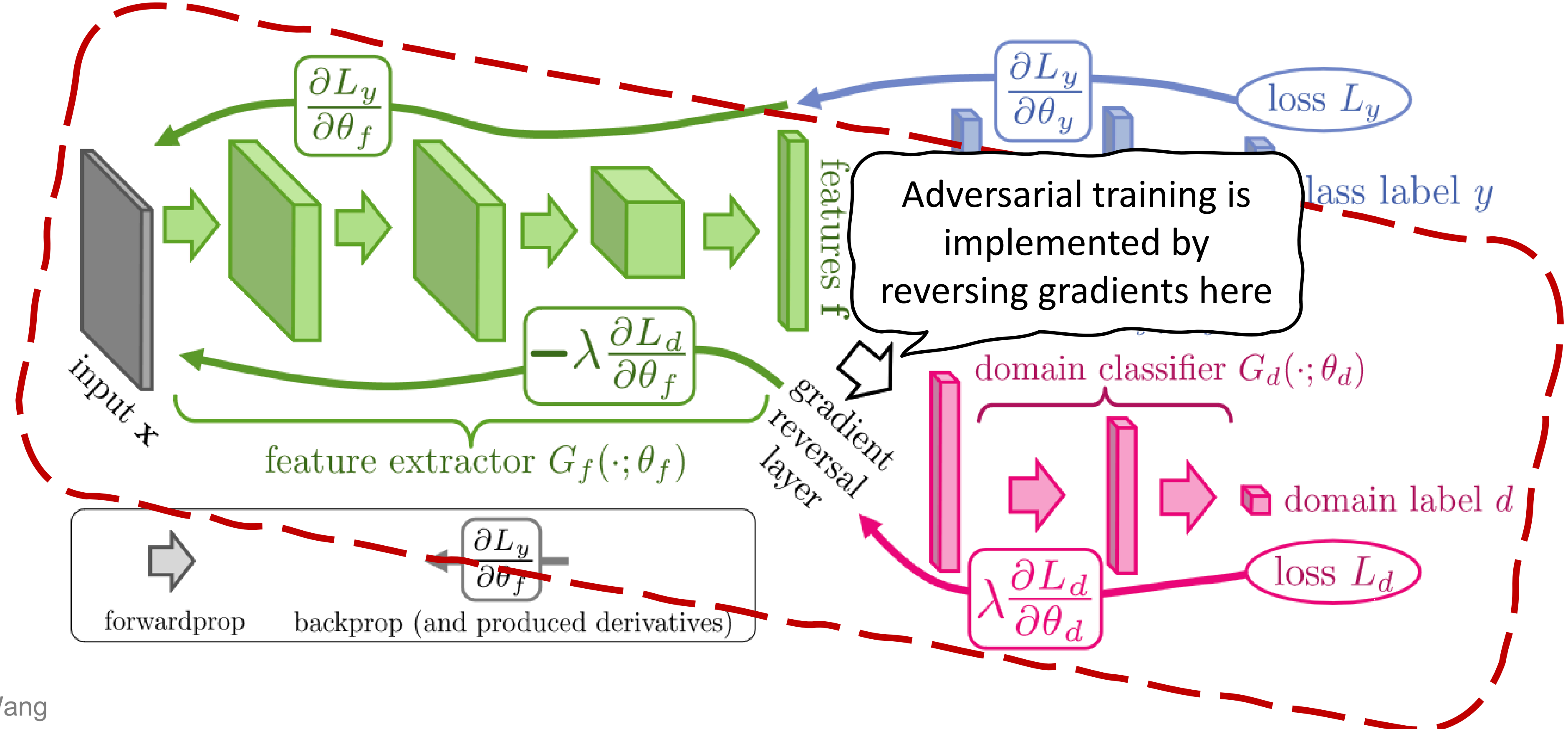
- Aim of main network: 1) Correctly predict label of source-domain data; 2) Using features that cannot distinguish between source and target domains



Slide credit: Yuxiong Wang

Domain Adversarial Training

- Adversarial training: Domain classifier θ_d **minimizes** discrimination loss L_d, while main network's feature extractor θ_f **maximizes** L_d



Slide credit: Yuxiong Wang

Domain Adversarial Training

- One mainstream of domain adaptation
  - Various follow-up methods study how to better learn **domain-invariant** models or feature representations

- Other ideas (may be combined with domain adversarial training)
  - Instance translation
  - Pseudo-labeling and self-training
  - Domain randomization

Slide credit: Yuxiong Wang

Summary

- Task adaptation for changed task objective
  - Transfer learning
  - Meta-learning
- Domain adaptation for changed data distribution
  - Instance translation
  - Domain adversarial training

Slide credit: Yuxiong Wang

Coming up

- Thursday: Ethics and Impact of AI