Transfer Learning for Image Recognition in Healthcare Industry

Michał Kierzynka, 12 December 2019, PyData Warsaw

Agenda

01 What is transfer learning?

02 What is the challenge in healthcare?

03 Design of experiments

04 Results

05 Conclusions

About me

Michał Kierzynka

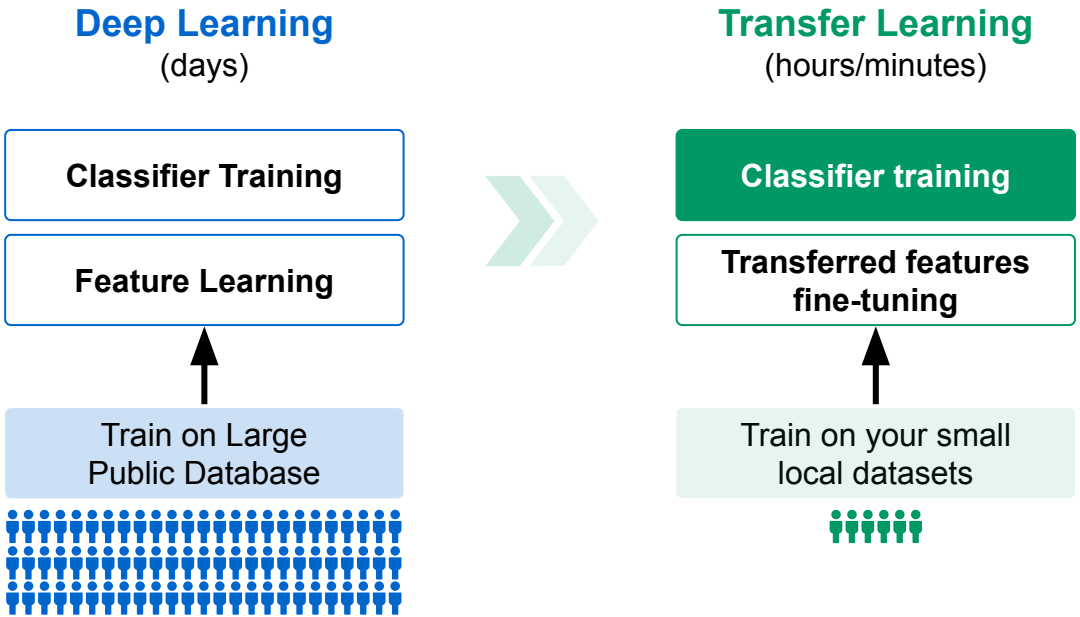
- Background: Computer Science, PhD
- Parallel Computing & HPC
- Bioinformatics
- Deep Learning

What is transfer learning?

01

What is transfer learning?

Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned.



Quote from: Chapter 11: Transfer Learning, Handbook of Research on Machine Learning Applications, 2009

Benefits of transfer learning

Benefits:

- shorter learning time
- better model performance (accuracy etc.)
- less training data needed
- better model generalization
  (less overfitted models)

Transfer learning from **natural image datasets**, particularly ImageNet, has become a de-facto **standard in healthcare imaging**.

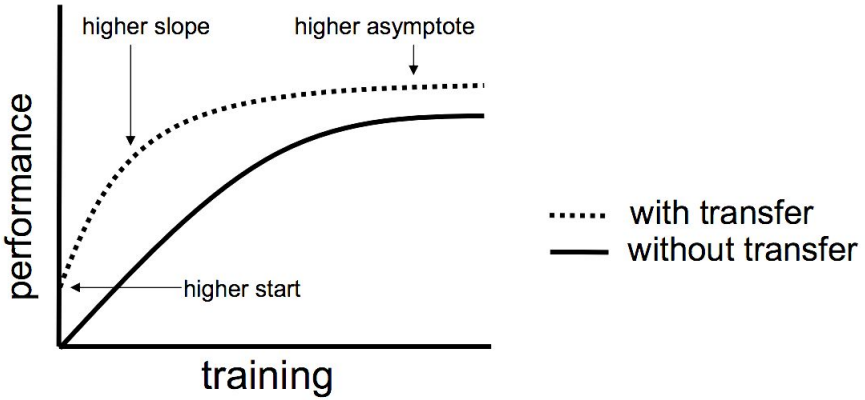


Image from:Transfer Learning, Handbook of Research on Machine Learning Applications, Chapter 11

*This process will tend to work **if the features are general**, meaning suitable to both base and target tasks, instead of specific to the base task.*

Quote from: How transferable are features in deep neural networks?, J. Yosinski, J. Clune, Y. Bengio, H. Lipson, NIPS 2014

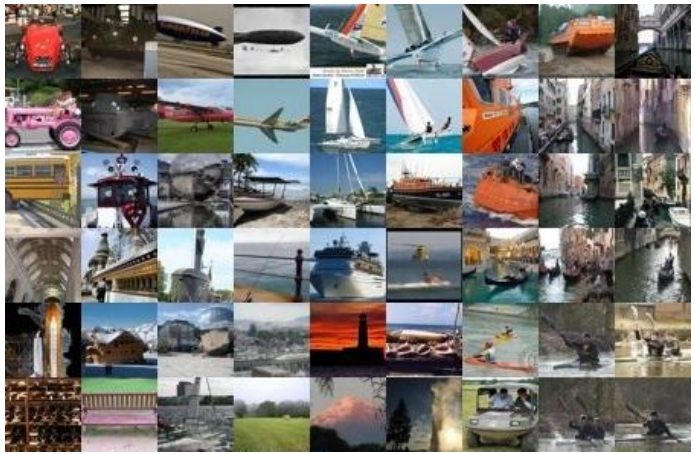
What is the challenge in healthcare?

02

What is the challenge in healthcare?

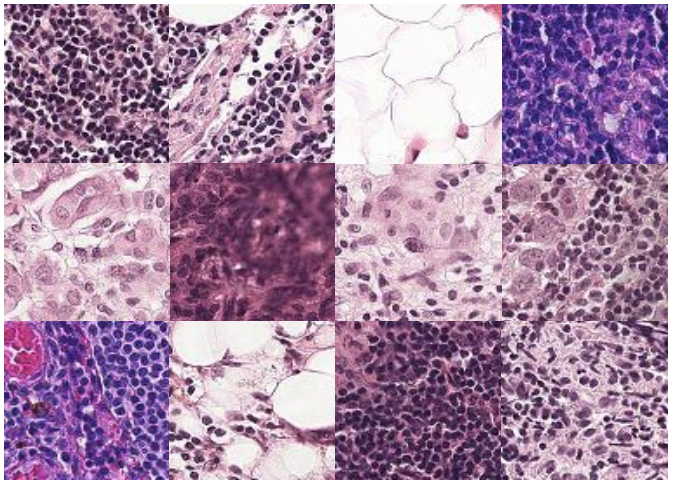
Publicly available pretrained models may not work well in healthcare industry.

Features useful for **natural images**



vs

...may not be that useful for more **specific images**.



Can we benefit from models pretrained on **natural images**
to boost the performance in **healthcare specific** models?

Design of experiments

03

Design of Experiments

What?

- **Task selection:** image classification
  - relatively easy to find public datasets for evaluation
  - more data sets -> better results generalization
  - generic: the same CNN architectures for feature extraction used in many other CV tasks
  - many new papers on the subject published recently are based on image classification

How?

- **Compare network initialization:** evaluate the performance of selected CNNs initialized:
  - randomly (`glorot_uniform`)
  - with pretrained weights
- **see what pretrained features can be reused** by freezing selected layers and fine-tuning the others
- pretrained weights - **ImageNet**

Design of Experiments - Networks

Network architectures selected for evaluation:

- architectures proven to work well on ImageNet (both large and small)
- custom CNN architecture for a reference

Model	Top-1 Accuracy	Top-5 Accuracy	Parameters
Inception v3	0.779	0.937	23,851,784
ResNet 50 v2	0.760	0.930	25,613,800
DenseNet 121	0.750	0.923	8,062,504
MobileNet	0.704	0.895	4,253,864
Small CNN	-	-	2,114,496

Design of Experiments - Networks

Small CNN:

1. Conv2D ▶ ReLu ▶ BatchNorm ▶ MaxPool
2. Conv2D ▶ ReLu ▶ BatchNorm ▶ MaxPool
3. Conv2D ▶ ReLu ▶ BatchNorm ▶ MaxPool
4. Conv2D ▶ ReLu ▶ BatchNorm ▶ MaxPool
5. GlobalAveragePooling2D
6. Dense ▶ ReLu ▶ Dropout ▶ Dense ▶ Softmax

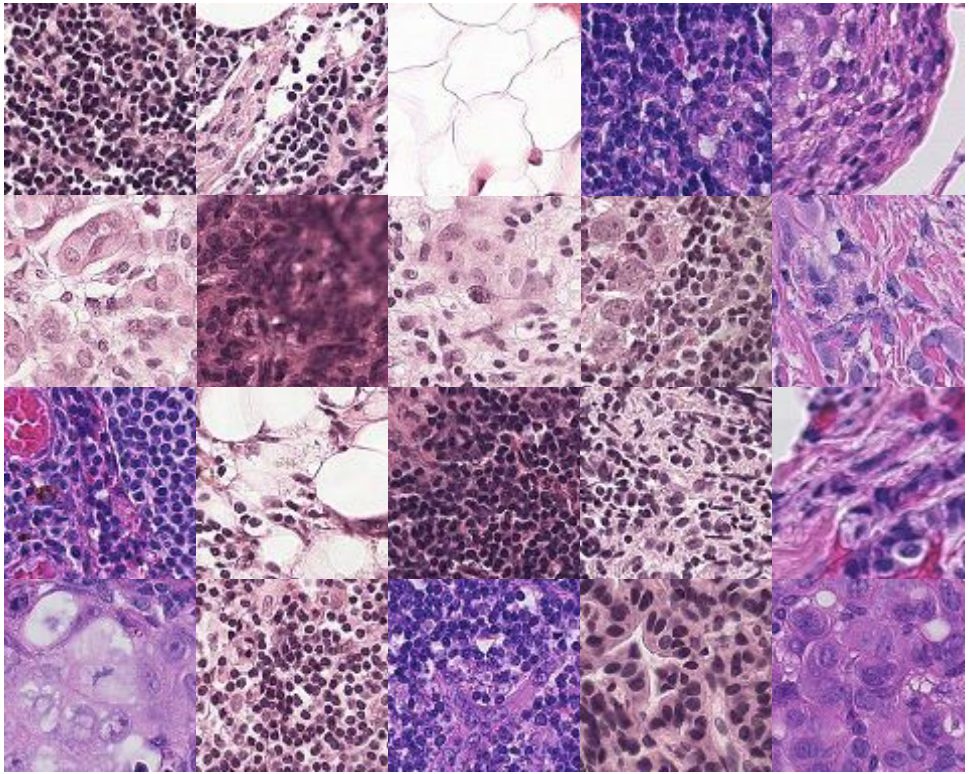
Conv2D kernel size: 7x7

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	4736
batch_normalization (BatchNo	(None, 128, 128, 32)	128
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_1 (Conv2D)	(None, 64, 64, 64)	100416
batch_normalization_1 (Batch	(None, 64, 64, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 32, 32, 64)	0
conv2d_2 (Conv2D)	(None, 32, 32, 128)	401536
batch_normalization_2 (Batch	(None, 32, 32, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 16, 16, 128)	0
conv2d_3 (Conv2D)	(None, 16, 16, 256)	1605888
batch_normalization_3 (Batch	(None, 16, 16, 256)	1024
max_pooling2d_3 (MaxPooling2	(None, 8, 8, 256)	0
global_average_pooling2d (Gl	(None, 256)	0

Design of Experiments - Data Sets

Cancer

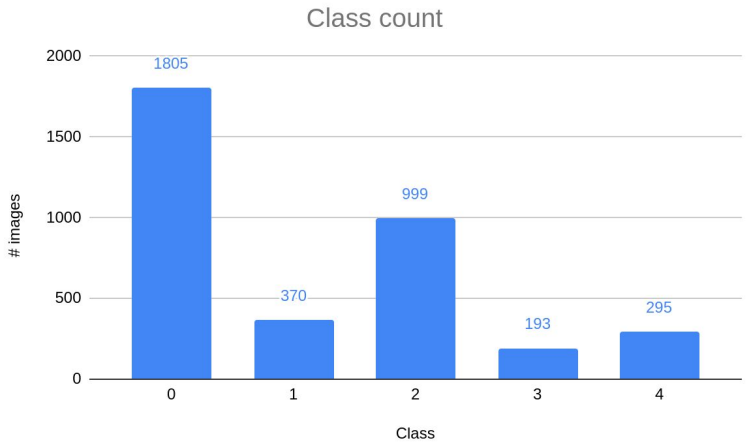
- histopathologic cancer detection
- 220 025 images
- 96x96x3
- binary classification
- license: Creative Commons Zero v1.0 Universal

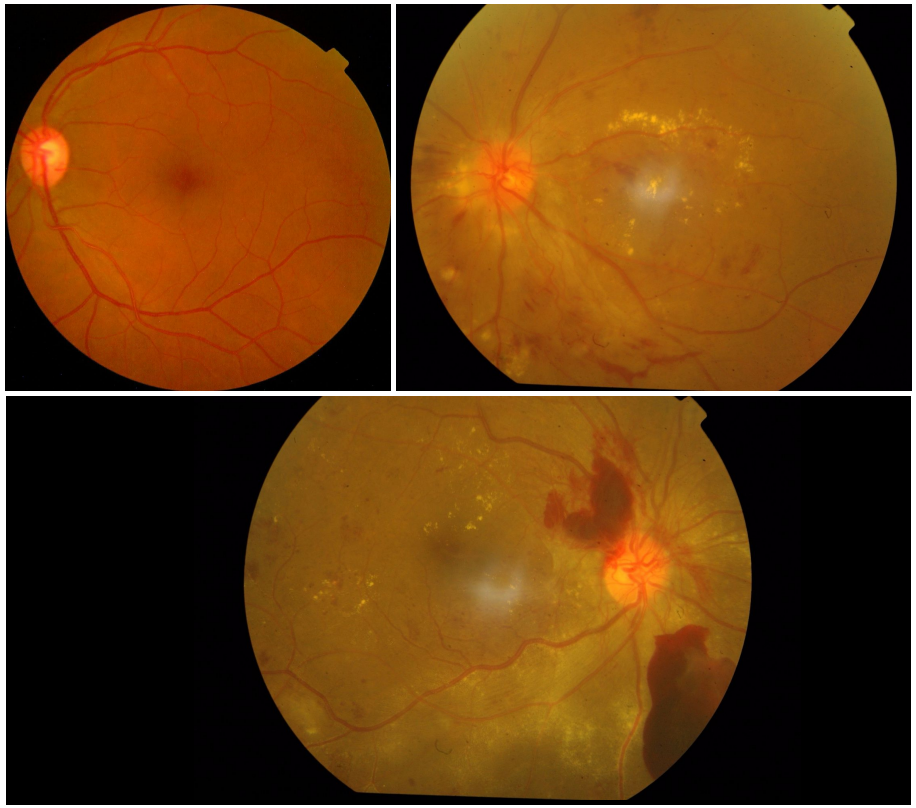


Design of Experiments - Data Sets

Retina

- diabetic retinopathy - the leading cause of blindness
- 3 662 images
- large images of various size, e.g. 1504x1000x3, 3216x2136x3
- 5 classes
- source: Kaggle

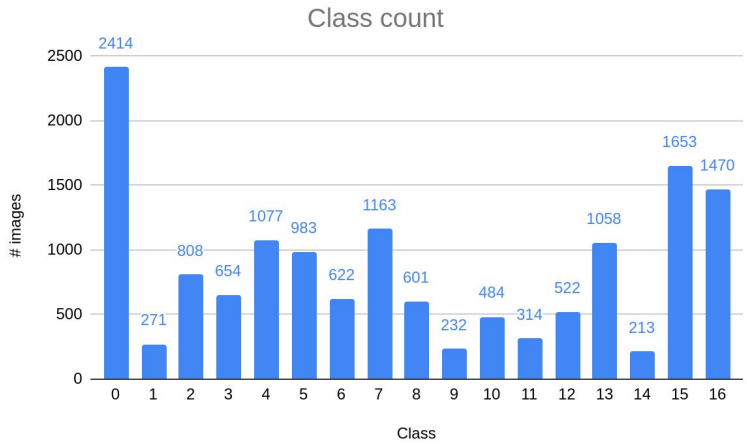


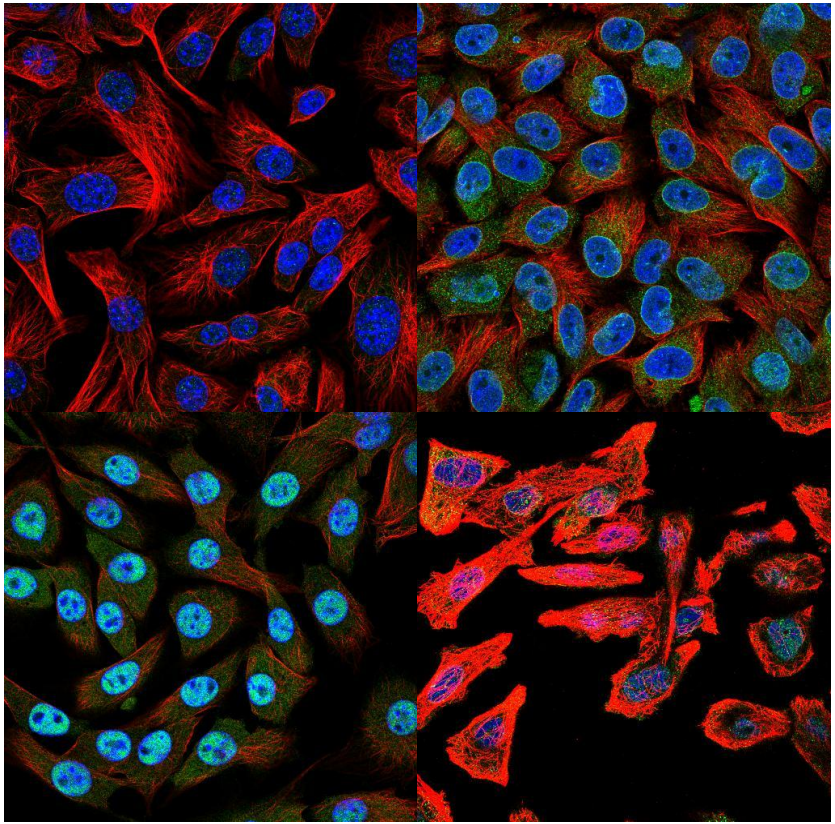


Design of Experiments - Data Sets

Proteins

- Human Protein Atlas - patterns of proteins in microscope images
- 14 539 images
- 512x512x3
- 17 classes
- source: Kaggle, open access data

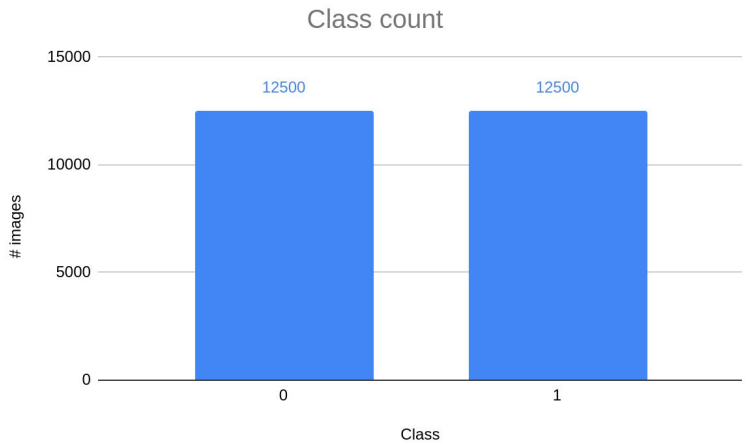




Design of Experiments - Data Sets

Cats vs dogs

- **toy data set** with natural images
- 25 000 images
- various resolutions
- binary classification
- source: Kaggle



Design of Experiments - Data Sets

Data set name	# classes	# images	resolution	class balance
Cancer	2	220 025	96x96x3	yes
Retina	5	3 662	various (large)	no
Proteins	17	14 539	512x512x3	no
Cats vs dogs	2	25 000	various (medium)	yes

- to account for class imbalance the default metric will be **AUC-ROC curve** (micro-average)
- train:validation:test split 8:1:1, stratified

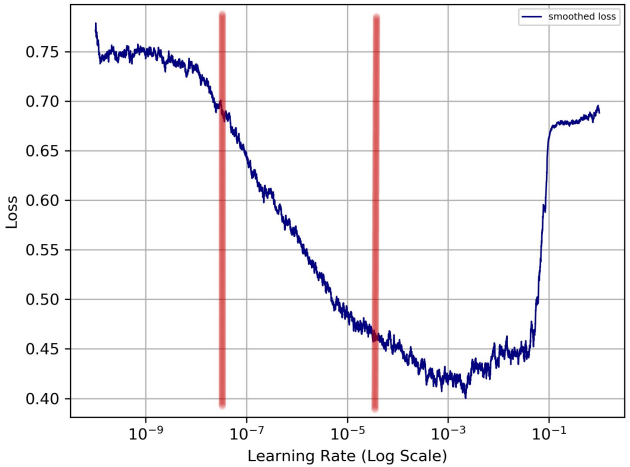
Design of Experiments - Training Regime

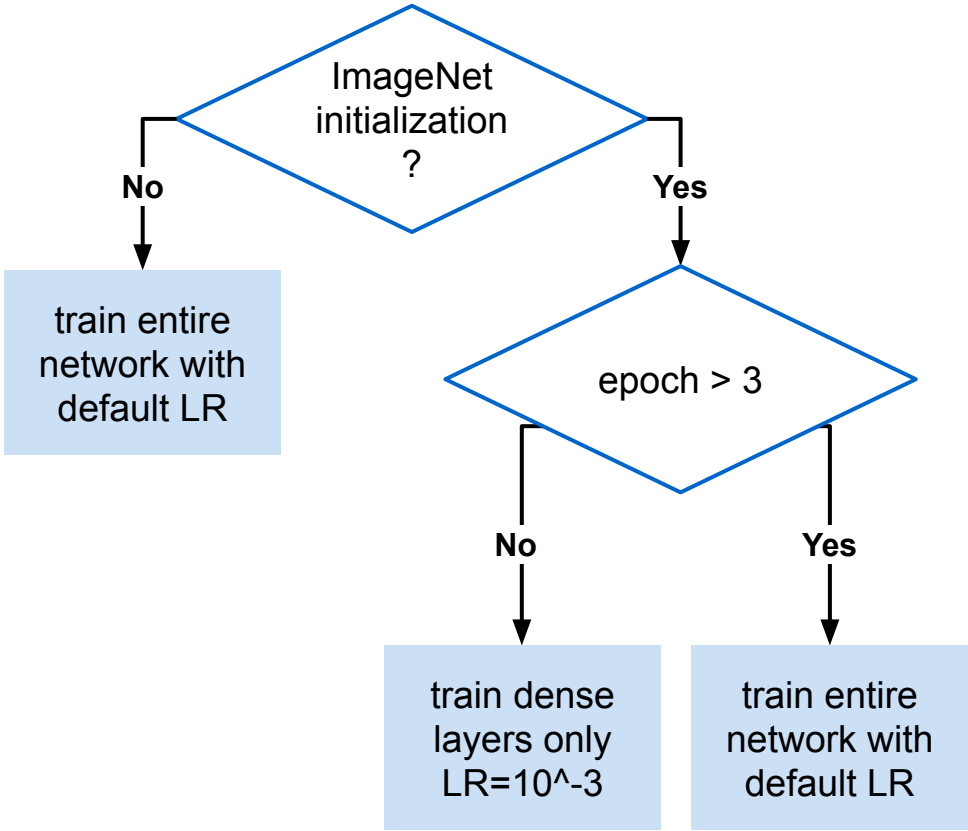
- **image preprocessing**
  - resize to default network resolution
  - scale pixel values to the range 0-1
- **image augmentation**
  - random horizontal and vertical flips
  - gaussian noise
  - distort image color (brightness, saturation, hue, contrast)
- **optimizer: Adam**
- **number of epochs: 3 + 50**
- **categorical cross entropy loss**
- **no hyperparameter tuning (besides learning rate)**

Design of Experiments - Training Regime

Default **learning rates** (LR):

- Cancer: 10^-5
- Retina: 10^-4
- Proteins: 10^-4
- Cats vs dogs: 10^-5





Design of Experiments - Technology Stack

- HPC cluster
- 4 days of computations on 16x Tesla V100
- ~2 months on 1x Tesla V100

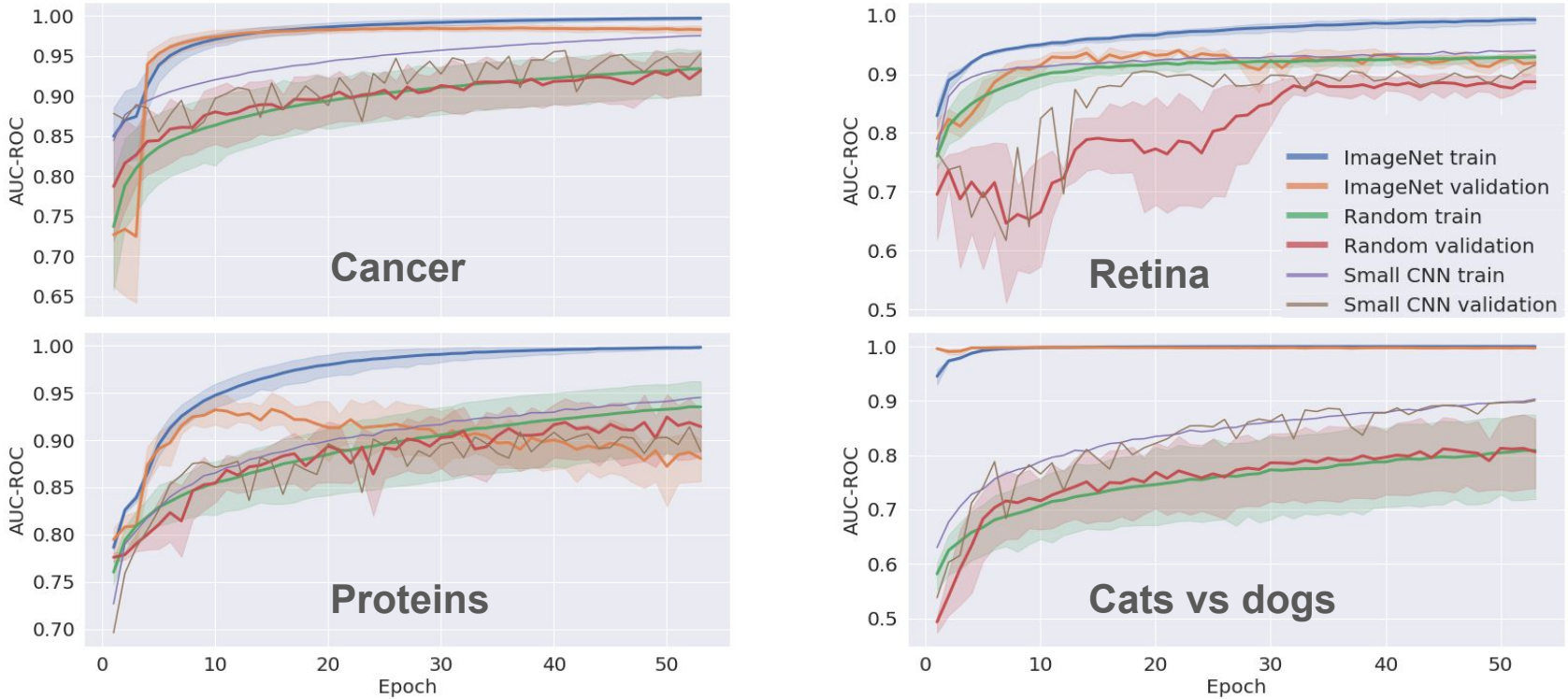
Results

04

Random vs ImageNet initialization

Training the entire network with the following initialization:

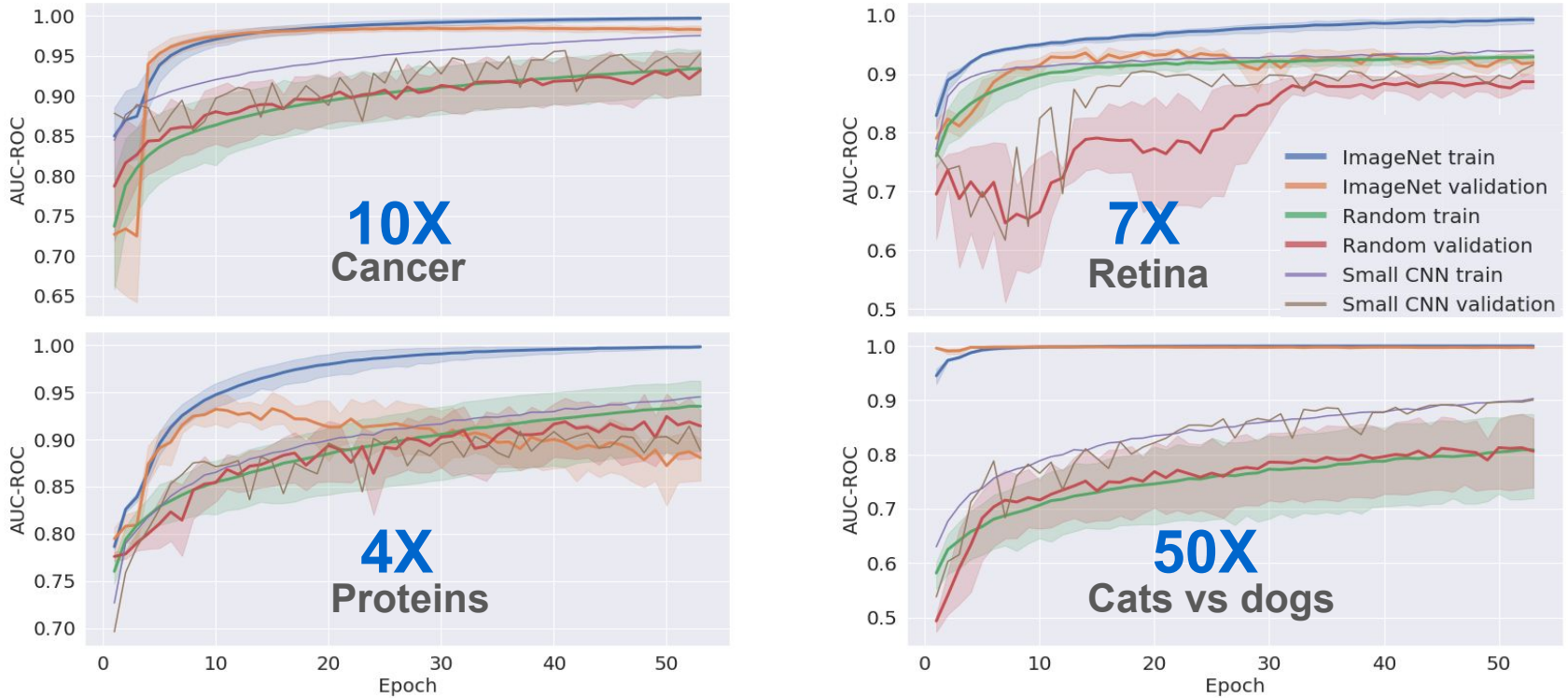
Random vs ImageNet initialization



Trend lines and confidence intervals calculated for the following networks:

Resnet 50 v2, Inception v3, DenseNet 121, MobileNet

Random vs ImageNet initialization - speedup



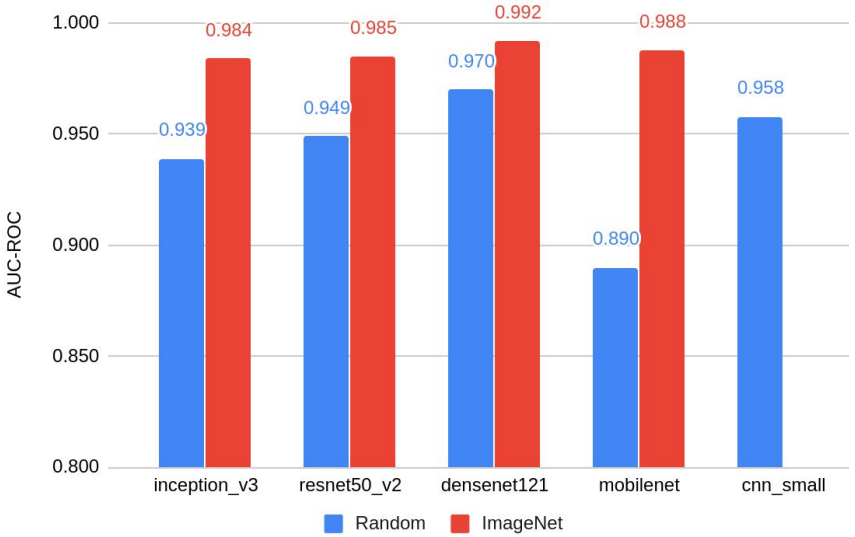
Trend lines and confidence intervals calculated for the following networks:

Resnet 50 v2, Inception v3, DenseNet 121, MobileNet

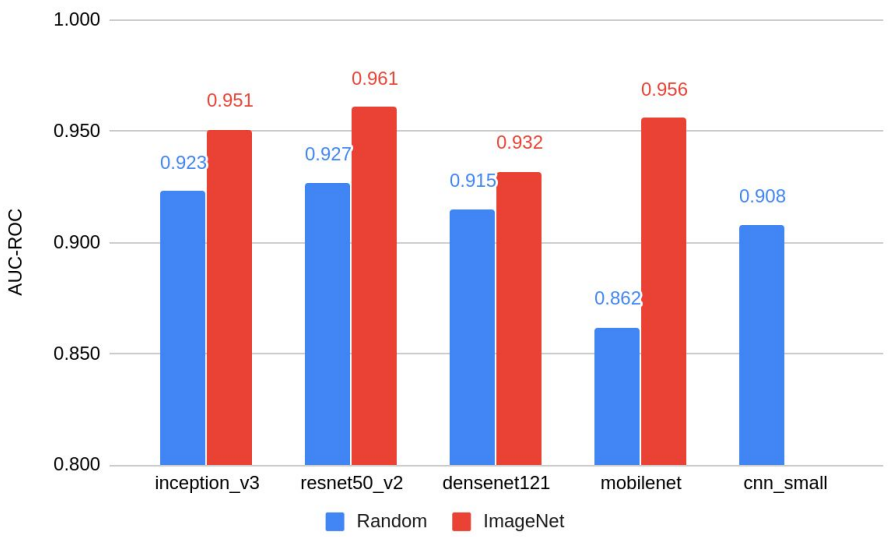
Random vs ImageNet initialization

Test set AUC-ROC (best checkpoint)

Cancer



Retina

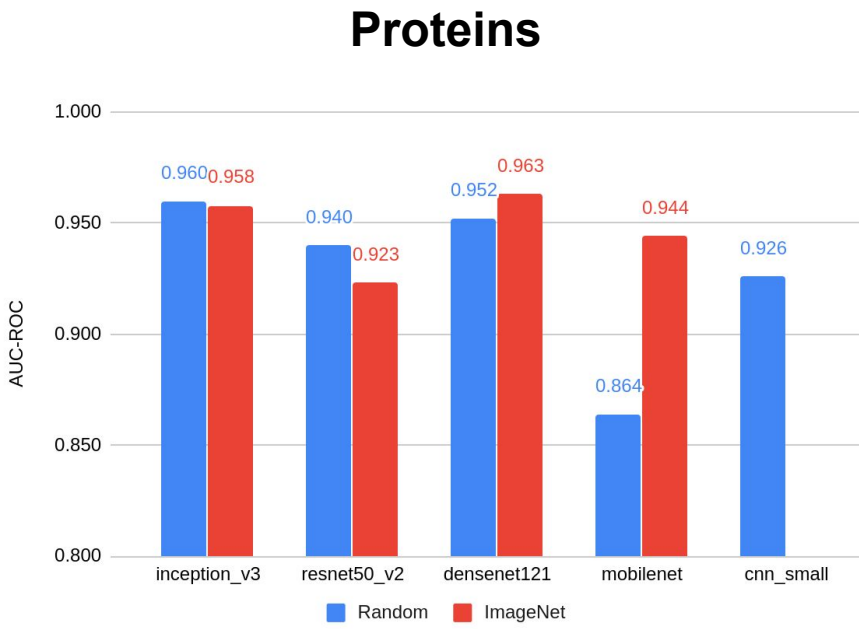


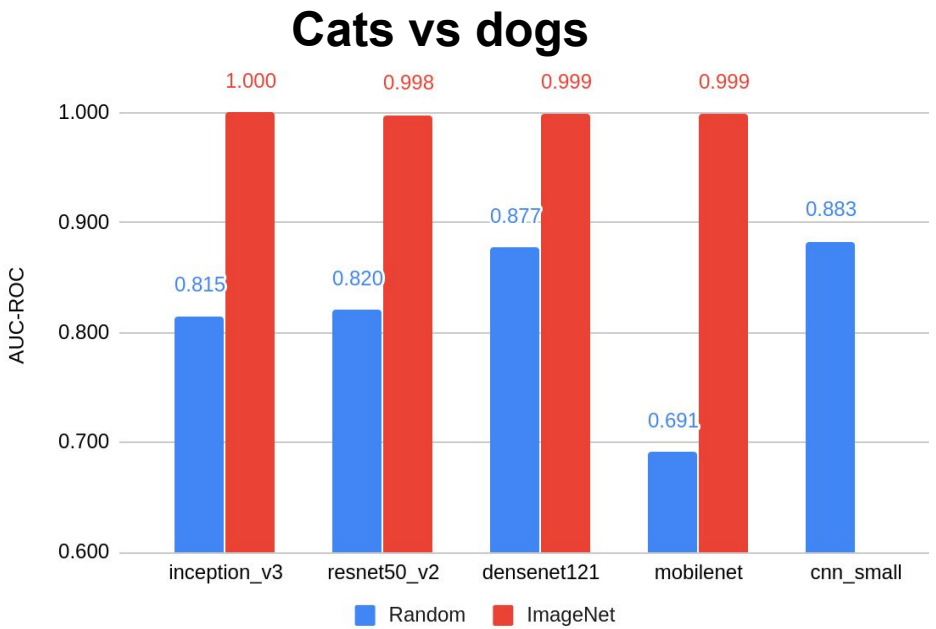
Average improvement:

5.5% **4.9%**

Random vs ImageNet initialization

Test set AUC-ROC *(best checkpoint)*





Average improvement:

2.1% **25.7%**

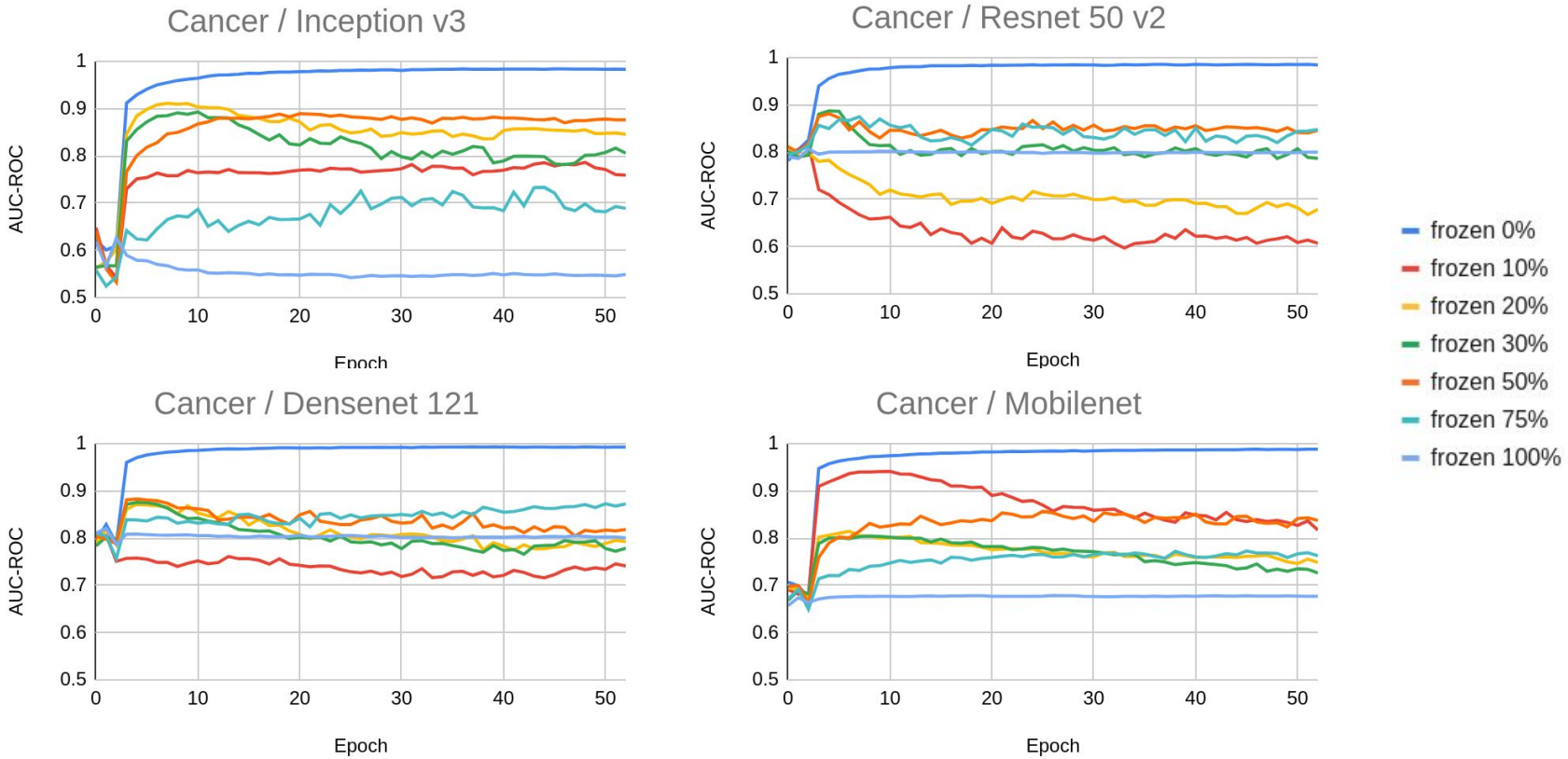
Feature Reuse - freezing pretrained layers

- initializing with pretrained weights improves the speed of training
  - how much of this speedup comes from reusing the features?
- see **what pretrained features can be reused** by freezing selected layers and fine-tuning the others
- the following percentage of bottom* layers were freezed:
  - 100% (effectively fixed-feature extractor)
  - 75%
  - 50%
  - 30%
  - 20%
  - 10%
  - 0% (all network is fine-tuned)

* layers closer to network input

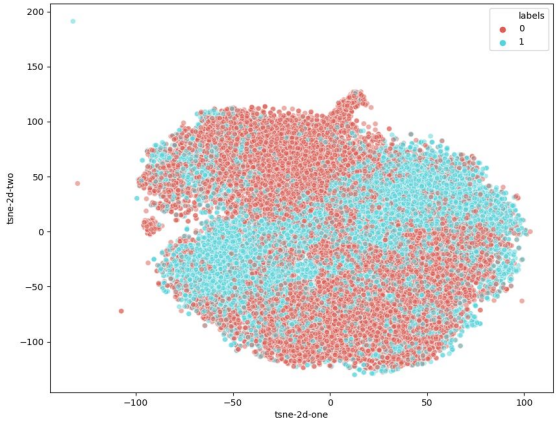
Feature Reuse - freezing pretrained layers

AUC-ROC on validation data set

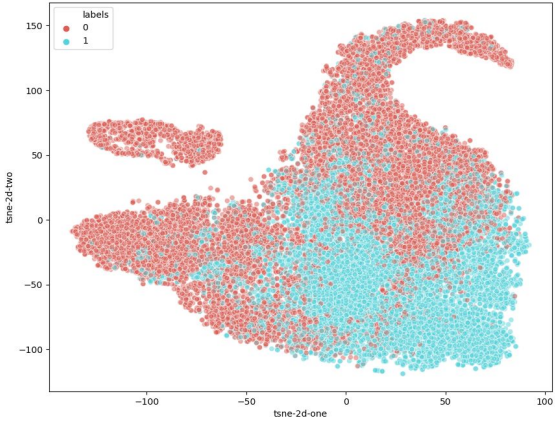


Feature Reuse - freezing pretrained layers

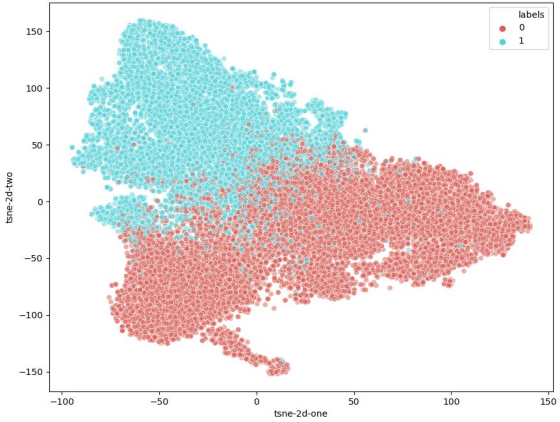
Cancer: T-SNE on feature vector for test set (best checkpoint, Inception v3).



100% frozen
(fixed feature extractor)



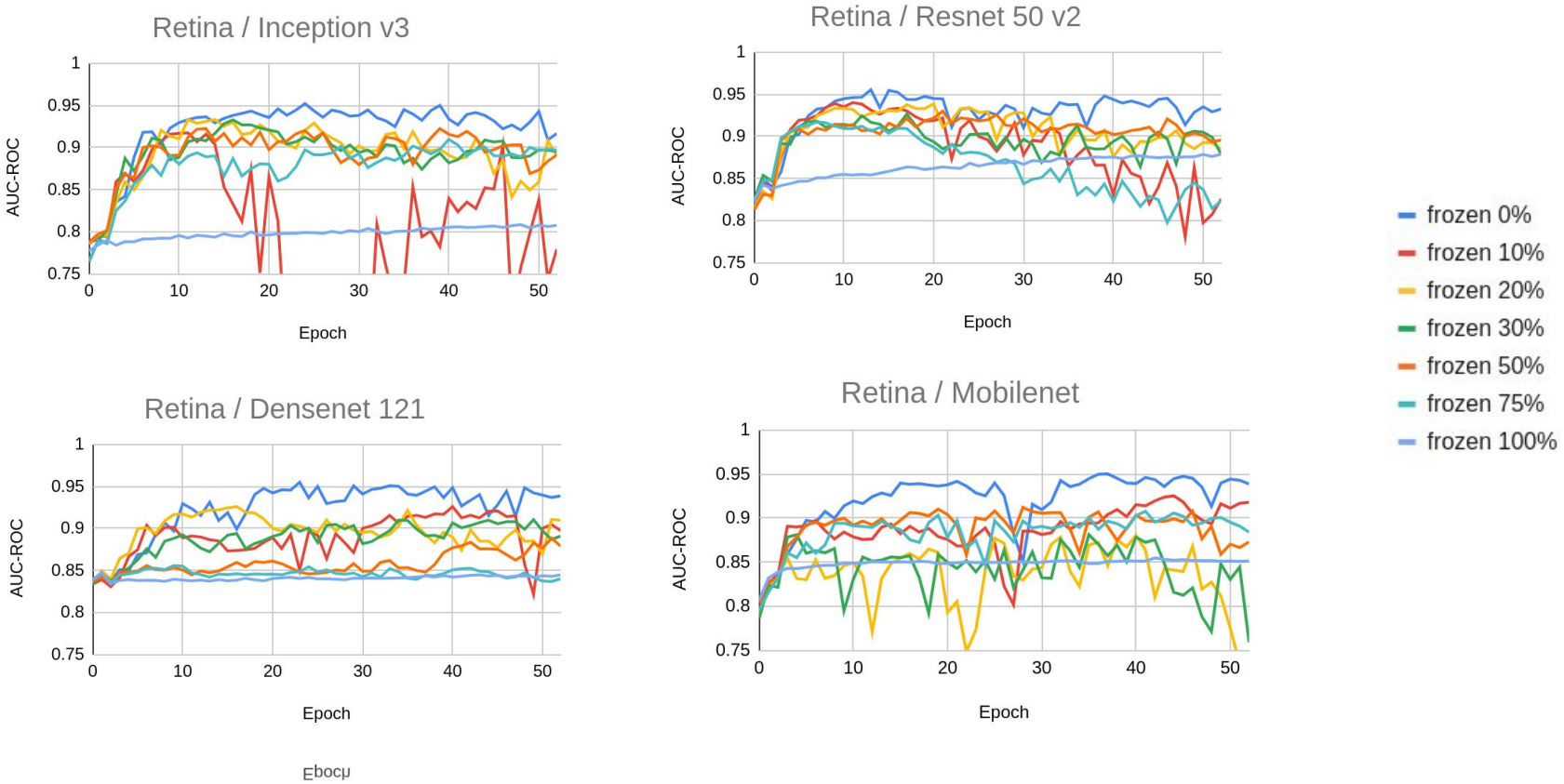
10% frozen,
90% fine-tuned



100% fine-tuned

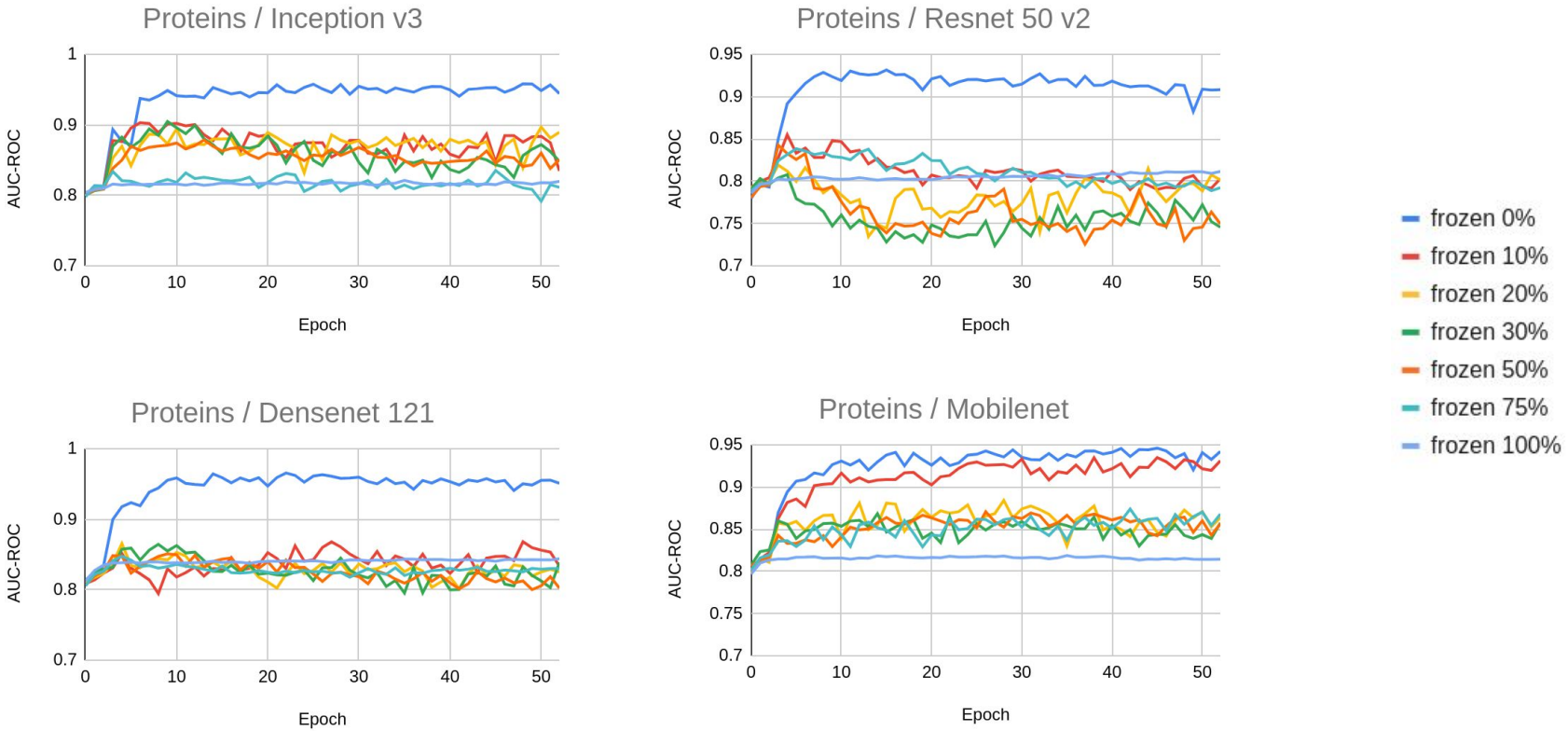
Feature Reuse - freezing pretrained layers

AUC-ROC on validation data set



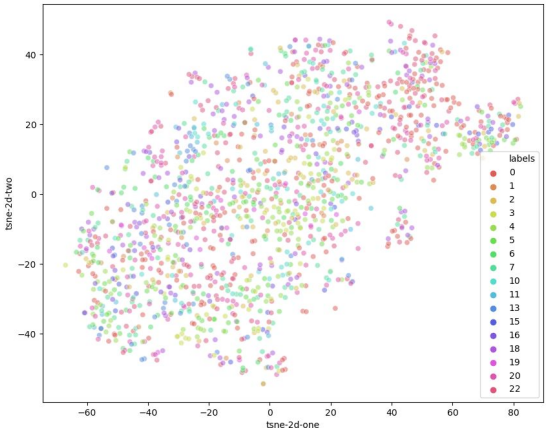
Feature Reuse - freezing pretrained layers

AUC-ROC on validation data set

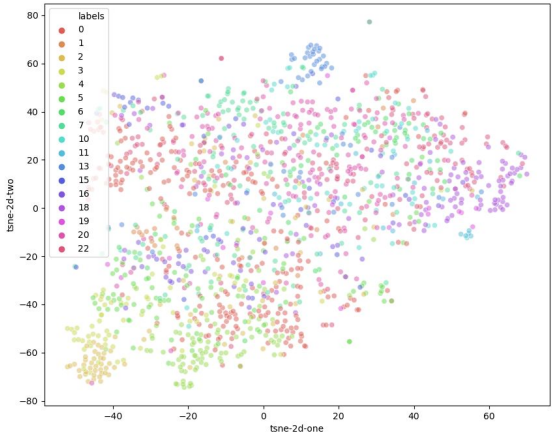


Feature Reuse - freezing pretrained layers

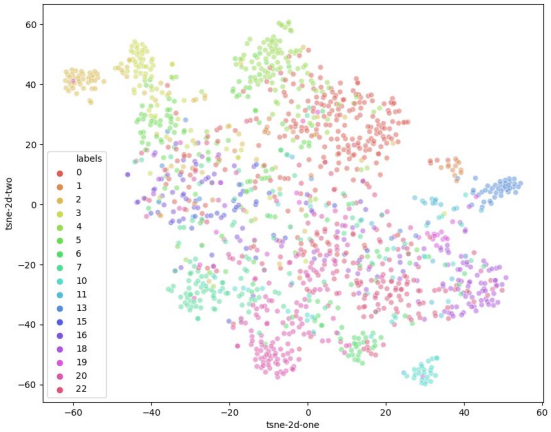
Proteins: T-SNE on feature vector for test set (best checkpoint, Inception v3).



100% frozen
(fixed feature extractor)



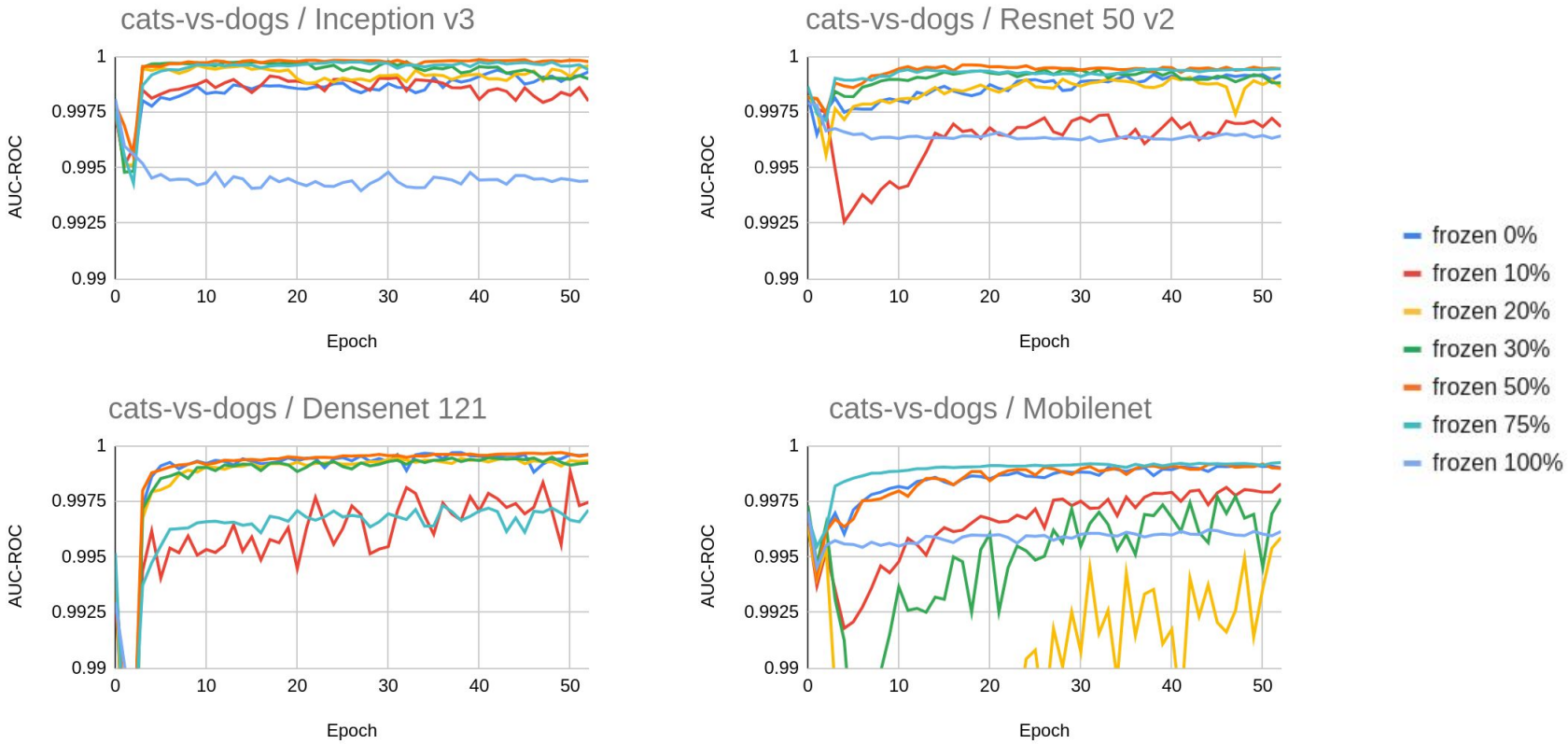
10% frozen,
90% fine-tuned



100% fine-tuned

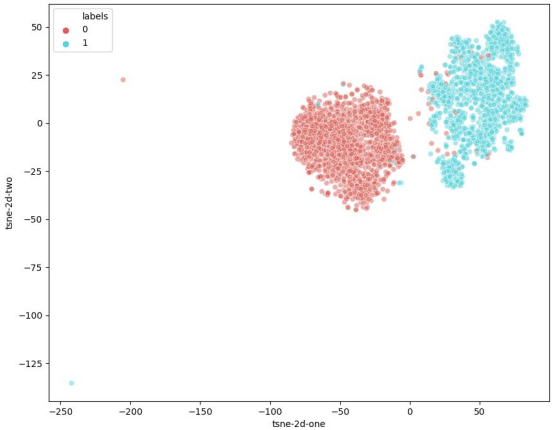
Feature Reuse - freezing pretrained layers

AUC-ROC on validation data set

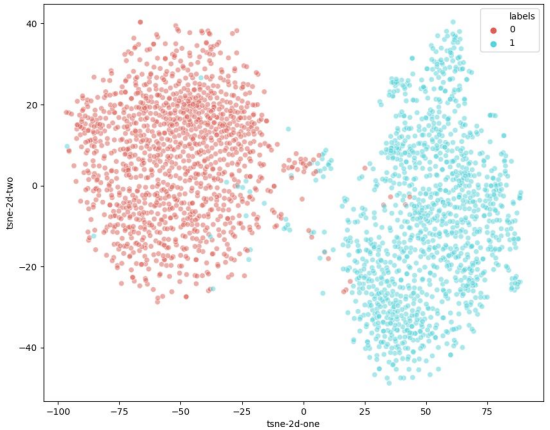


Feature Reuse - freezing pretrained layers

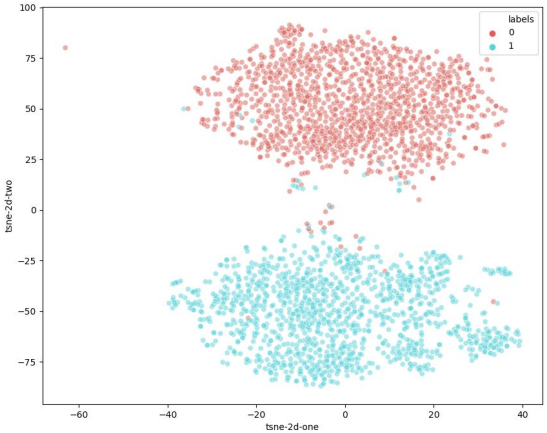
Cats vs dogs: T-SNE on feature vector for test set (best checkpoint, Inception v3).



100% frozen
(fixed feature extractor)



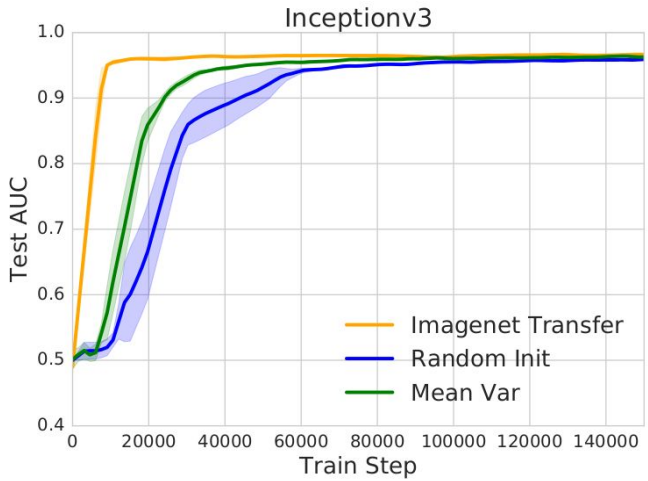
10% frozen,
90% fine-tuned

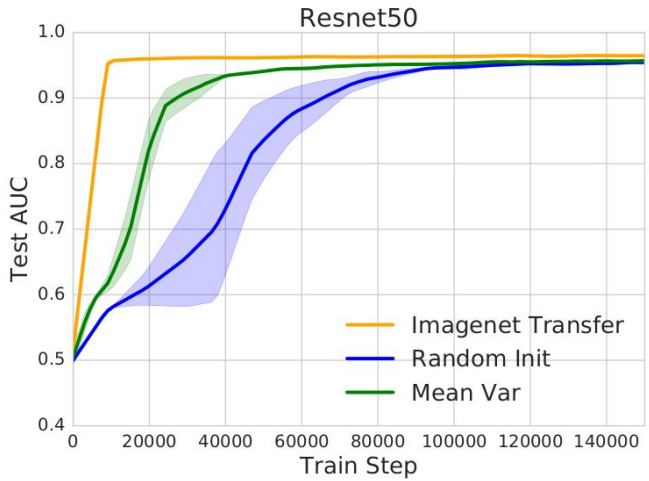


100% fine-tuned

Feature Independent Benefits of Transfer

- Proper scaling of the randomly initialized weights is critical
- Mean Var init inherits the scaling of weights, but destroys all of the features
- **Using only the scaling of the pretrained weights helps with convergence speed**





Charts from “Transfusion: Understanding Transfer Learning for Medical Imaging”, M. Raghu, C. Zhang, J. Kleinberg, S. Bengio, 2019, Google Brain & Cornell University

Conclusions

05

Conclusions

- Initializing with pretrained weights **improves the speed of training**
  - on average **7X speedup** based on validation set
  - but network may also overfit earlier than randomly initialized
- ImageNet initialized networks almost always **achieved higher performance** on test set
  - average **improvement 4.2%** (for 50+3 epochs)
- ImageNet features may not be reusable, but are a **very good initialization**
  - fine-tuning all the pretrained layers consistently gave the best results
- **Small custom CNNs** can achieve very good results in healthcare data sets
  - they perform **on pair with large networks** trained from scratch
  - large networks designed for ImageNet may be overparameterized for the problem

Doing now what patients need next