ROB 498/599: Deep Learning for Robot Perception (DeepRob)

Lecture 10: Training Neural Networks - Part 2

02/12/2025

https://deeprob.org/w25/

ROBOTICS

Today

- Feedback and Recap (5min)
- Training NNs
  - Learning Rate scheduling
  - Choosing Hyperparameters
  - Model Ensembles, Transfer Learning
- Summary and Takeaways (5min)

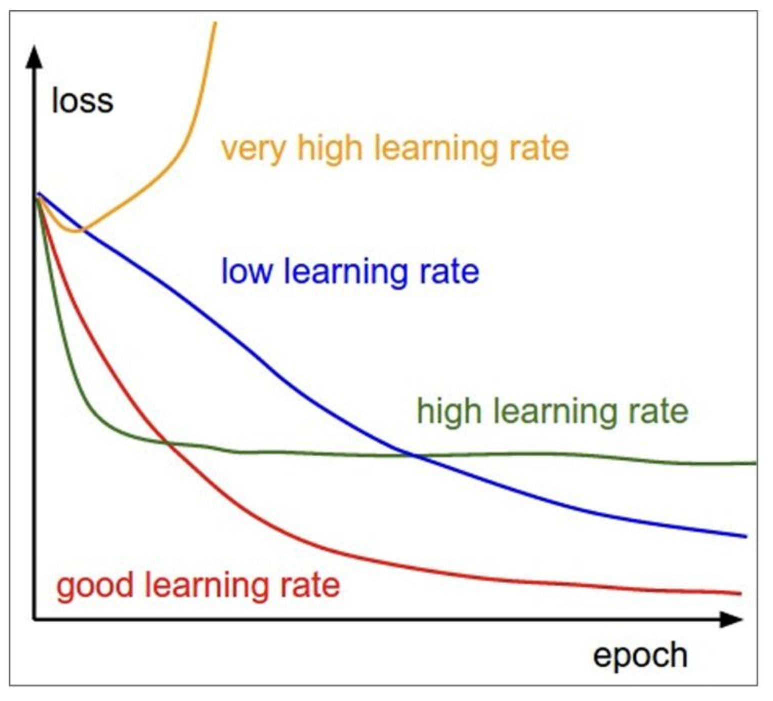
Recap

1. **One time setup:** **Last time**
   - Activation functions, data preprocessing, weight initialization, regularization
2. **Training dynamics:** **Today**
   - Learning rate schedules; large-batch training; hyperparameter optimization
3. **After training:**
   - Model ensembles, transfer learning

1. **One time setup:**
   - Activation functions, data preprocessing, weight initialization, regularization
2. **Training dynamics:** **Now**
   - Learning rate schedules; large-batch training; hyperparameter optimization
3. **After training:**
   - Model ensembles, transfer learning

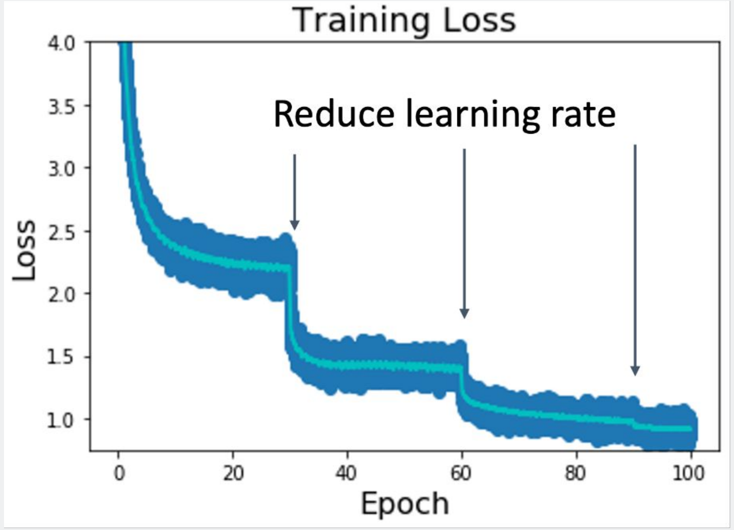
Learning Rate Scheduling

SGD, SGD+Momentum, Adagrad, RMSProp, Adam all have **learning rate** as hyper parameter

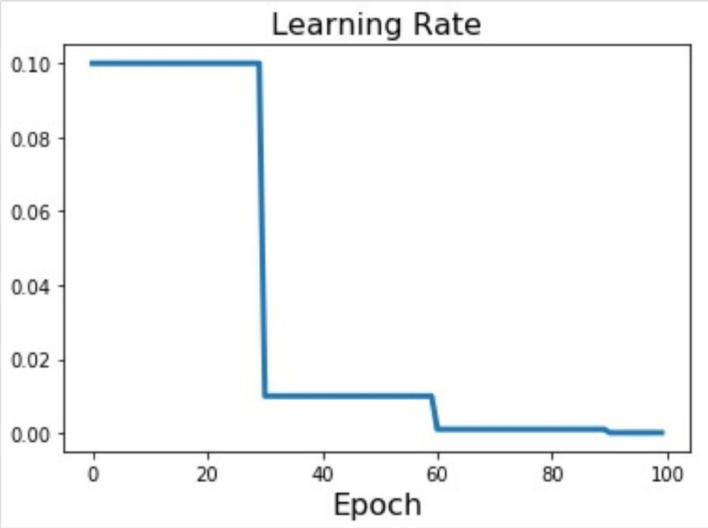


Q: Which one of these learning rates is **best** to use?

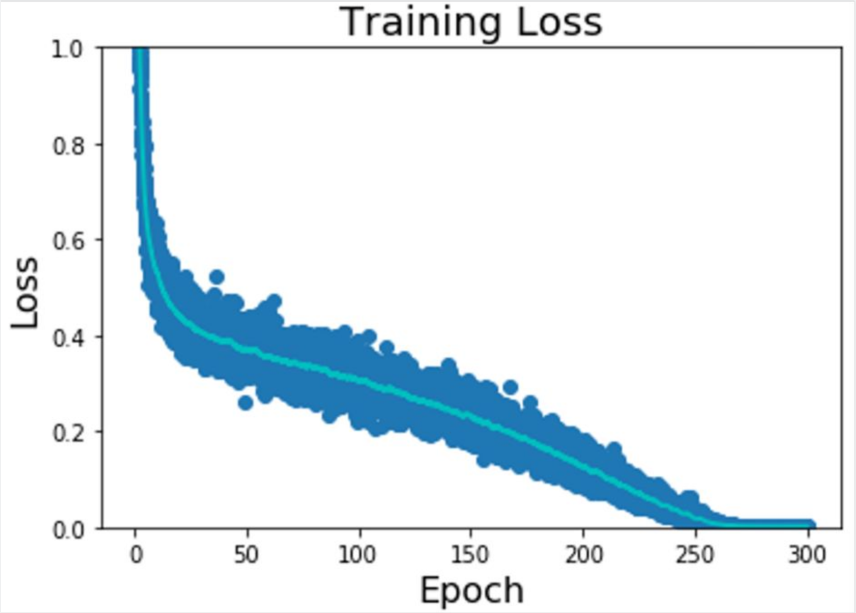
Learning Rate Decay: Step



Step: Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

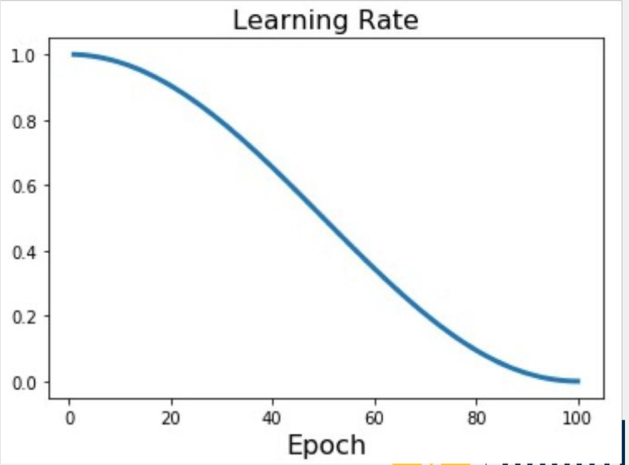


Learning Rate Decay: Cosine



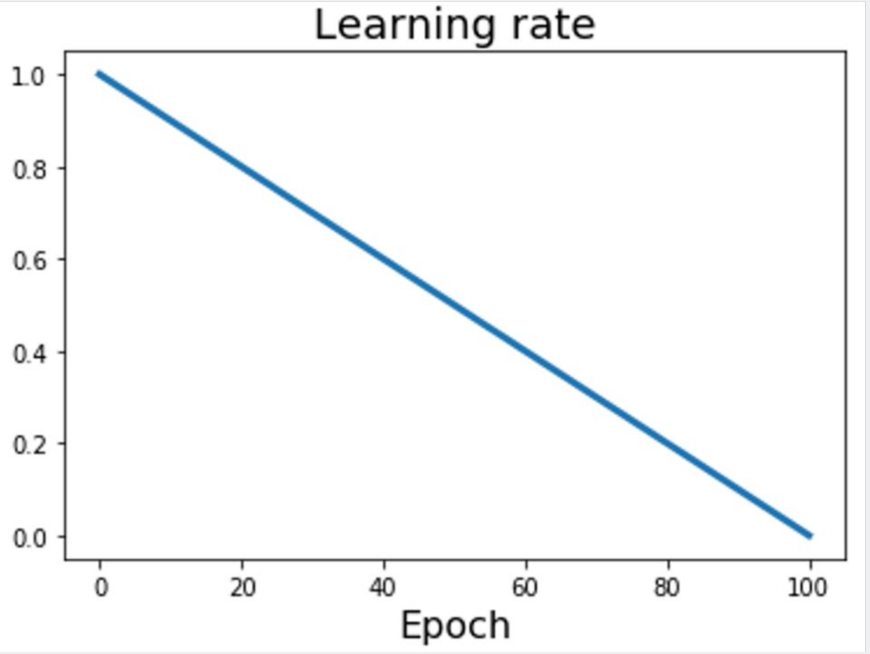
Step: Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

Cosine: $\alpha_t = \frac{1}{2}\alpha_0(1 + \cos(\frac{t\pi}{T}))$



Loshchilov and Hutter, “SGDR: Stochastic Gradient Descent with Warm Restarts”, ICLR 2017
Radford et al, “Improving Language Understanding by Generative Pre-Training”, 2018
Feichtenhofer et al, “SlowFast Networks for Video Recognition”, ICCV 2019
Radosavovic et al, “On Network Design Spaces for Visual Recognition”, ICCV 2019
Child at al, “Generating Long Sequences with Sparse Transformers”, arXiv 2019

Learning Rate Decay: Linear



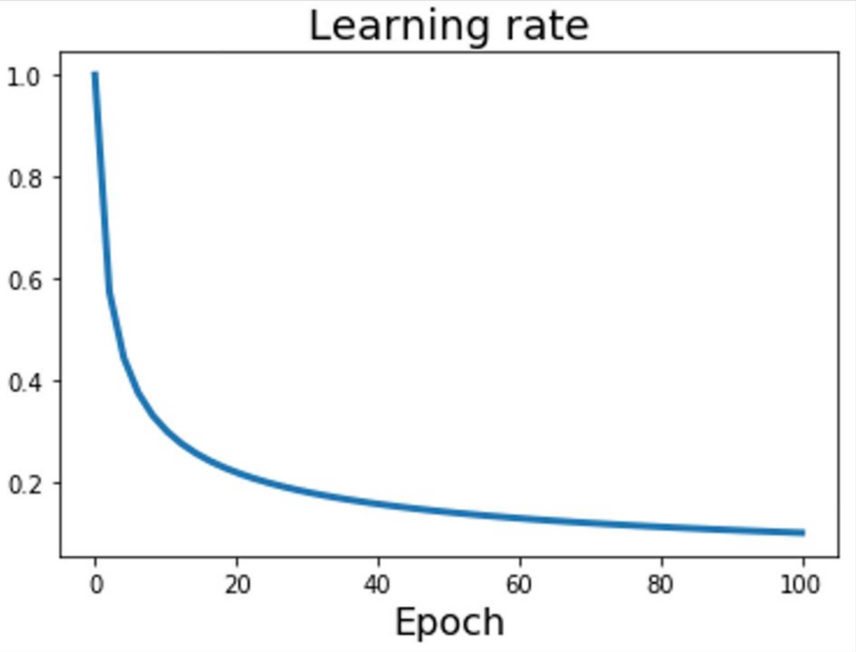
Step: Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

Cosine: $\alpha_t = \frac{1}{2}\alpha_0(1 + \cos(\frac{t\pi}{T}))$

Linear: $\alpha_t = \alpha_0(1 - \frac{t}{T})$

Devlin et al, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, NAACL 2018
Liu et al, “RoBERTa: A Robustly Optimized BERT Pretraining Approach”, 2019Yang et al, “XLNet: Generalized Autoregressive Pretraining for Language Understanding”, NeurIPS 2019

Learning Rate Decay: Inverse Sqrt



Step: Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

Cosine: $\alpha_t = \frac{1}{2}\alpha_0(1 + \cos(\frac{t\pi}{T}))$

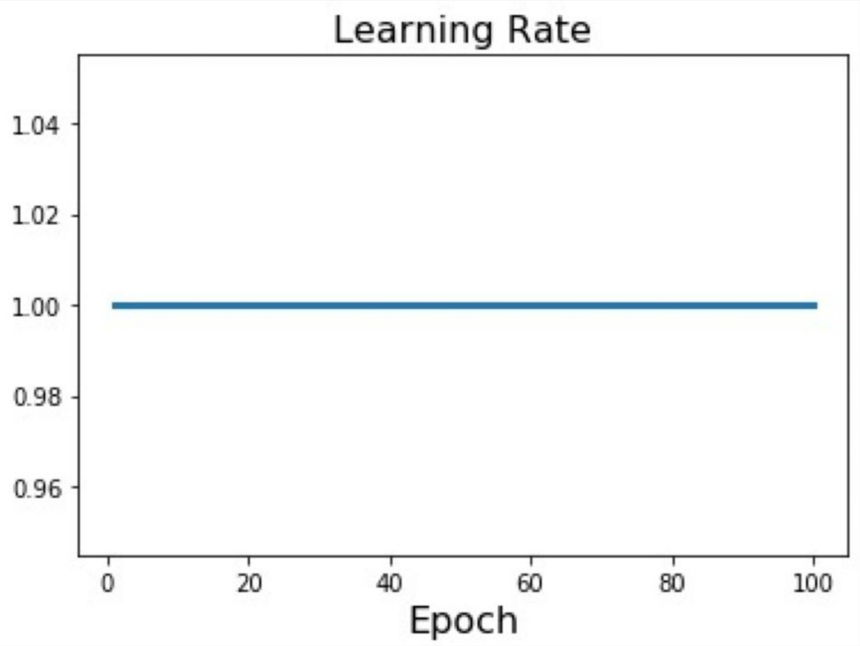
Linear: $\alpha_t = \alpha_0(1 - \frac{t}{T})$

Inverse sqrt: $\alpha_t = \alpha_0/\sqrt{t}$

Vaswani et al, "Attention is all you need", NIPS 2017

ROBOTICS

Learning Rate Decay: Constant



Step: Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

Cosine: $\alpha_t = \frac{1}{2}\alpha_0(1 + \cos(\frac{t\pi}{T}))$

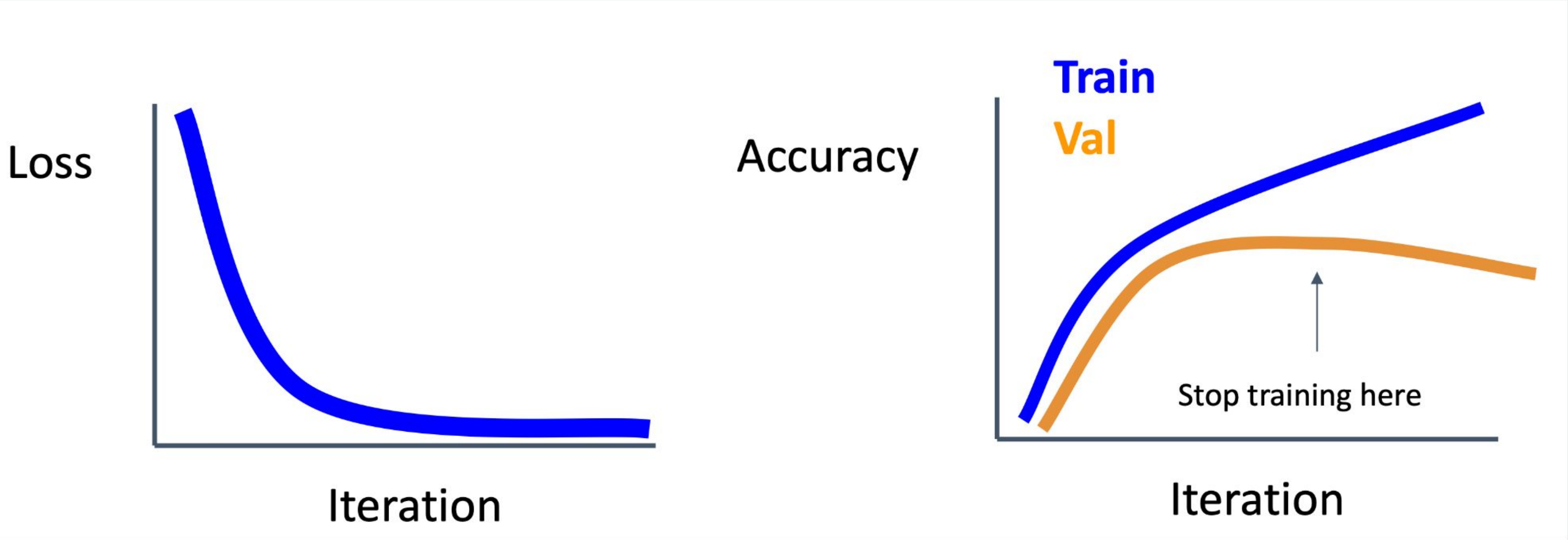
Linear: $\alpha_t = \alpha_0(1 - \frac{t}{T})$

Inverse sqrt: $\alpha_t = \alpha_0/\sqrt{t}$

Constant: $\alpha_t = \alpha_0$

Brock et al, "Large Scale GAN Training for High Fidelity Natural Image Synthesis", ICLR 2019
Donahue and Simonyan, "Large Scale Adversarial Representation Learning", NeurIPS 2019

How long to train? Early Stopping



Stop training the model when accuracy on the validation set decreases Or train for a long time, but always keep track of the model snapshot that worked best on val. **Always a good idea to do this!**

Choosing Hyperparameters

Choosing Hyperparameters: Grid Search

Choose several values for each hyper parameter
(Often space choices log-linearly)

Example:
Weight decay: [1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 1×10^{-1}]
Learning rate: [1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 1×10^{-1}]

Evaluate all possible choices on this **hyperparameter grid**

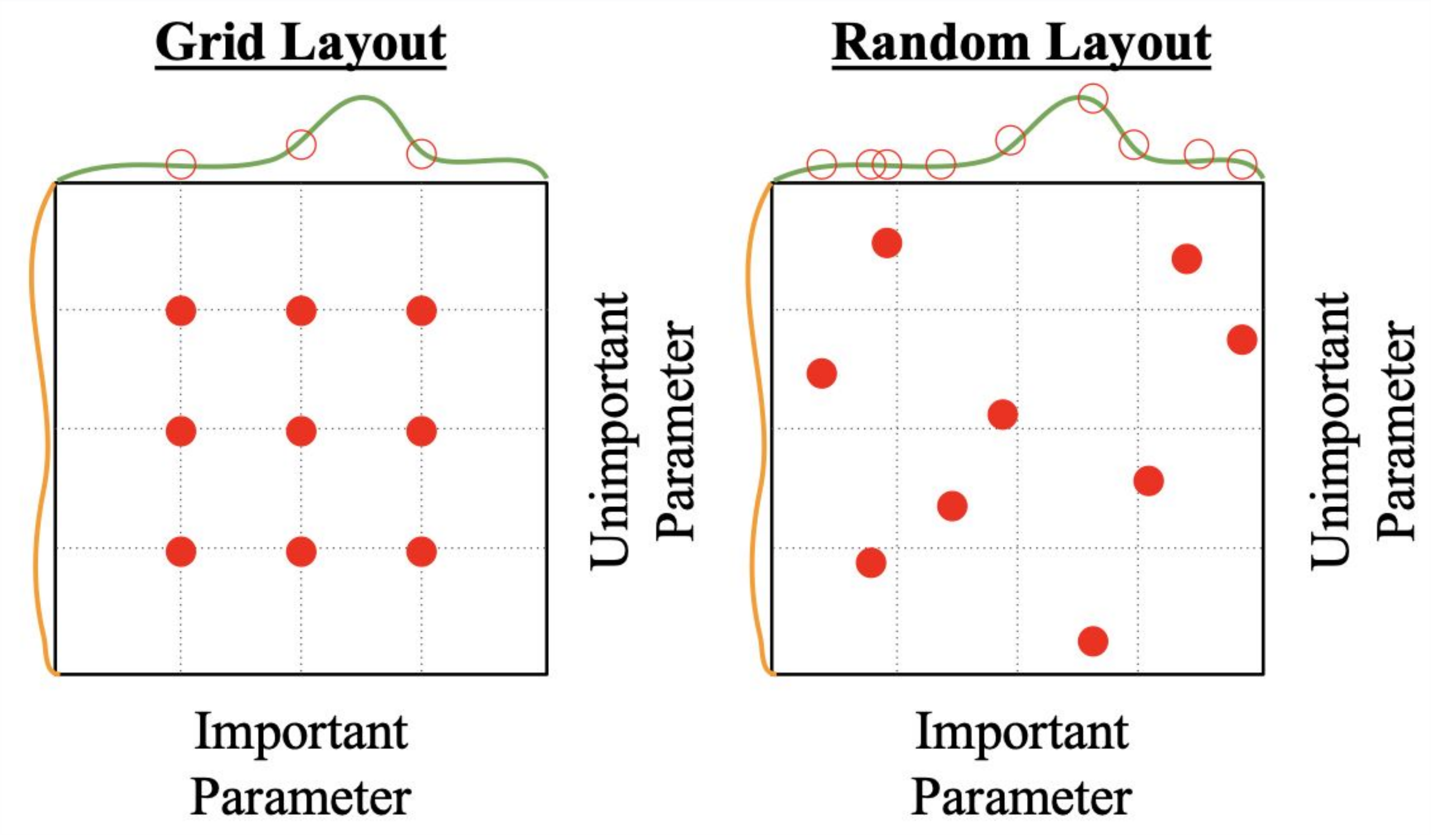
Choosing Hyperparameters: Random Search

Choose several values for each hyper parameter
(Often space choices log-linearly)

Example:
Weight decay: log-uniform on $[1 \times 10^{-4}, 1 \times 10^{-1}]$
Learning rate: log-uniform on $[1 \times 10^{-4}, 1 \times 10^{-1}]$

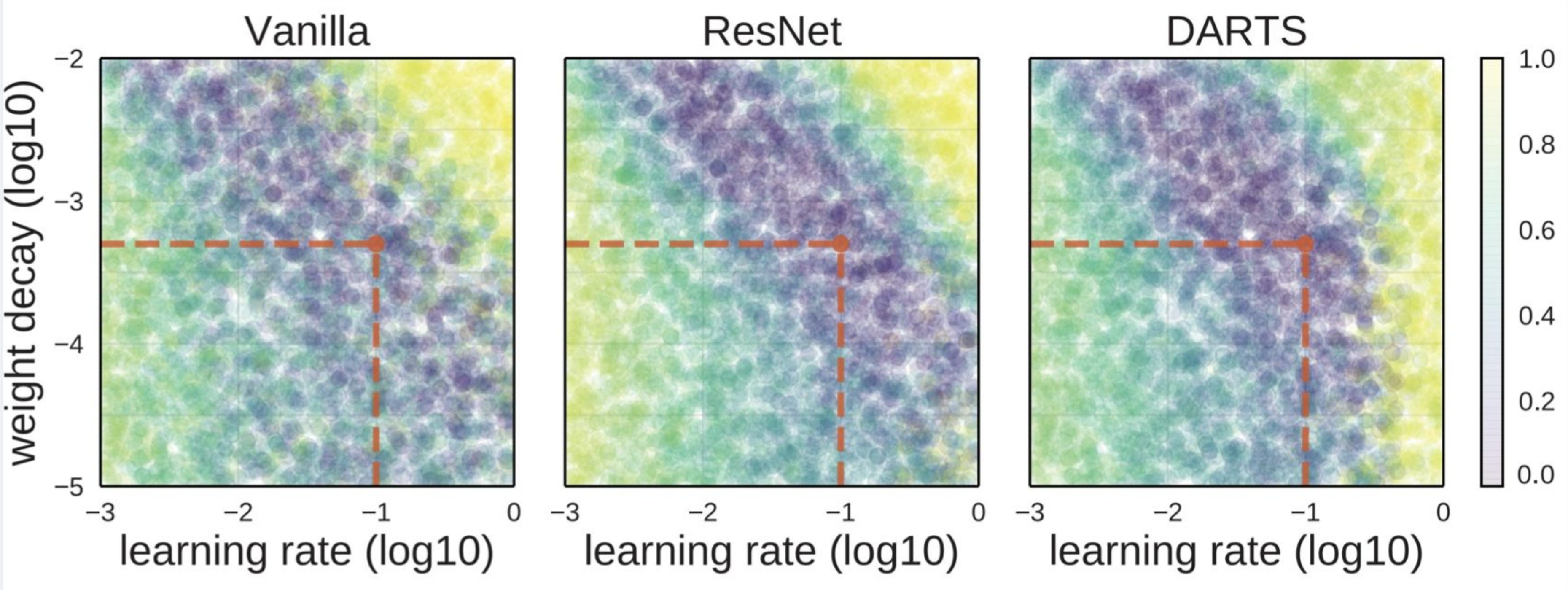
Run many different trials

Hyperparameters: Random vs Grid Search



Bergstra and Bengio, "Random Search for Hyper-Parameter Optimization", JMLR 2012

Choosing Hyperparameters: Random Search



Radosavovic et al, "On Network Design Spaces for Visual Recognition", ICCV 2019

Choosing Hyperparameters

(without tons of GPUs)

Choosing Hyperparameters

Step 1: Check initial loss

Turn off weight decay, sanity check loss at initialization
e.g. log(C) for softmax with C classes

Choosing Hyperparameters

Step 1: Check initial loss

Step 2: Overfit a small sample

Try to train to 100% training accuracy on a small sample of training data (~5-10 mini batches); fiddle with architecture, learning rate, weight initialization. Turn off regularization.

Loss not going down? LR too low, bad initialization
Loss explodes to Inf or NaN? LR too high, bad initialization

Choosing Hyperparameters

Step 1: Check initial loss

Step 2: Overfit a small sample

Step 3: Find LR that makes loss go down

Use the architecture from the previous step, use all training data, turn on small weight decay, find a learning rate that makes the loss drop significantly within ~100 iterations

Good learning rates to try: 1e-1, 1e-2, 1e-3, 1e-4

Choosing Hyperparameters

Step 1: Check initial loss

Step 2: Overfit a small sample

Step 3: Find LR that makes loss go down

Step 4: Coarse grid, train for ~1-5 epochs

Choose a few values of learning rate and weight decay around what worked from Step 3, train a few models for ~1-5 epochs

Good learning rates to try: 1e-4, 1e-5, 0

Choosing Hyperparameters

Step 1: Check initial loss

Step 2: Overfit a small sample

Step 3: Find LR that makes loss go down

Step 4: Coarse grid, train for ~1-5 epochs

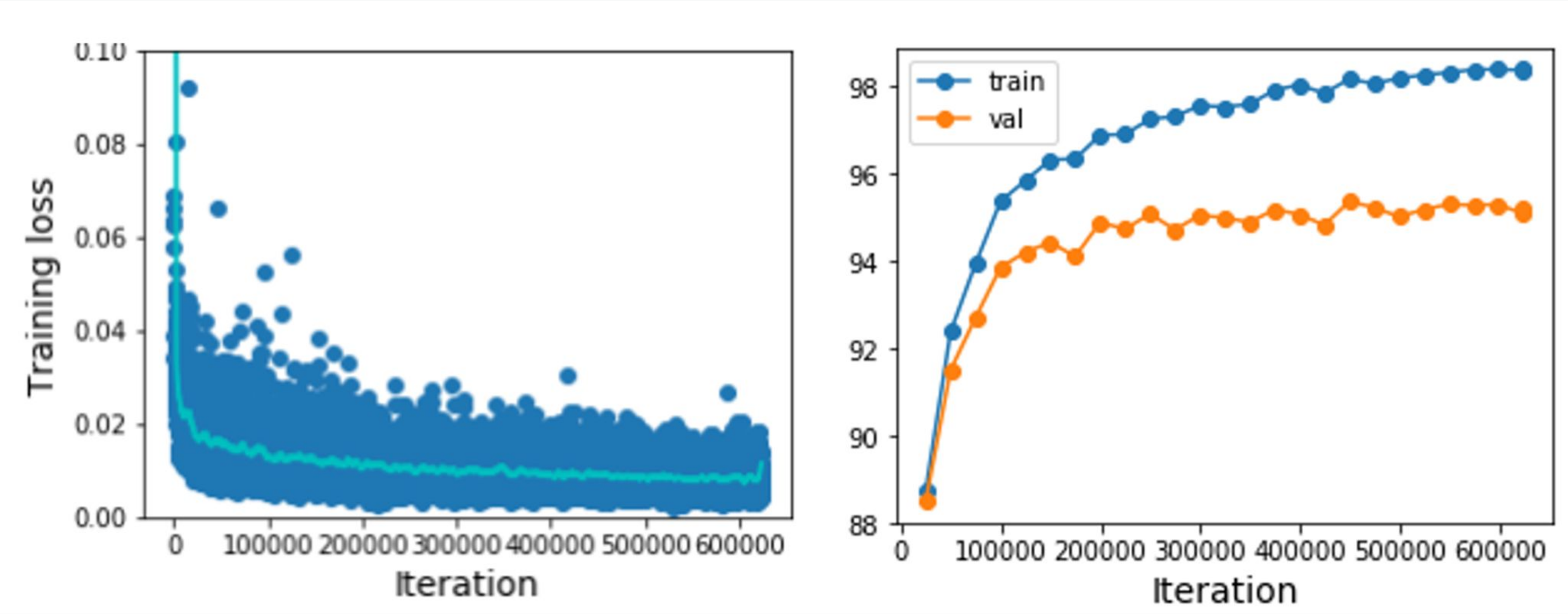
Step 5: Refine grid, train longer

Pick best models from Step 4, train them for longer (~10-20 epochs) without learning rate decay

Choosing Hyperparameters - Summary

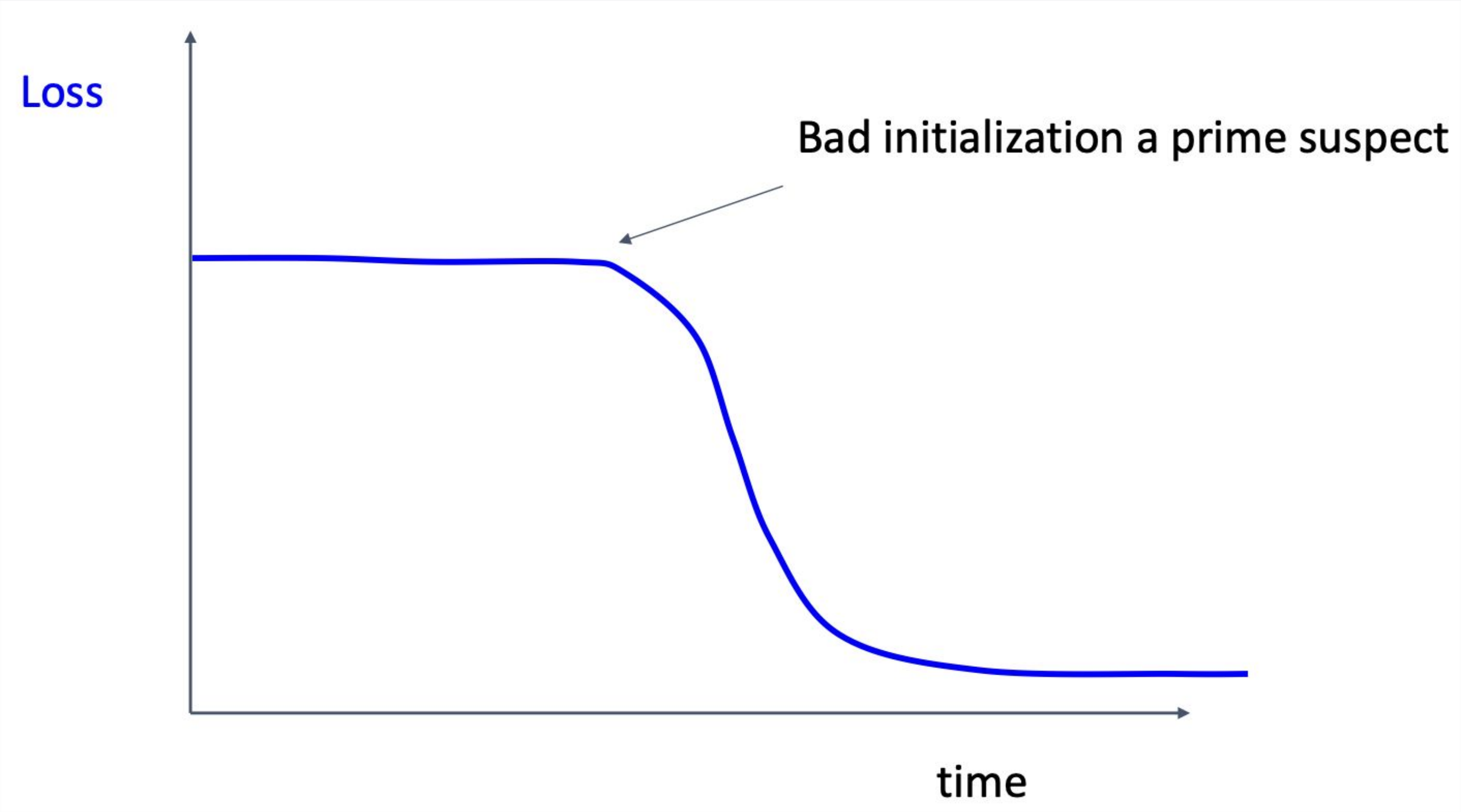
Step 1: Check initial loss
Step 2: Overfit a small sample
Step 3: Find LR that makes loss go down
Step 4: Coarse grid, train for ~1-5 epochs
Step 5: Refine grid, train longer
Step 6: Look at learning curves

Looking at Learning Curves

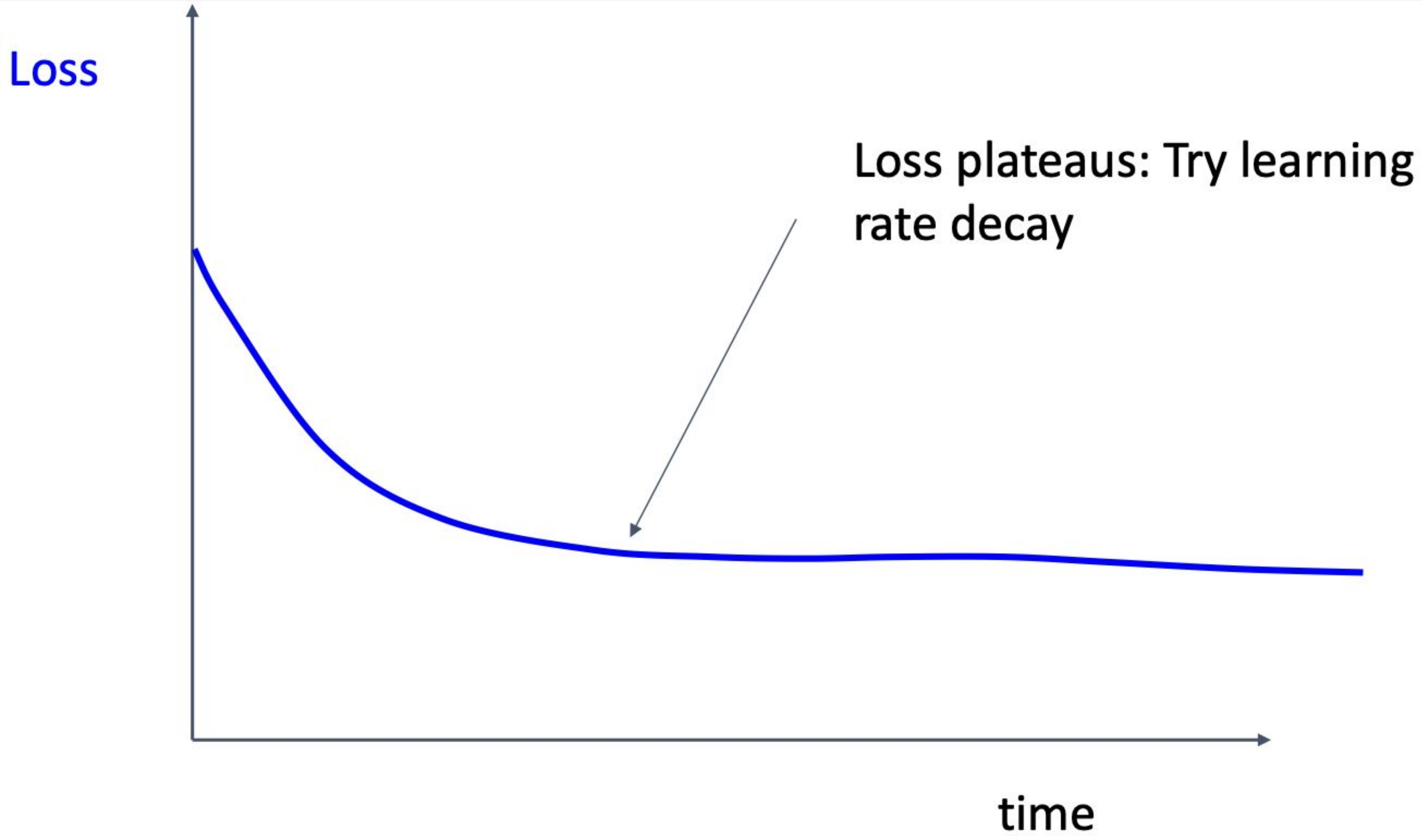


Losses may be noisy, use a scatter plot and also plot moving average to see trends better

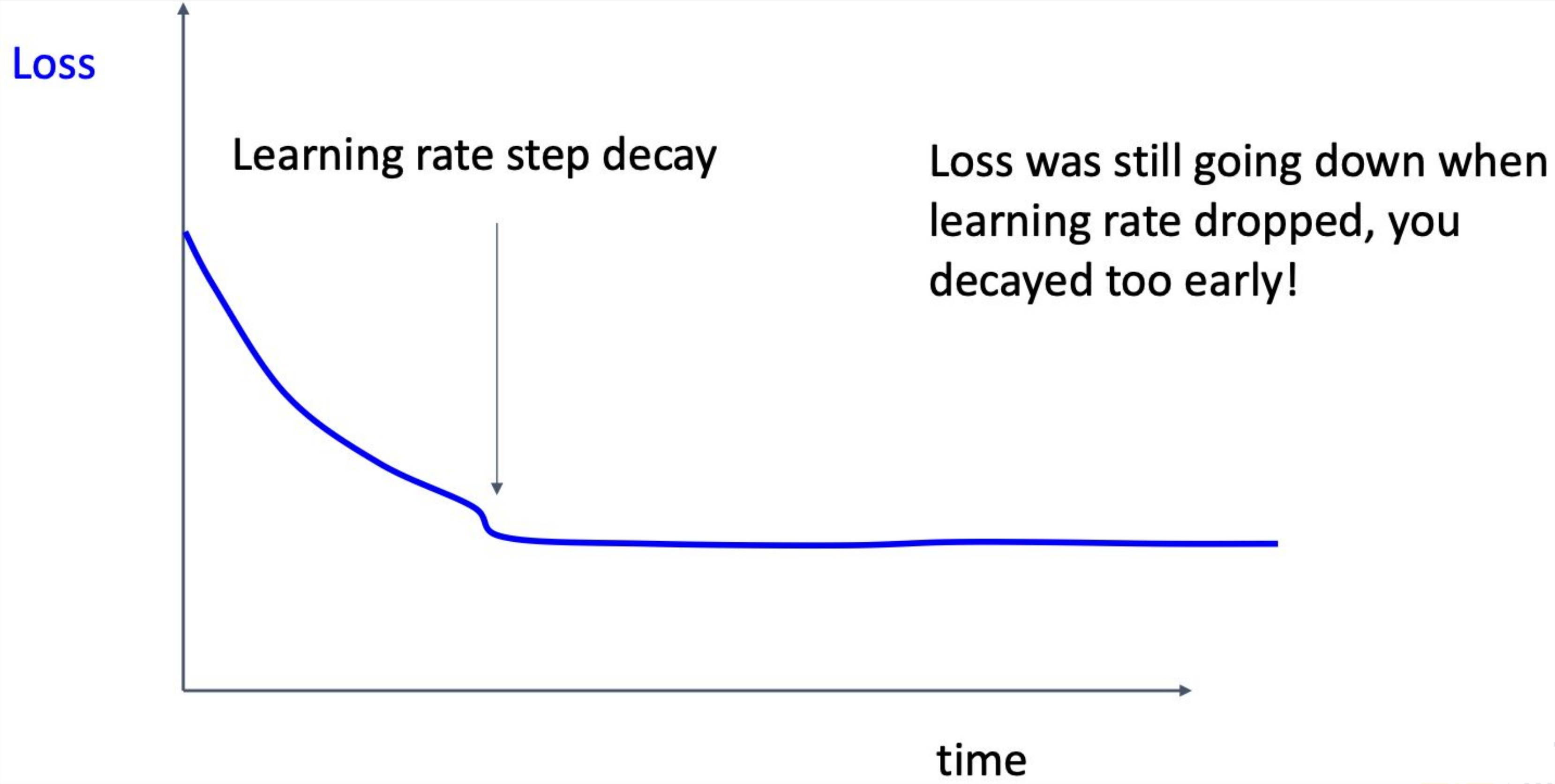
Choosing Hyperparameters



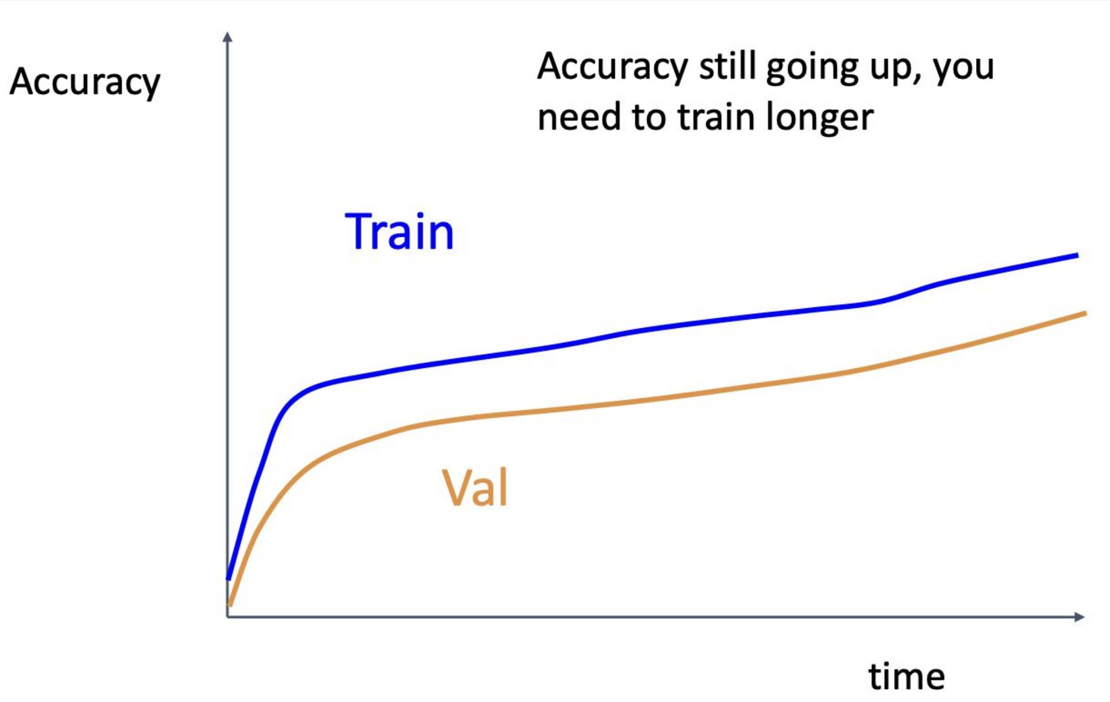
Choosing Hyperparameters



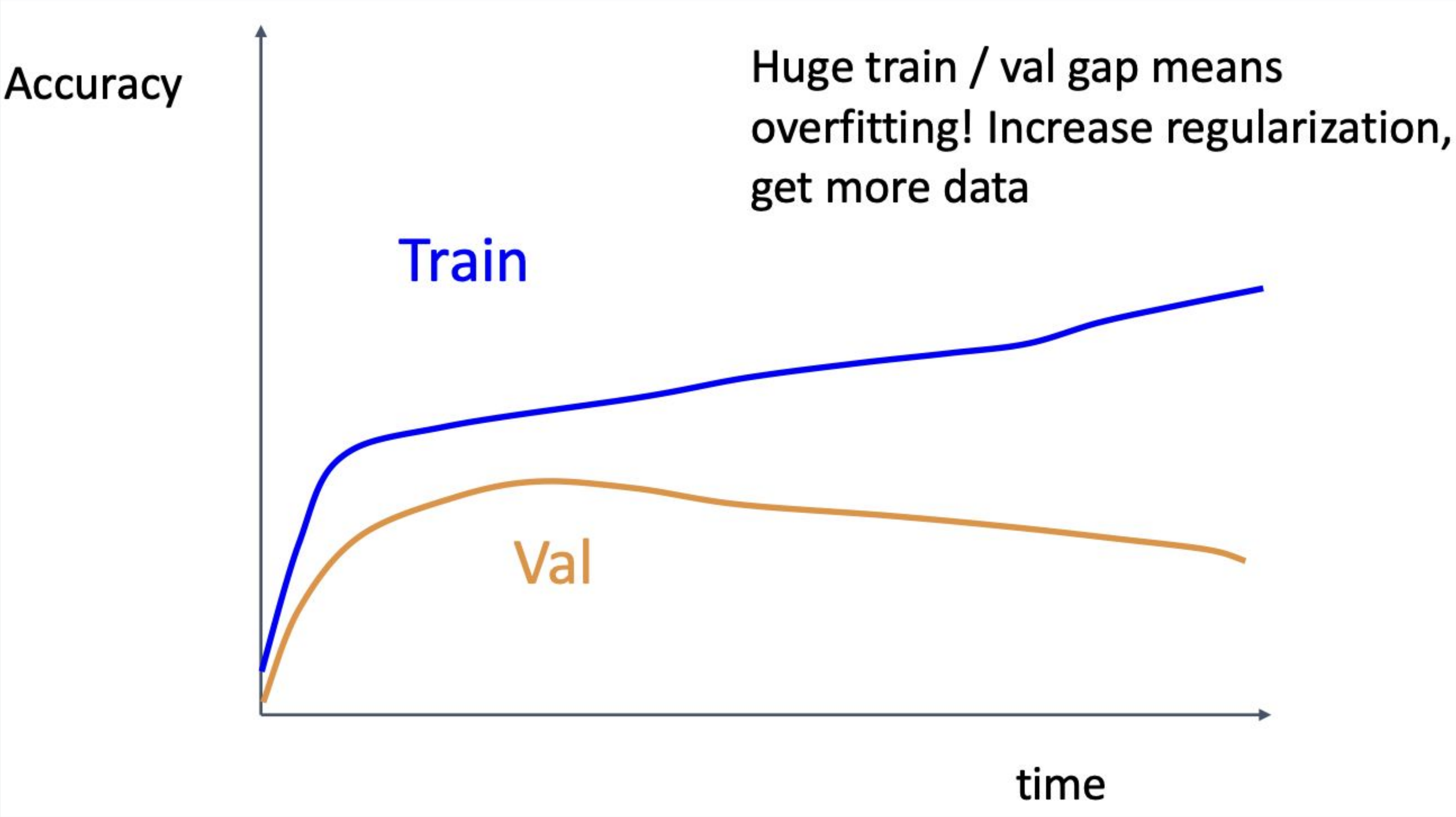
Choosing Hyperparameters



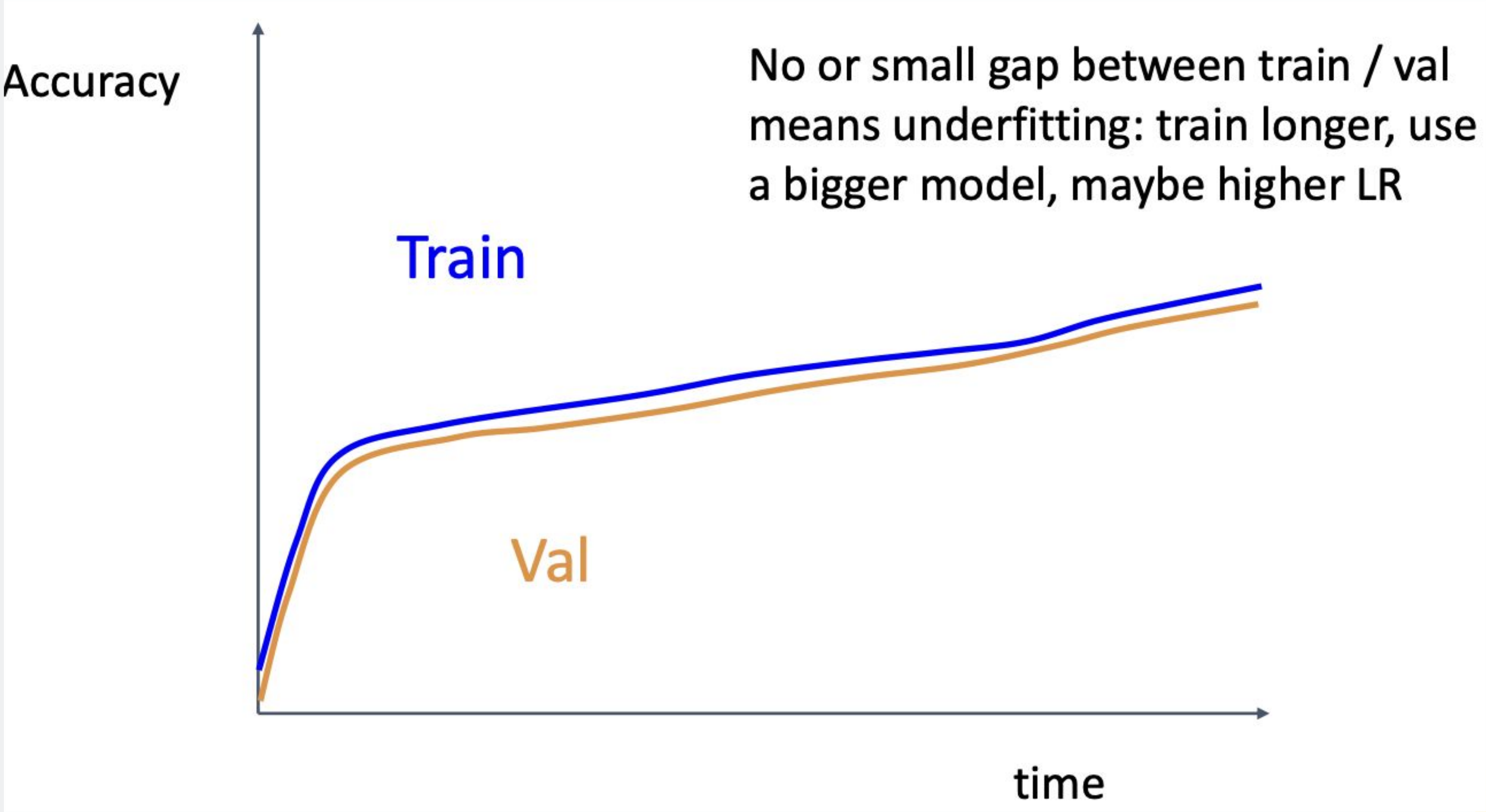
Choosing Hyperparameters



Choosing Hyperparameters



Choosing Hyperparameters



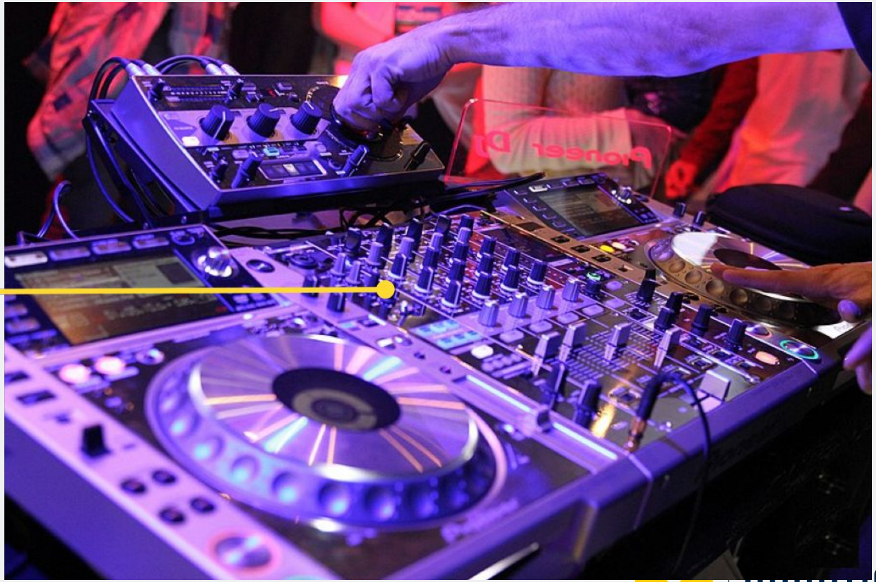
Choosing Hyperparameters

Step 1: Check initial loss
Step 2: Overfit a small sample
Step 3: Find LR that makes loss go down
Step 4: Coarse grid, train for ~1-5 epochs
Step 5: Refine grid, train longer
Step 6: Look at ~~learning curves~~ loss curves
Step 7: GOTO step 5

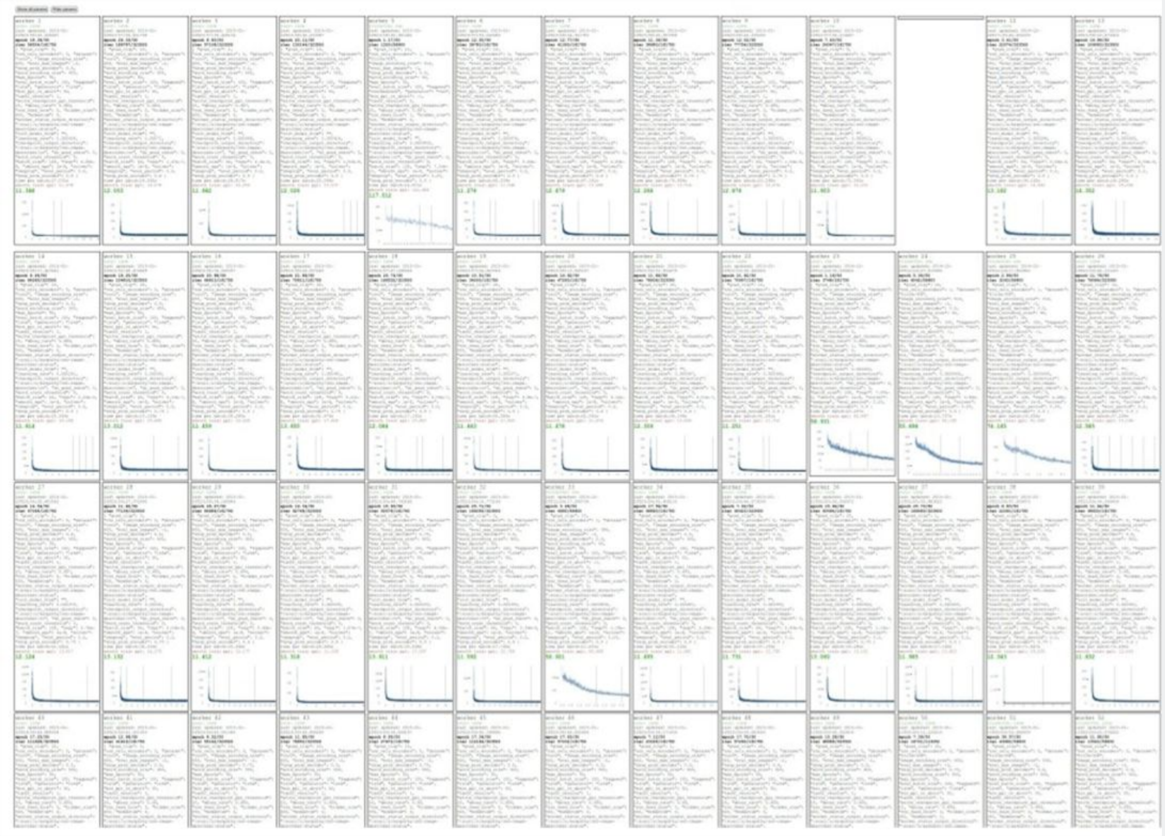
Hyperparameters to play with

- Network architecture
- Learning rate, its decay schedule, update type
- Regularization (L2/ Dropout strength)

Neural networks practitioner
Music = loss function



Cross-validation “command center”



- Tensorboard
- wandb.ai

https://docs.wandb.ai/tutorials/pytorch/

ROBOTICS

Track ratio of weight update / weight magnitude

```
# assume parameter vector W and its gradient vector dW
param_scale = np.linalg.norm(W.ravel())
update = -learning_rate*dW # simple SGD update
update_scale = np.linalg.norm(update.ravel())
W += update # the actual update
print update_scale / param_scale # want ~1e-3
```

Ratio between the updates and values: ~0.0002 / 0.02 = 0.01 (about okay)
want this to be somewhere around 0.001 or so

1. **One time setup:**
   - Activation functions, data preprocessing, weight initialization, regularization
2. **Training dynamics:**
   - Learning rate schedules; hyperparameter optimization
3. **After training:**
   - Model ensembles, transfer learning, large-batch training

Model Ensembles

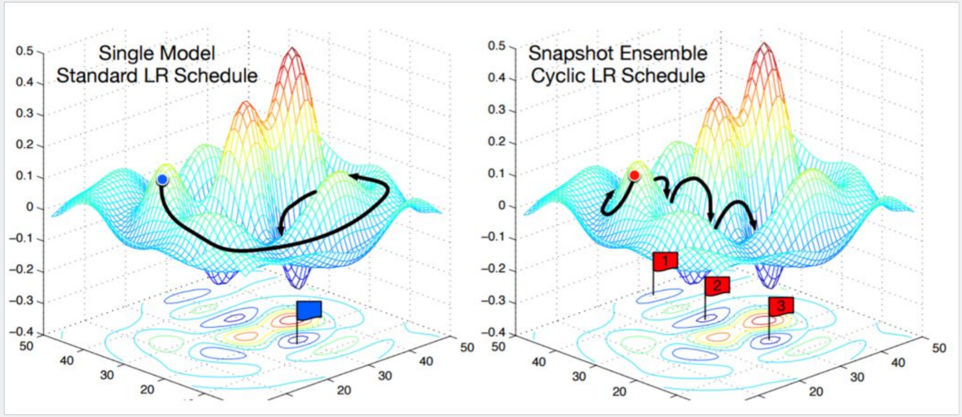
1. **Train multiple independent models**
2. **At test time average their results:**

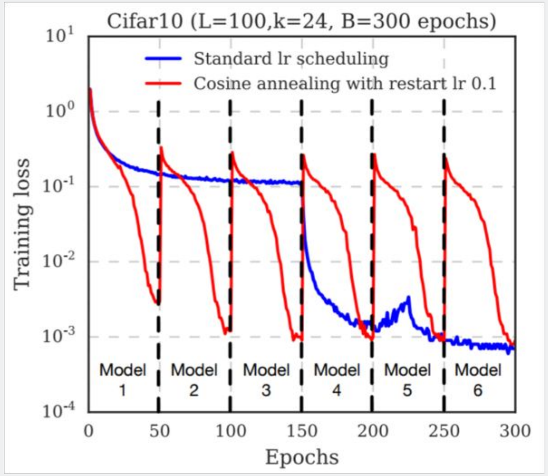
   (Take average of predicted probability distributions, then choose argmax)

Enjoy 2% extra performance

Model Ensembles: Tips and Tricks

Instead of training independent models, use multiple snapshots of a single model during training!





Loshchilov and Hutter, "SGDR: Stochastic gradient descent with restarts", arXiv 2016
Huang et al, "Snapshot ensembles: train 1, get M for free", ICLR 2017

Cyclic learning rate schedules can make this work even better!

Model Ensembles: Tips and Tricks

Instead of using actual parameter vector, keep a moving average of the parameter vector and use that at test time (Polyak averaging)

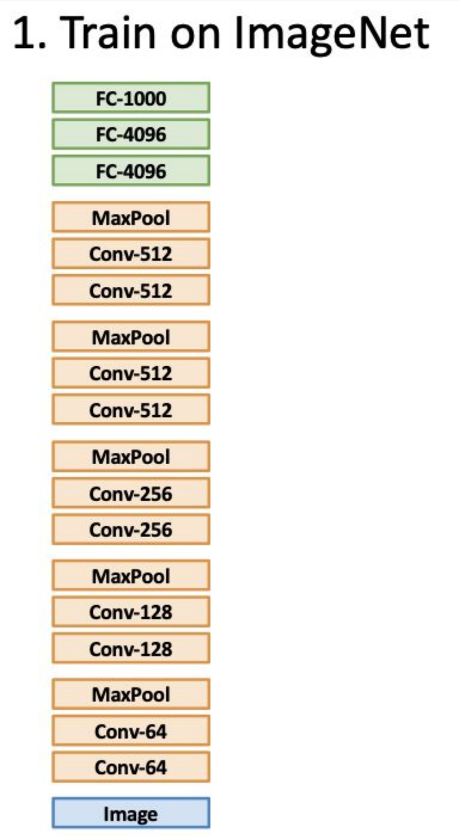
```
while True:
  data_batch = dataset.sample_data_batch()
  loss = network.forward(data_batch)
  dx = network.backward()
  x += - learning_rate * dx
  x_test = 0.995*x_test + 0.005*x # use for test set
```

Polyak and Juditsky, “Acceleration of stochastic approximation by averaging”, SIAM Journal on Control and Optimization, 1992.
Karras et al, “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018Brock et al, “Large Scale GAN Training for High Fidelity Natural Image Synthesis”, ICLR 201

Transfer Learning:

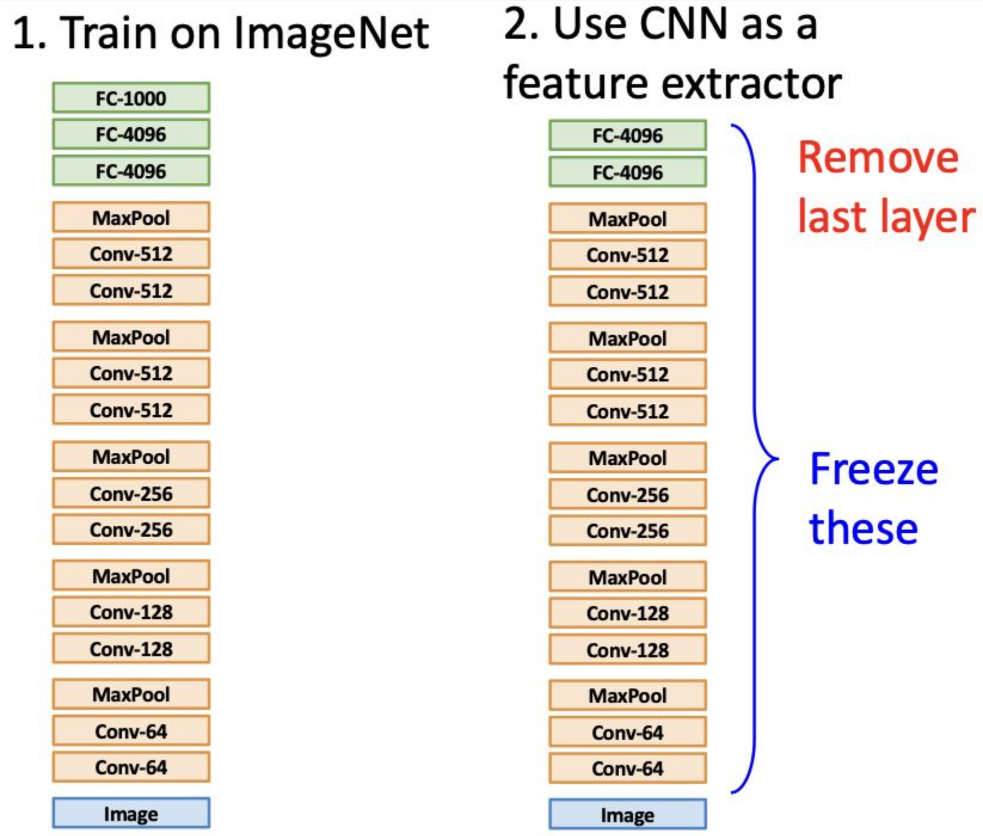
Generalizing to New Tasks

Transfer Learning with CNNs

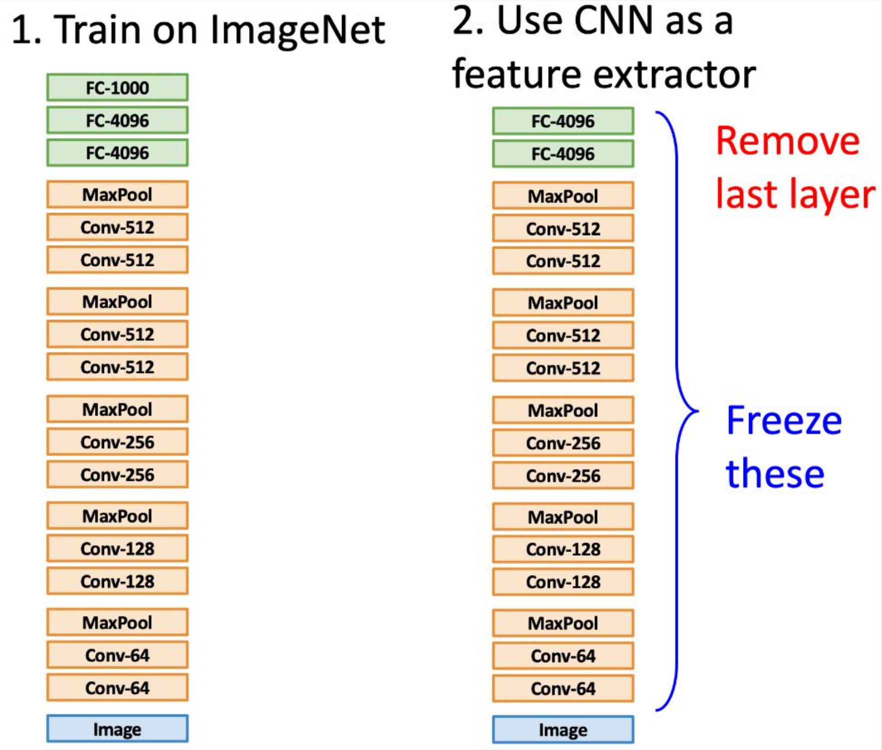


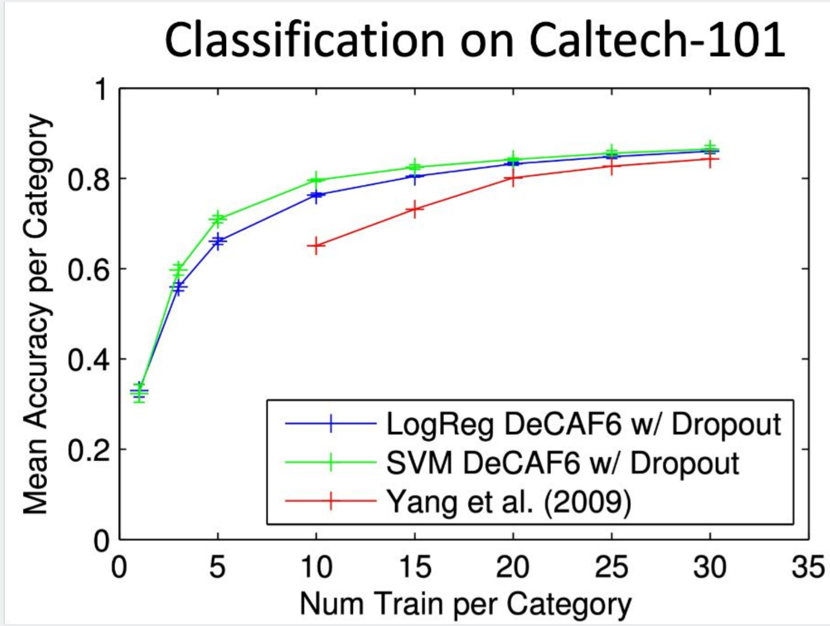
Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014
https://arxiv.org/abs/1310.1531

Transfer Learning with CNNs

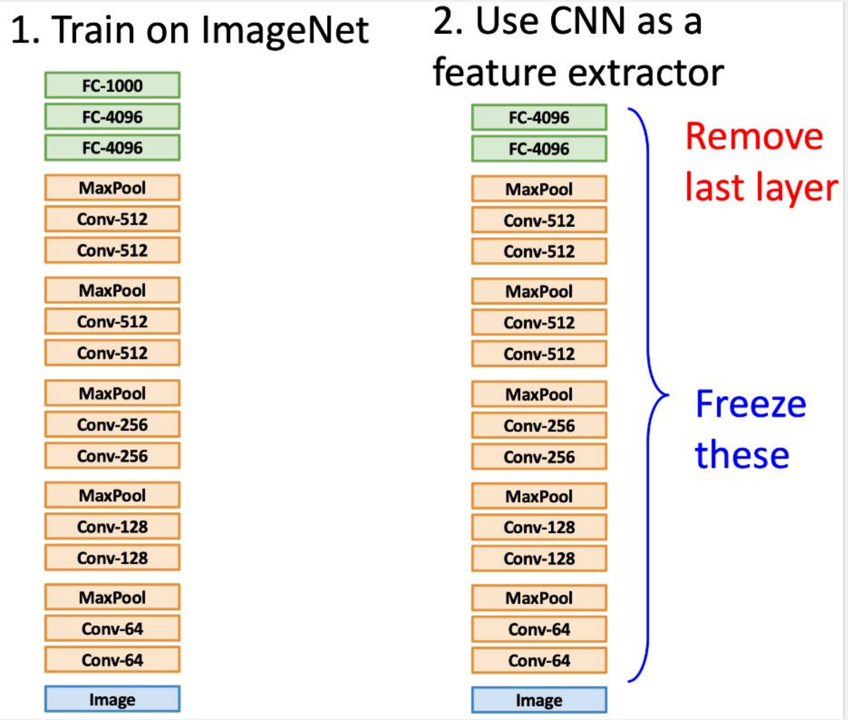


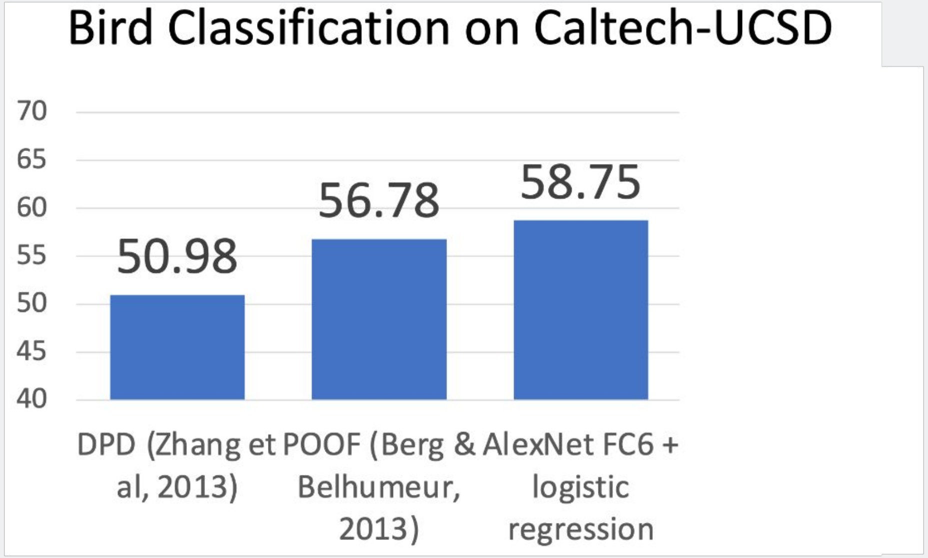
Transfer Learning: Feature Extraction



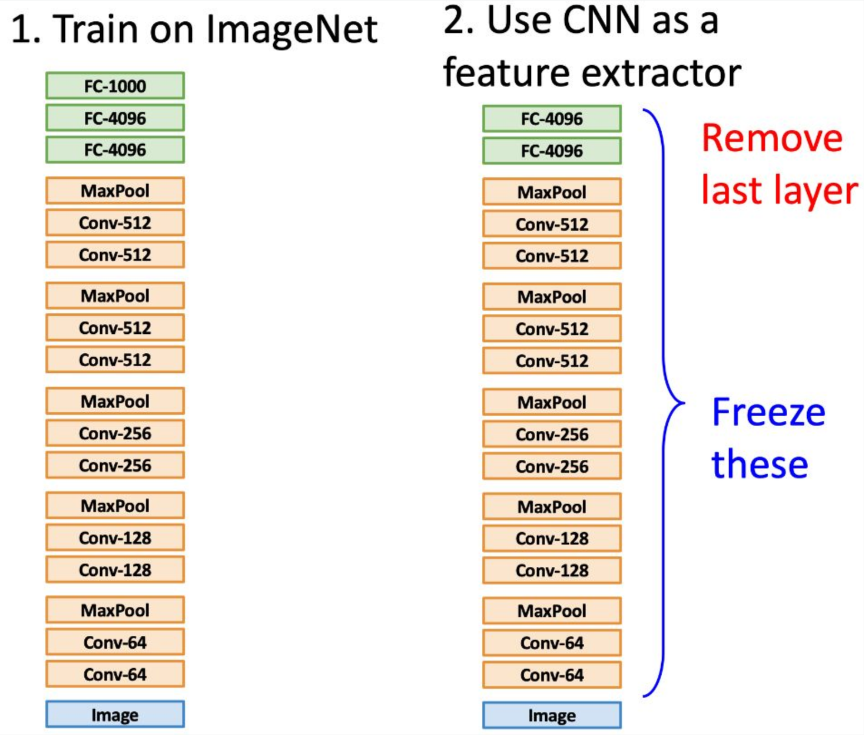


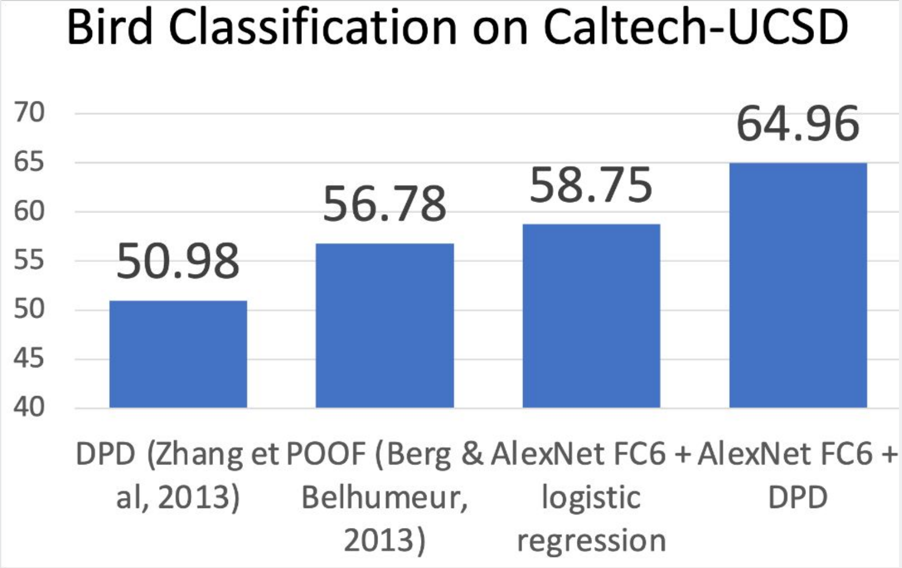
Transfer Learning: Feature Extraction



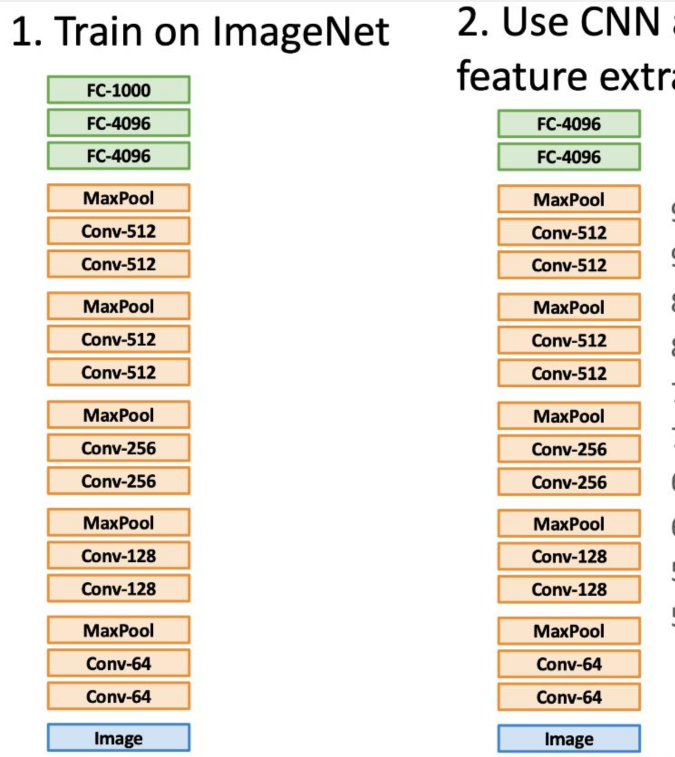


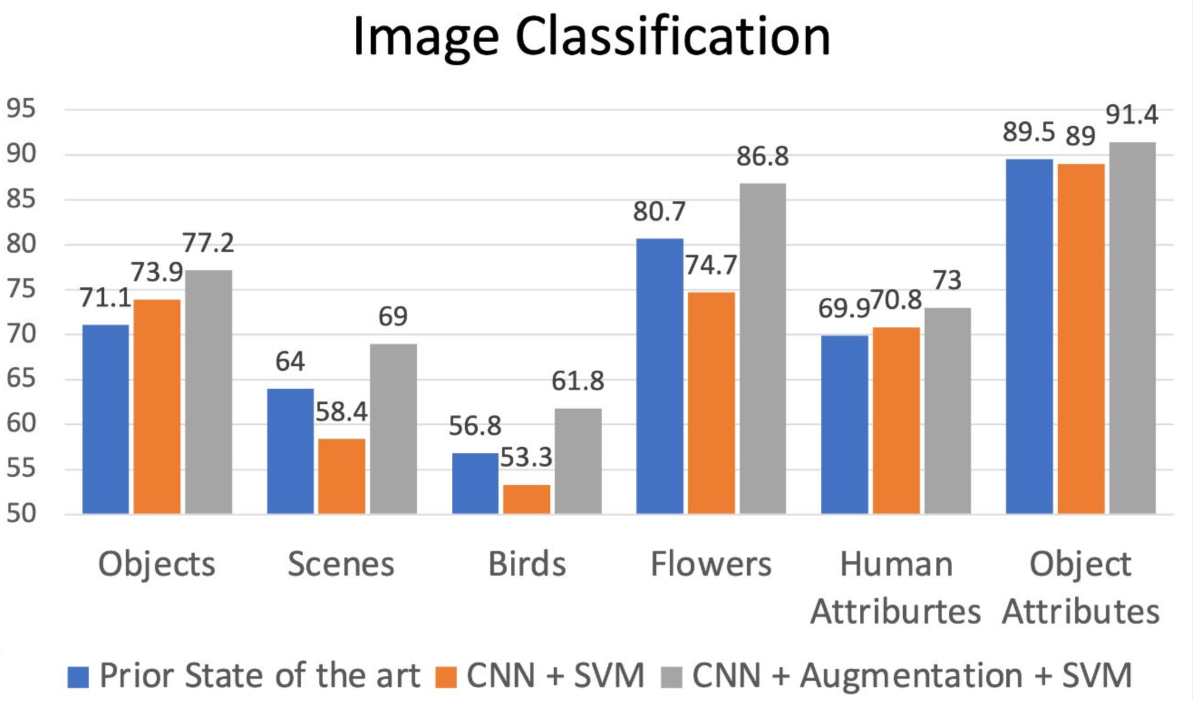
Transfer Learning: Feature Extraction





Transfer Learning: Feature Extraction



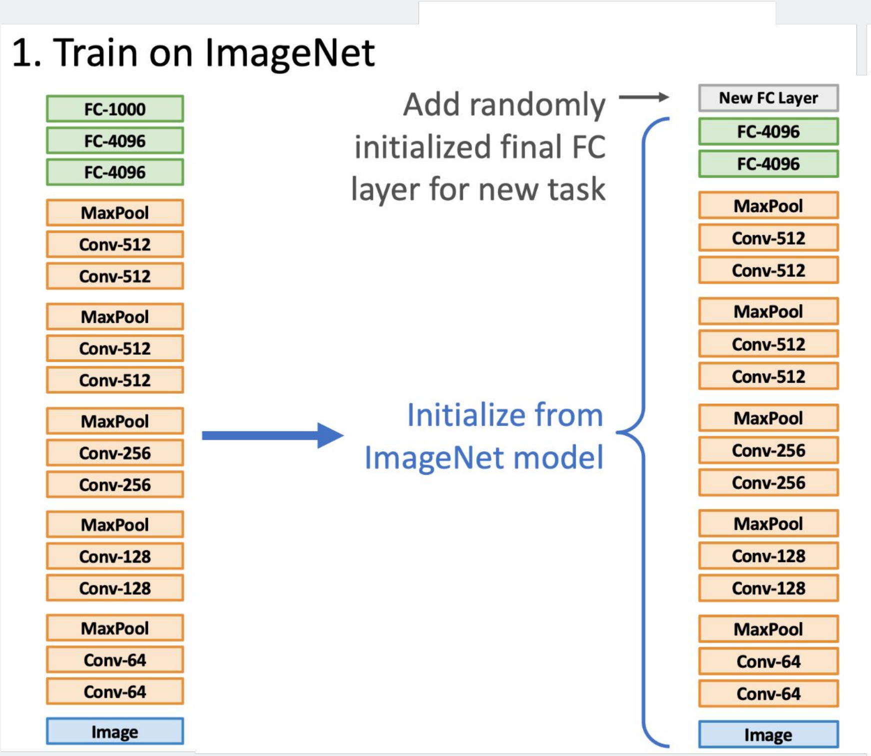


Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014
https://openaccess.thecvf.com/content_cvpr_workshops_2014/W15/papers/Razavian_CNN_Features_Off-the-Shelf_2014_CVPR_paper.pdf

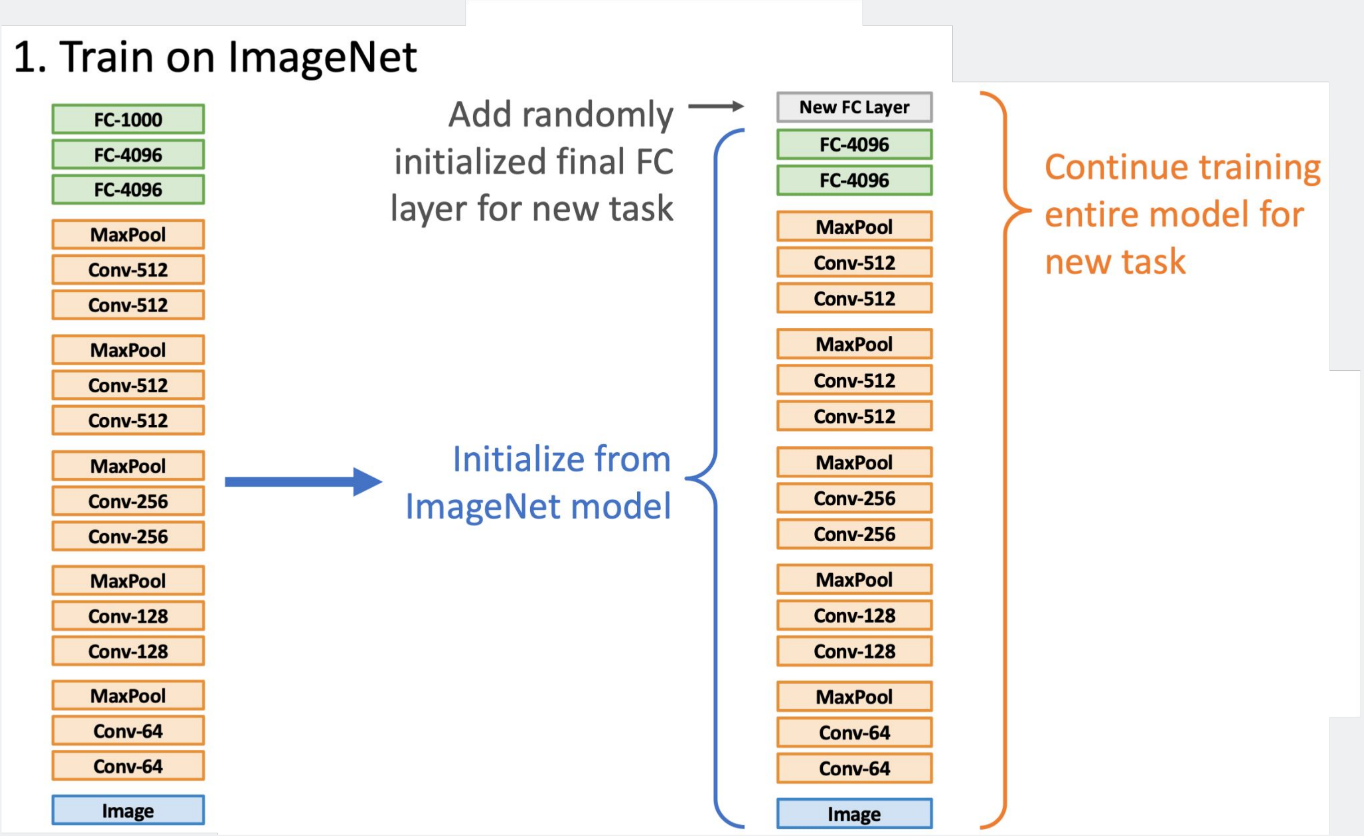
Transfer Learning: Fine Tuning

1. Train on ImageNet

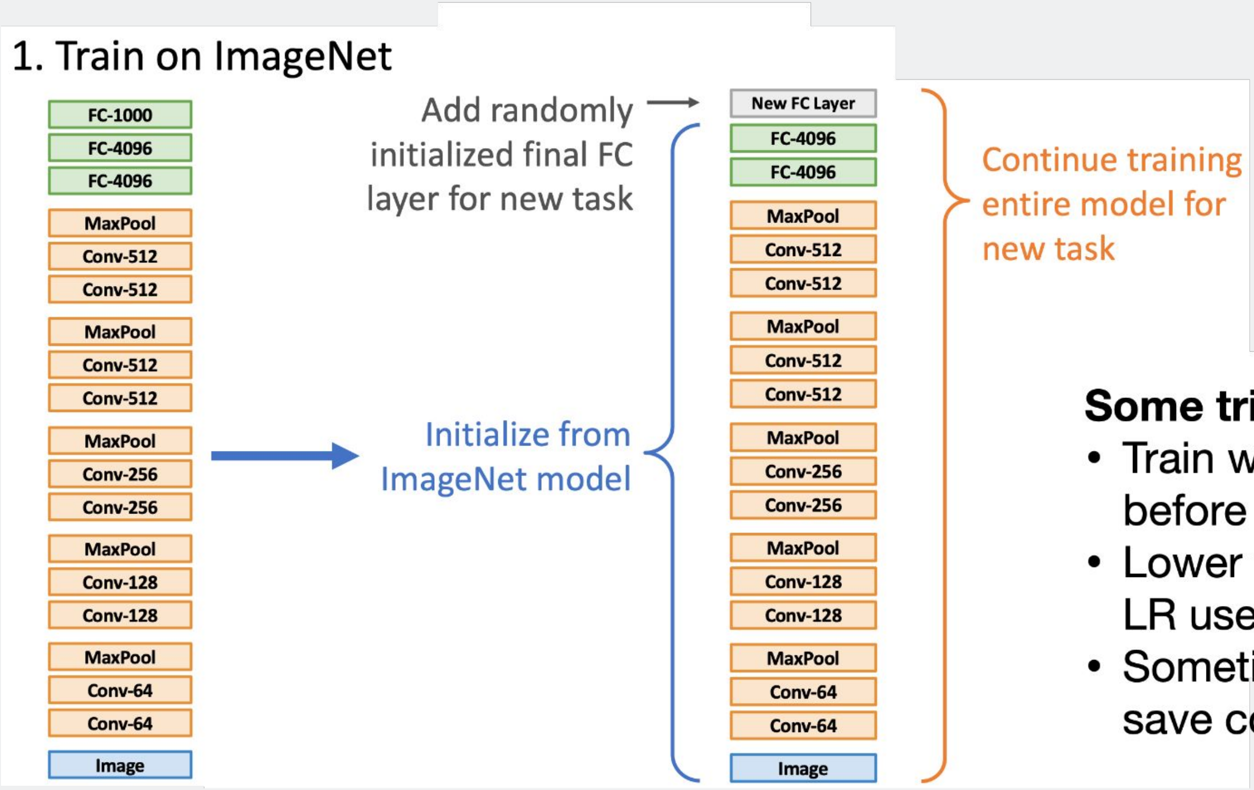
Transfer Learning: Fine Tuning



Transfer Learning: Fine Tuning



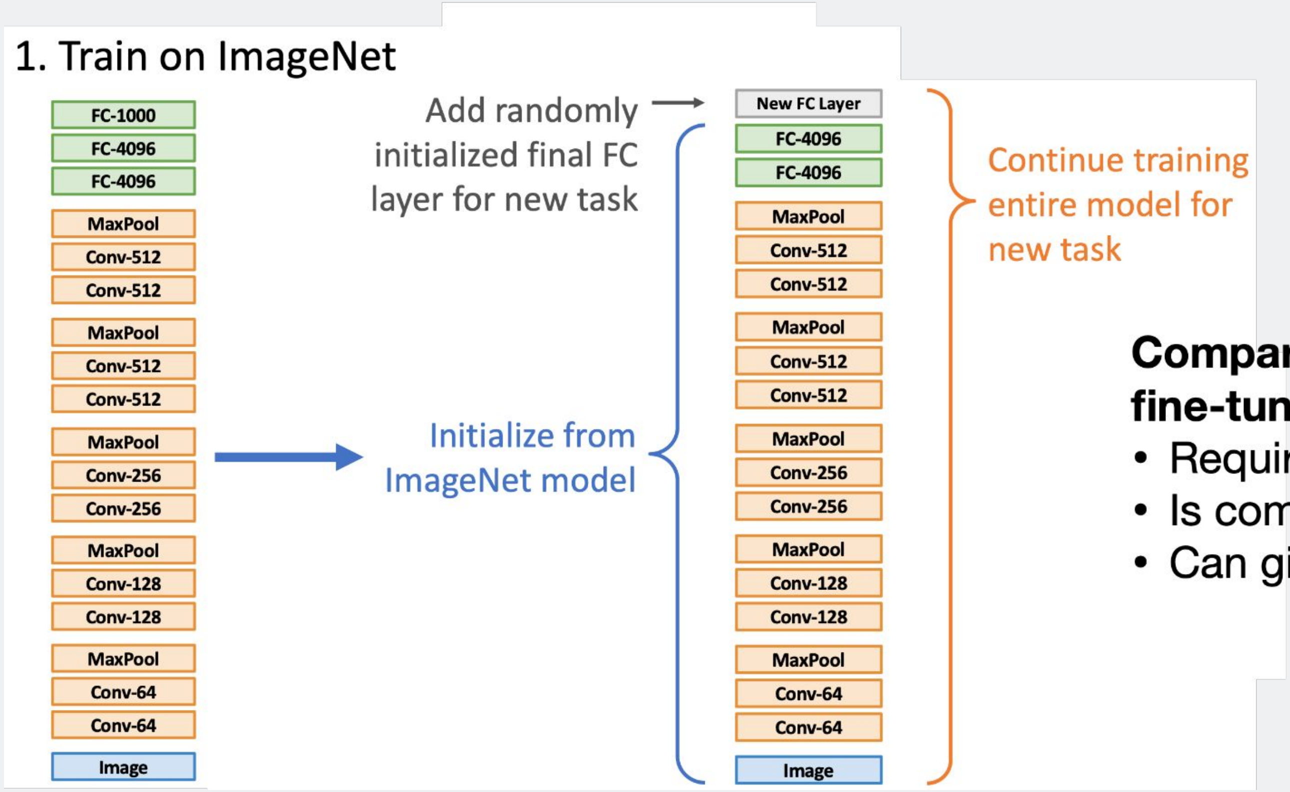
Transfer Learning: Fine Tuning



Some tricks:

- Train with feature extraction first before finetuning
- Lower the learning rate: use ~1/10 of LR used in original training
- Sometimes freeze lower layers to save computation

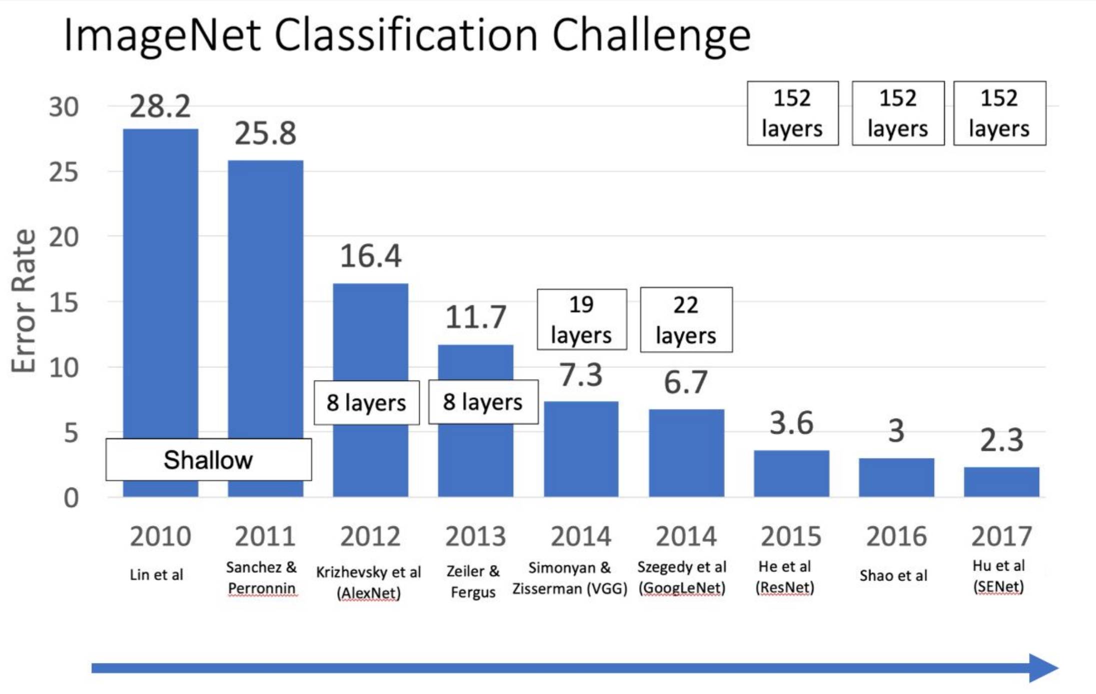
Transfer Learning: Fine Tuning



Compared with feature extraction, fine-tuning:

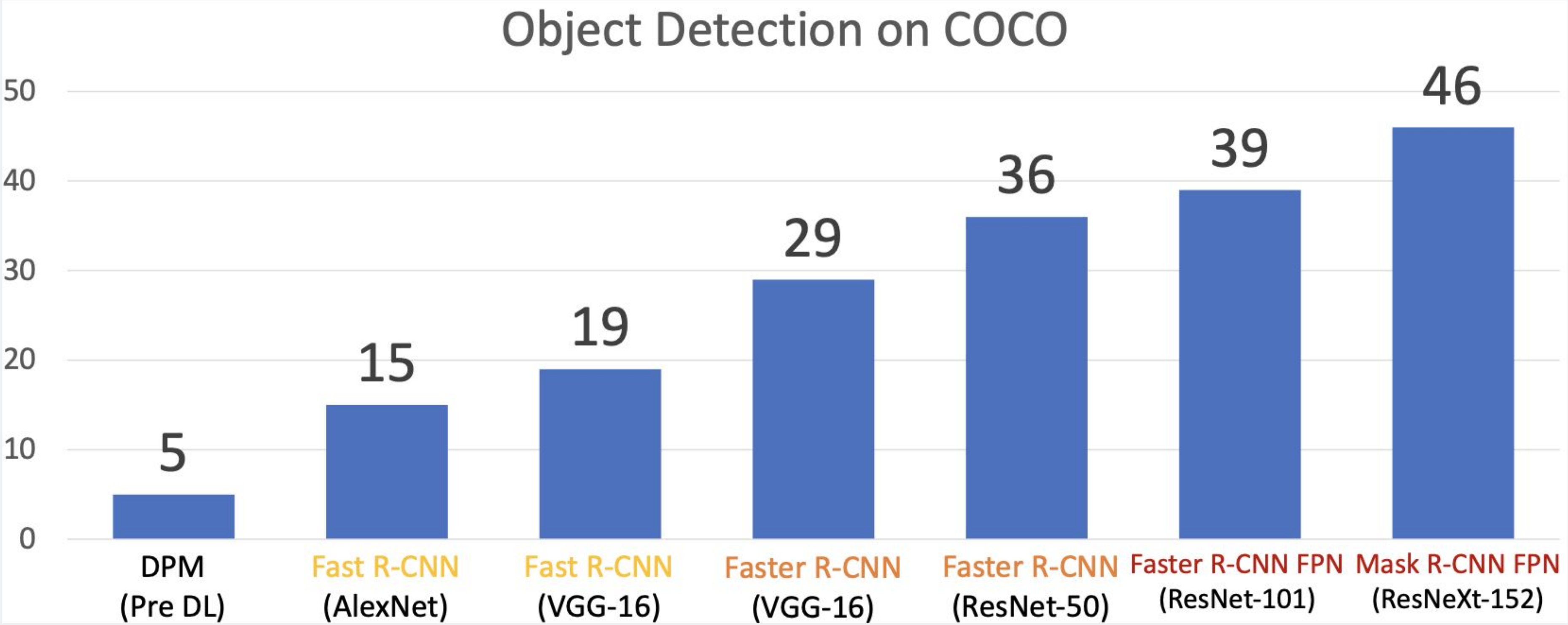
- Requires more data
- Is computationally expensive
- Can give higher accuracies

Transfer Learning: Architecture Matters!



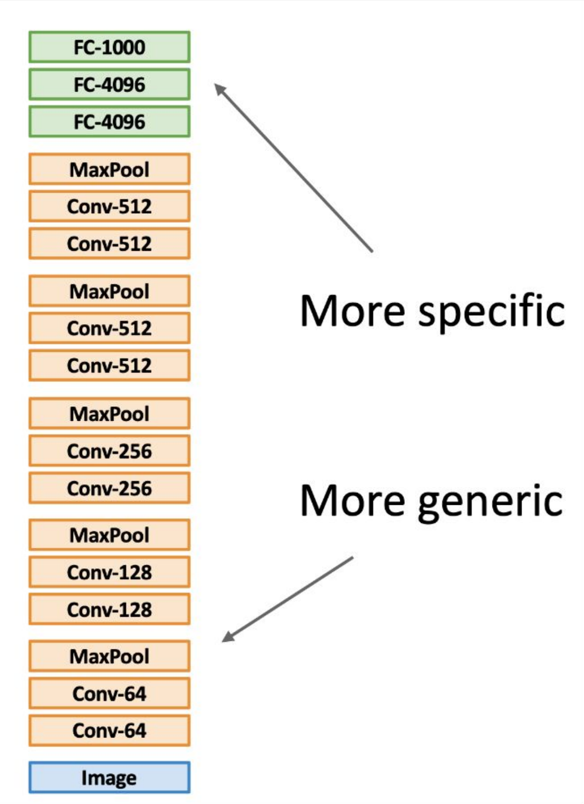
Improvements in CNN architecture leads to improvements in many down stream tasks thanks to transfer learning!

Transfer Learning: Architecture Matters!



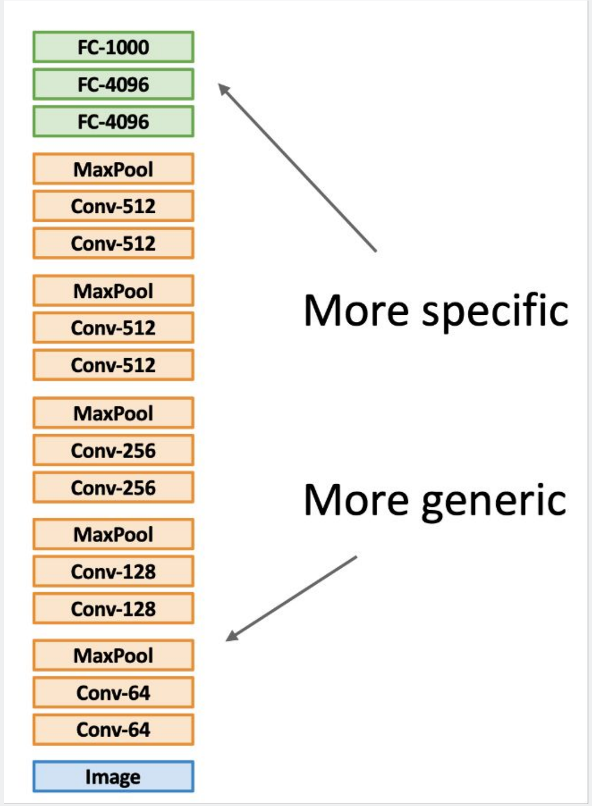
Ross Girshick, “The Generalized R-CNN Framework for Object Detection”, ICCV 2017 Tutorial on Instance-Level Visual Recognition
https://instancetutorial.github.io/

Transfer Learning with CNNs



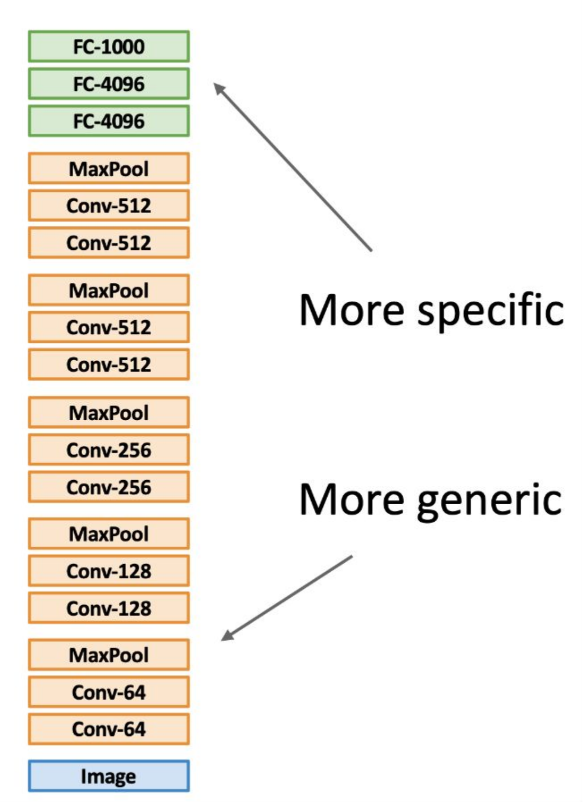
	Dataset similar to ImageNet	Dataset very different from ImageNet
Very little data (10s to 100s)	?	?
Quite a lot of data (100s to 1000s)	?	?

Transfer Learning with CNNs



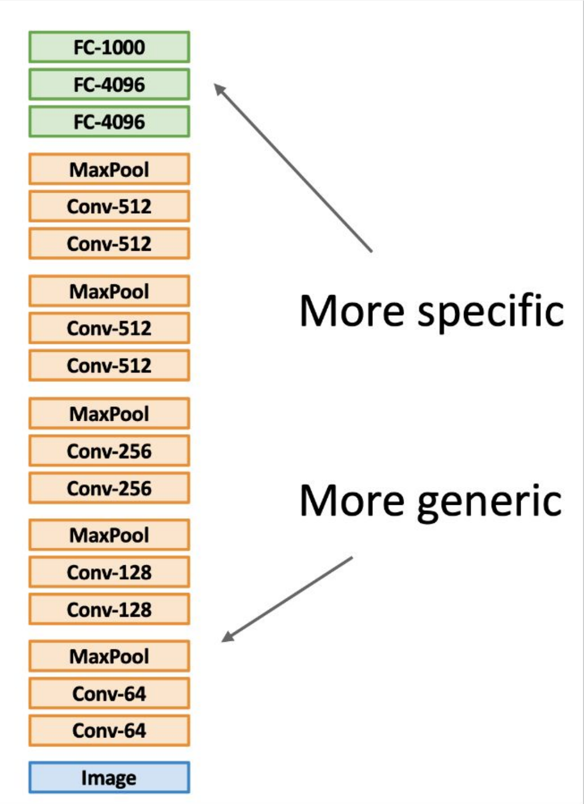
	Dataset similar to ImageNet	Dataset very different from ImageNet
Very little data (10s to 100s)	Use Linear Classifier on top layer	?
Quite a lot of data (100s to 1000s)	?	?

Transfer Learning with CNNs



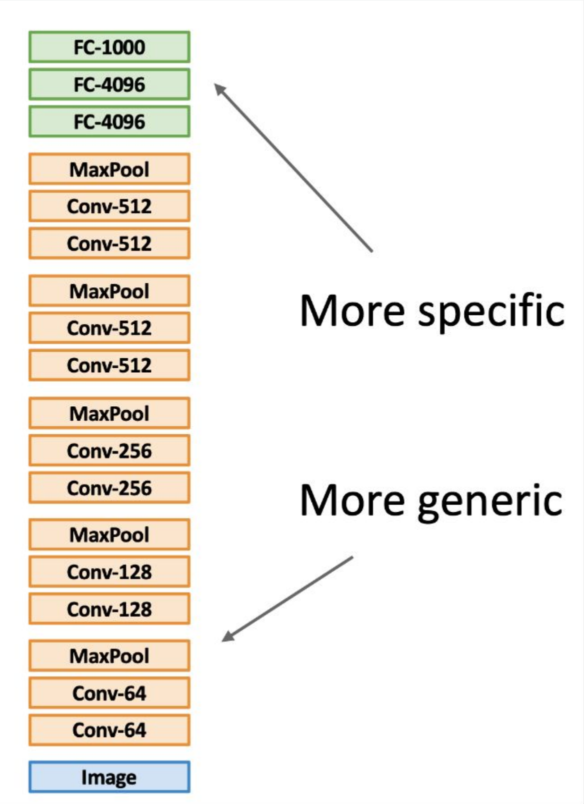
	Dataset similar to ImageNet	Dataset very different from ImageNet
Very little data (10s to 100s)	Use Linear Classifier on top layer	?
Quite a lot of data (100s to 1000s)	Finetune a few layers	?

Transfer Learning with CNNs



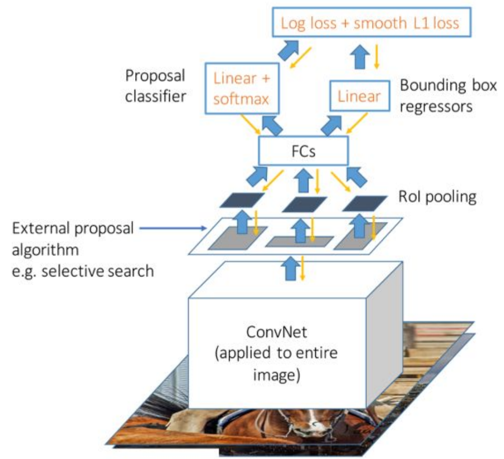
	Dataset similar to ImageNet	Dataset very different from ImageNet
Very little data (10s to 100s)	Use Linear Classifier on top layer	?
Quite a lot of data (100s to 1000s)	Finetune a few layers	Finetune a larger number of layers

Transfer Learning with CNNs

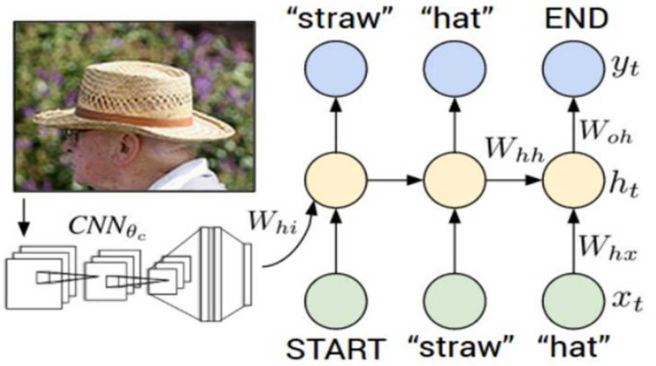


	Dataset similar to ImageNet	Dataset very different from ImageNet
Very little data (10s to 100s)	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
Quite a lot of data (100s to 1000s)	Finetune a few layers	Finetune a larger number of layers

Transfer Learning - “the norm”, not the exception

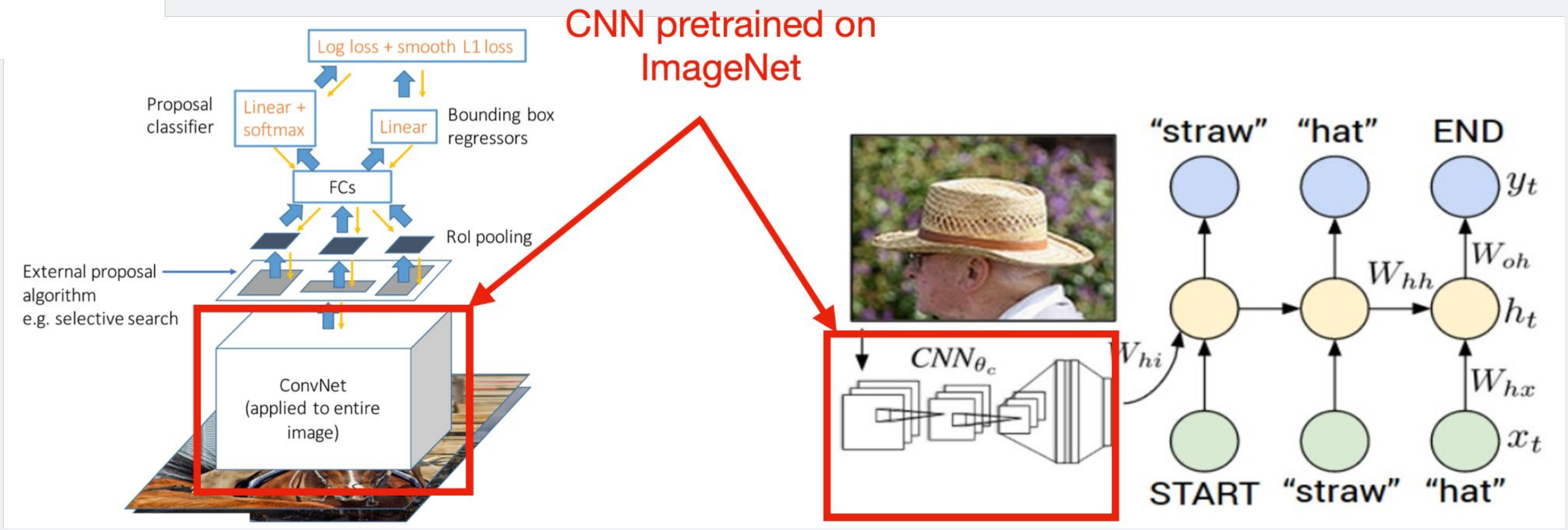


Girshick, “Fast R-CNN”, ICCV 2015

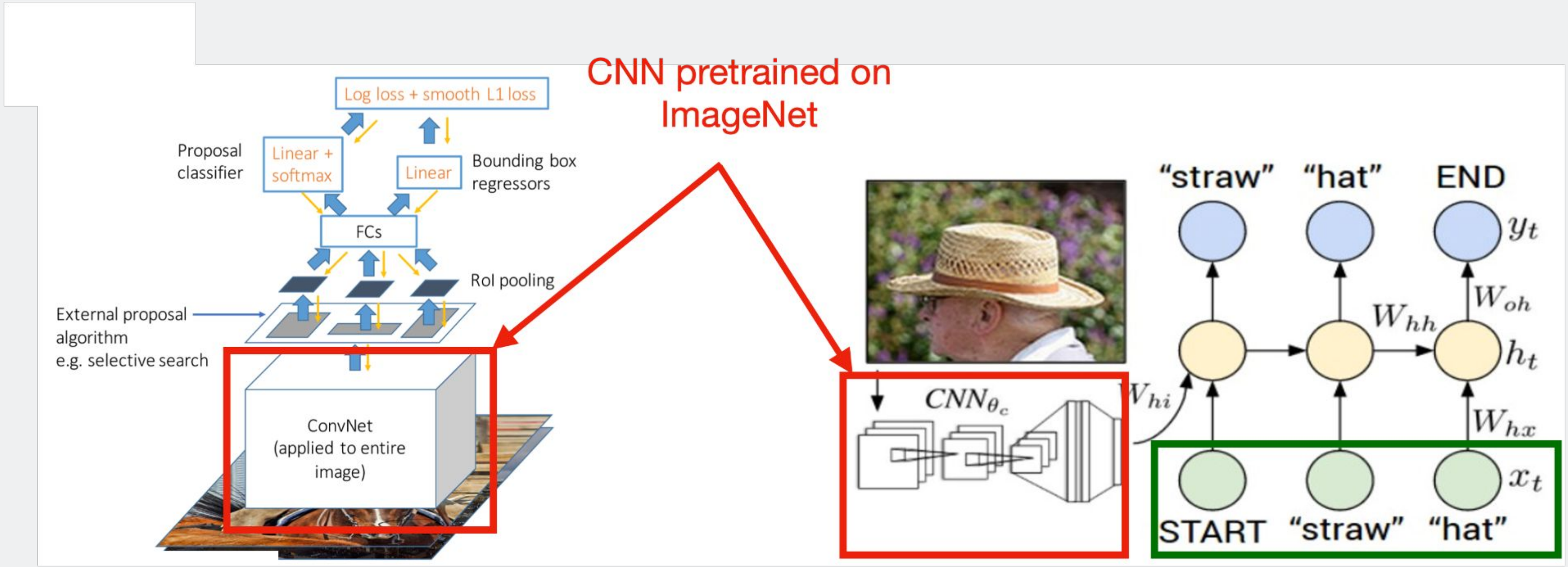


Karpathy and Fei-Fei, “Deep Visual-Semantic Alignments for Generating Image Descriptions”, CVPR 2015

Transfer Learning - "the norm", not the exception

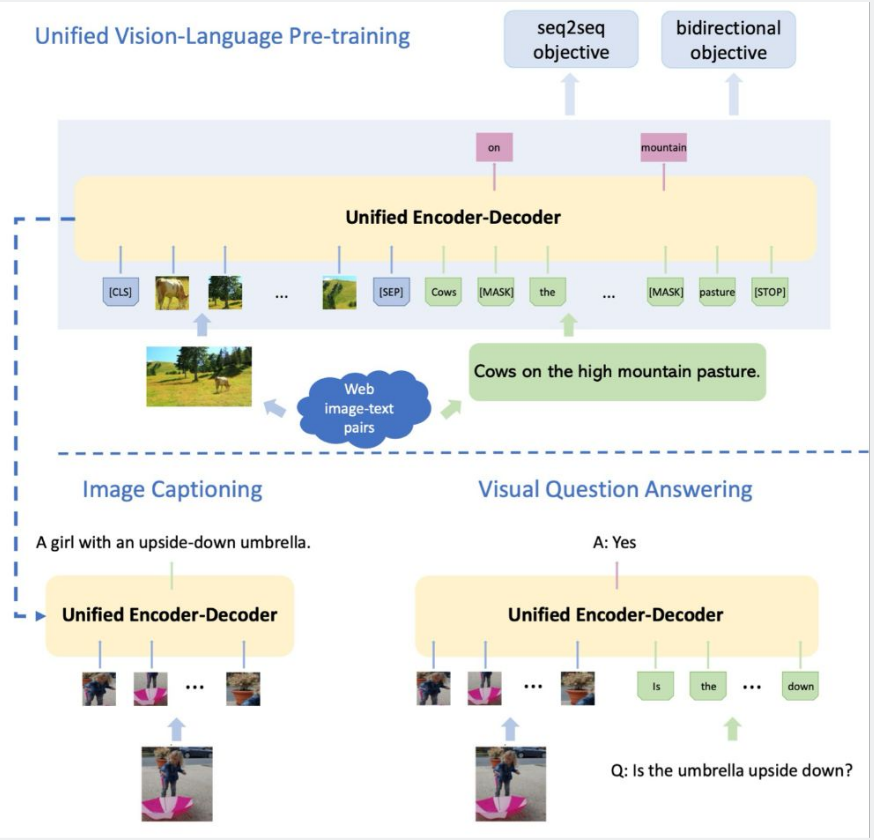


Transfer Learning - “the norm”, not the exception



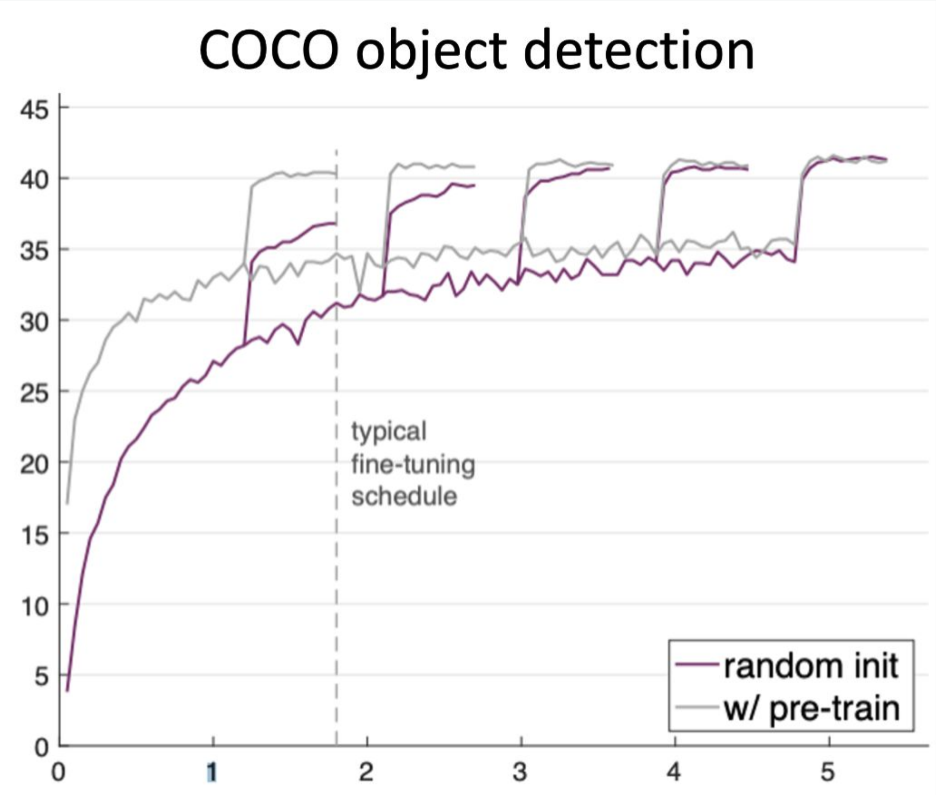
Word vectors pretrained with word2vec

Transfer Learning - "the norm", not the exception



1. Train CNN on ImageNet
2. Fine-Tune (1) for object detection on Visual Genome
3. Train BERT language model on lots of text
4. Combine (2) and (3), train for joint image / language modeling
5. Fine-tune (5) for image captioning, visual question answering, etc.

Transfer Learning: Helps you converge faster!



If you have enough data and train for much longer, random initialization can sometimes do as well as transfer learning

He et al, "Rethinking ImageNet Pre-Training", ICCV 2019
https://openaccess.thecvf.com/content_ICCV_2019/papers/He_Rethinking_ImageNet_Pre-Training_ICCV_2019_paper.pdf

Transfer Learning: Helps you converge faster!

Very active area of research!

Summary

1. One time setup: **Today**
   - Activation functions, data preprocessing, weight initialization, regularization
2. Training dynamics: **Next time**
   - Learning rate schedules; large-batch training; hyperparameter optimization
3. After training:
   - Model ensembles, transfer learning