Microsoft

Hands-on Tutorial: Active learning and transfer learning at scale with R and Python

Mario Inchiosa
Ali-Kazim Zaidi
John-Mark Agosta
Robert Horton
Tomas Singliar
Olga Liakhovich
Vanja Paunic
Hang Zhang

Setting up your Virtual Machine

- Instructions at https://github.com/Azure/active-learning-workshop
- Open https://hostname:8000 (use https, not http; replace "hostname" with the hostname on the slip of paper you received when arriving). Disregard warnings about certificate errors.
- Open a bash terminal window by clicking the New button and then clicking Terminal.
- In the bash terminal, run these four commands:

```
cd ~/notebooks
git clone https://github.com/Azure/active-learning-workshop.git
cd active-learning-workshop
source startup.sh
```

- You can now log in to RStudio Server at http://hostname:8787 (unlike JupyterHub, be sure to use http, not https).

Note: please, download assets for exercise

- URLs: are in the tutorial readme

https://github.com/Azure/active-learning-workshop/

- Unzip

KDD 2018 Hands-on Tutorial: Active learning and transfer learning at scale with R and Python

Instructions

Provision an Ubuntu Linux Data Science Virtual Machine; the size "Standard_DS12_v2" works well
(**Note: at the start of the tutorial, credentials for pre-provisioned VMs will be handed out**):
https://azuremarketplace.microsoft.com/en-us/marketplace/apps/microsoft-ads.linux-data-science-vm-ubuntu

Log in to JupyterHub by pointing your web browser to https://hostname:8000 (be sure to use https, not http, and replace "hostname" with the hostname or IP address of your virtual machine). Please disgregard warnings about certificate errors.

Open a bash terminal window in JupyterHub by clicking the New button and then clicking Terminal.

In the terminal, run these four commands:

```
cd ~/notebooks

git clone https://github.com/Azure/active-learning-workshop.git

cd active-learning-workshop

source startup.sh
```

You can now log in to RStudio Server at http://hostname:8787 (unlike JupyterHub, be sure to use http, not https).

To provision many Data Science Virtual Machines using automation, see the scripts and the README file in
https://github.com/Azure/active-learning-workshop/blob/master/automation_scripts

Download assets for image labeling actvity:

1. Download release package (zip) of Visual Object Tagging Tool (VOTT)
2. Dowload images pre-labeled by Active Learning pipeline from here.

Session Goals

Learn how to:

- use **transfer learning** from pre-trained deep learning models to generate **features** that can be used in traditional machine learning approaches.
- run these types of featurization **at scale**.
- use **active learning** to take advantage of large sets of unlabeled data to build more accurate classifiers by selecting the most useful additional examples to label for training.
- optimize models by **tuning hyperparameters**.
- **deploy** models as web services.

Use Cases

Wiki detox

- Active learning from text data.
- Binary classifier: is this comment a personal attack?
- Featurization from pre-trained language model (USE).

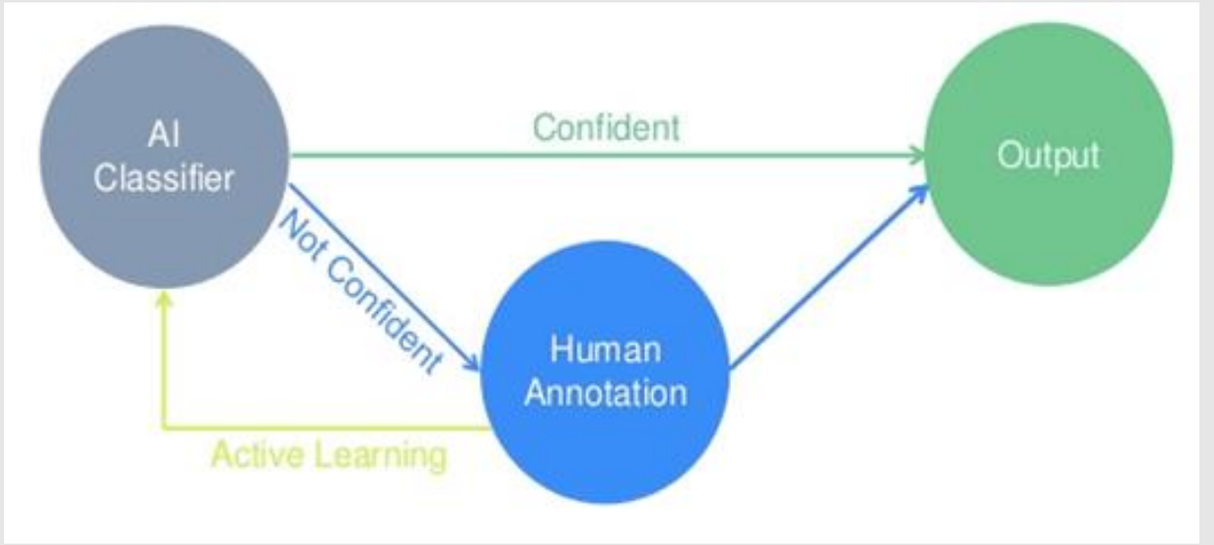
Wood knot images

- Active learning from image data.
- Multi-class classifier: which type of knot is this?
- Featurization from pre-trained deep learning model (Resnet)

Active Learning

- Active Learning intersperses labelling of samples with incremental re-training. The currently trained model is used to select new samples to label. There are variety of approaches for selecting samples, depending on the data and model.

- Data (unlabeled) is often easier to come by than expert labelers



* Image taken from https://www.crowdflower.com

Use Case 1: active learning for text classification

data, data everywhere

Wikipedia detox dataset

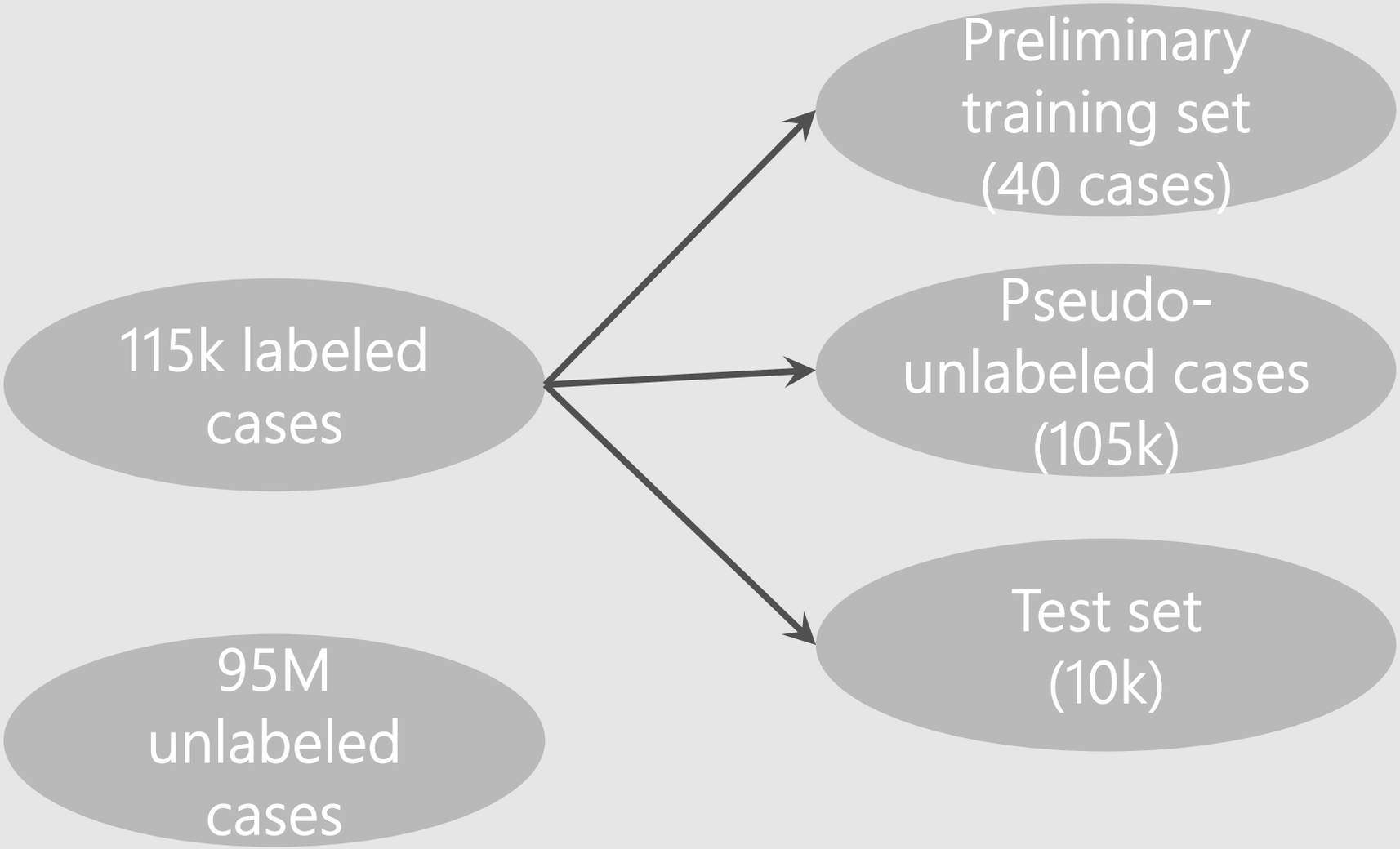
Ex Machina: Personal Attacks Seen at Scale

Ellery Wulczyn *
Wikimedia Foundation
ellery@wikimedia.org

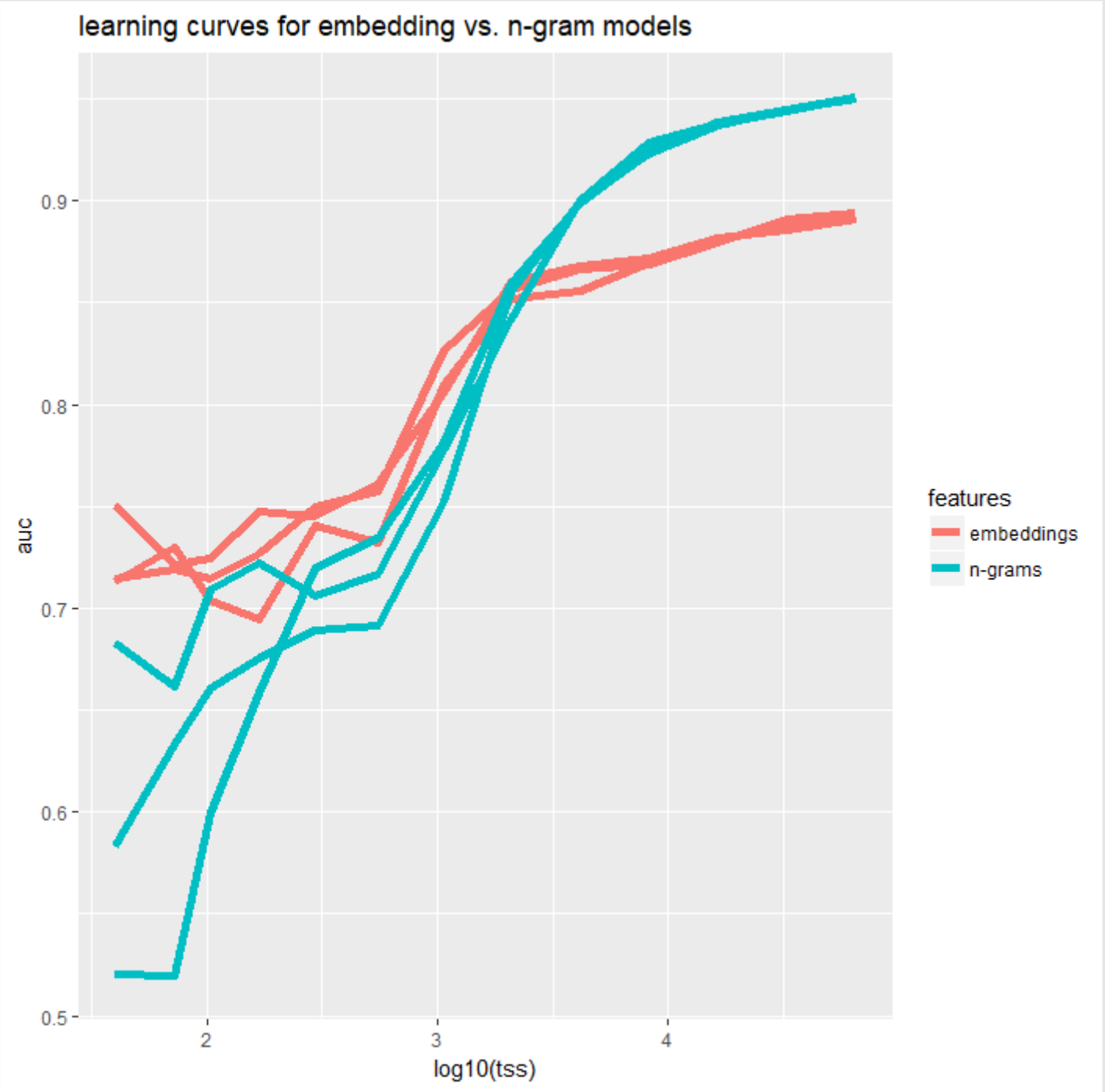
Nithum Thain *
Jigsaw
nthain@google.com

Lucas Dixon
Jigsaw
ldixon@google.com

arXiv:1610.08914v2

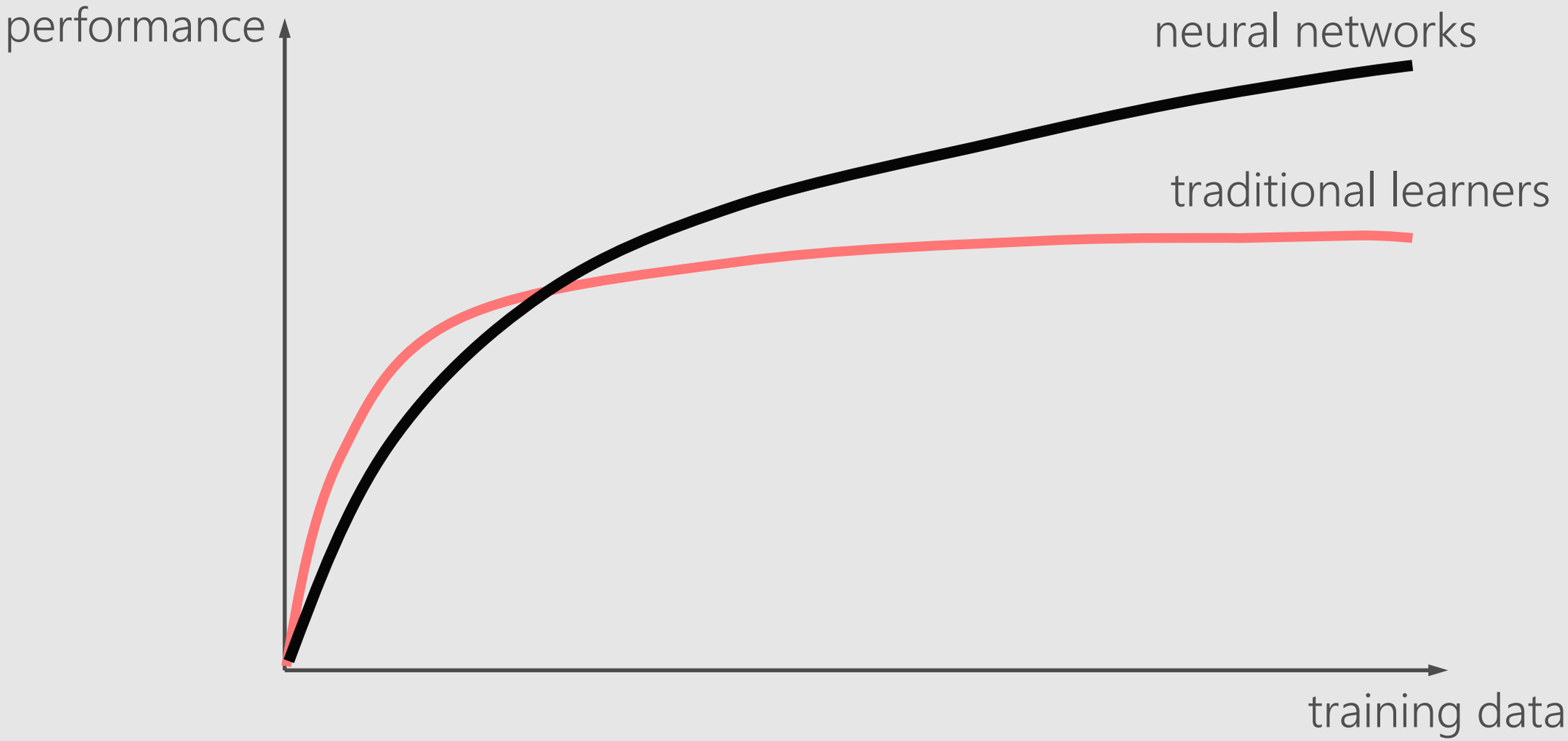


- Low complexity models work better with small training sets.
- With more data more complex models may give better performance.

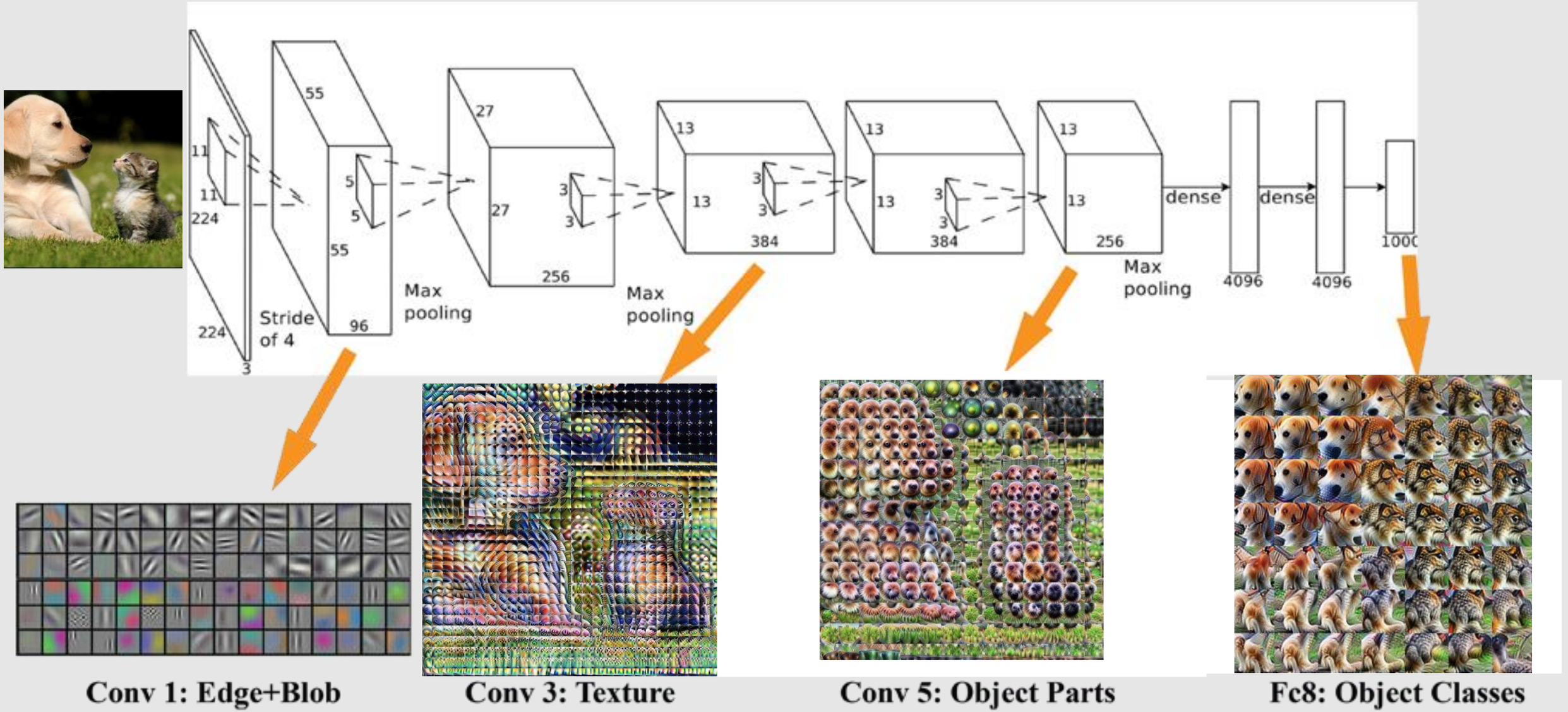


Text Featurization with Word Vectors and Universal Embeddings

Capacity and Inductive Biases



Revisiting Unreasonable Effectiveness of Data in Deep Learning Era, Google



CV: large datasets (ImageNet), good inductive biases (Convolutions + Classification), pre-training + transfer learning has become the norm

NLP: smaller datasets, unsure about the canonical task, stuck at word embeddings, end-to-end models for each task

What's The Canonical Task for Language?

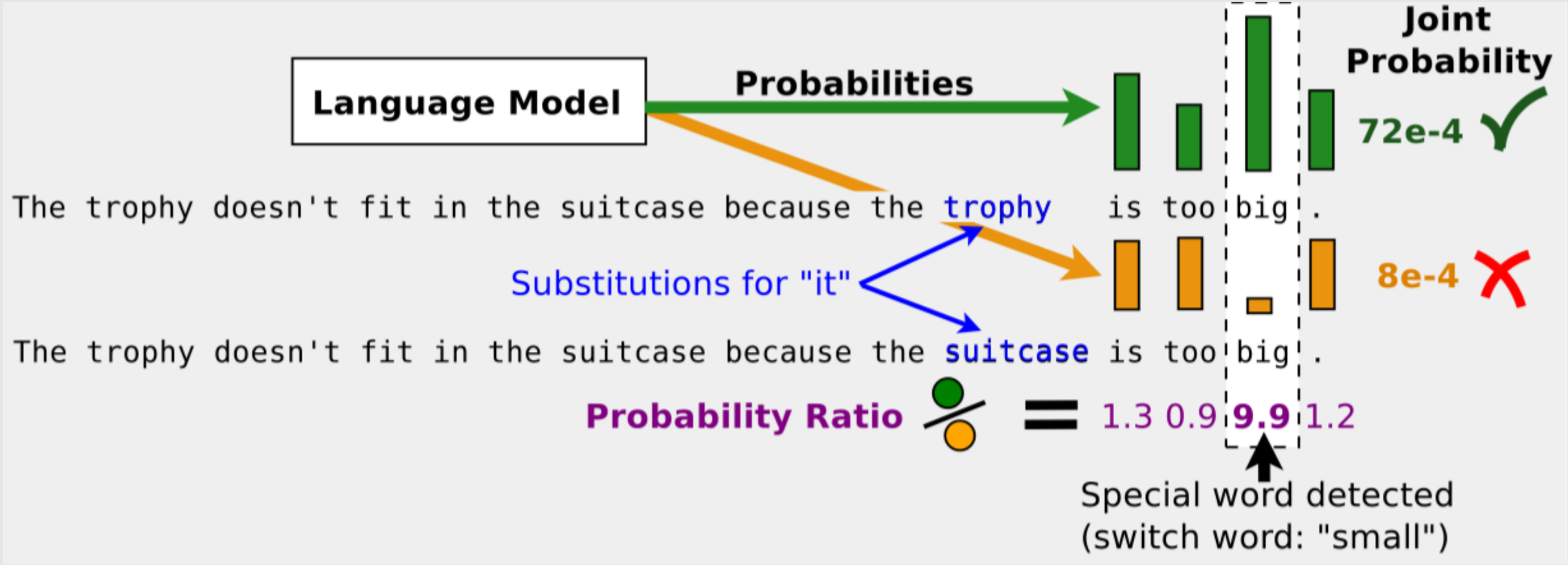
- Language modeling: predict the next word given the preceding words:
  - Given a sequence of words $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$, , compute the CPD:

$$\mathbb{P}\left(x^{(t+1)} = w_j | x^{(t)}, \ldots, x^{(1)}\right)$$

  - Completely unsupervised / self-supervised
  - Can leverage large unlabeled corpora, like Wikipedia, news, twitter, etc.
  - *Might* be the canonical task for NLU, e.g., Language Modeling Teaches You More than Translation Does: Lessons Learned Through Auxiliary Task Analysis, 2018
    - Other work has argued for NMT, McCann et. al 2017, Learned in Translation, dialogue, Q&A, McCann et. al 2018

Generative Pre-Training for NLP

- OpenAI: Improving Language Understanding by Generative Pre-Training
- fastAI: Universal Language Model Fine-tuning for Text Classification
- Trieu H. Trinh & Quoc Le: A Simple Method for Commonsense Reasoning



Activity 1A

Hands-On: Featurizing Comments from Wikipedia Talk Edits Using Word Vectors and Language Model Encoders

Active Learning by Uncertainty Sampling

Algorithm Sketch

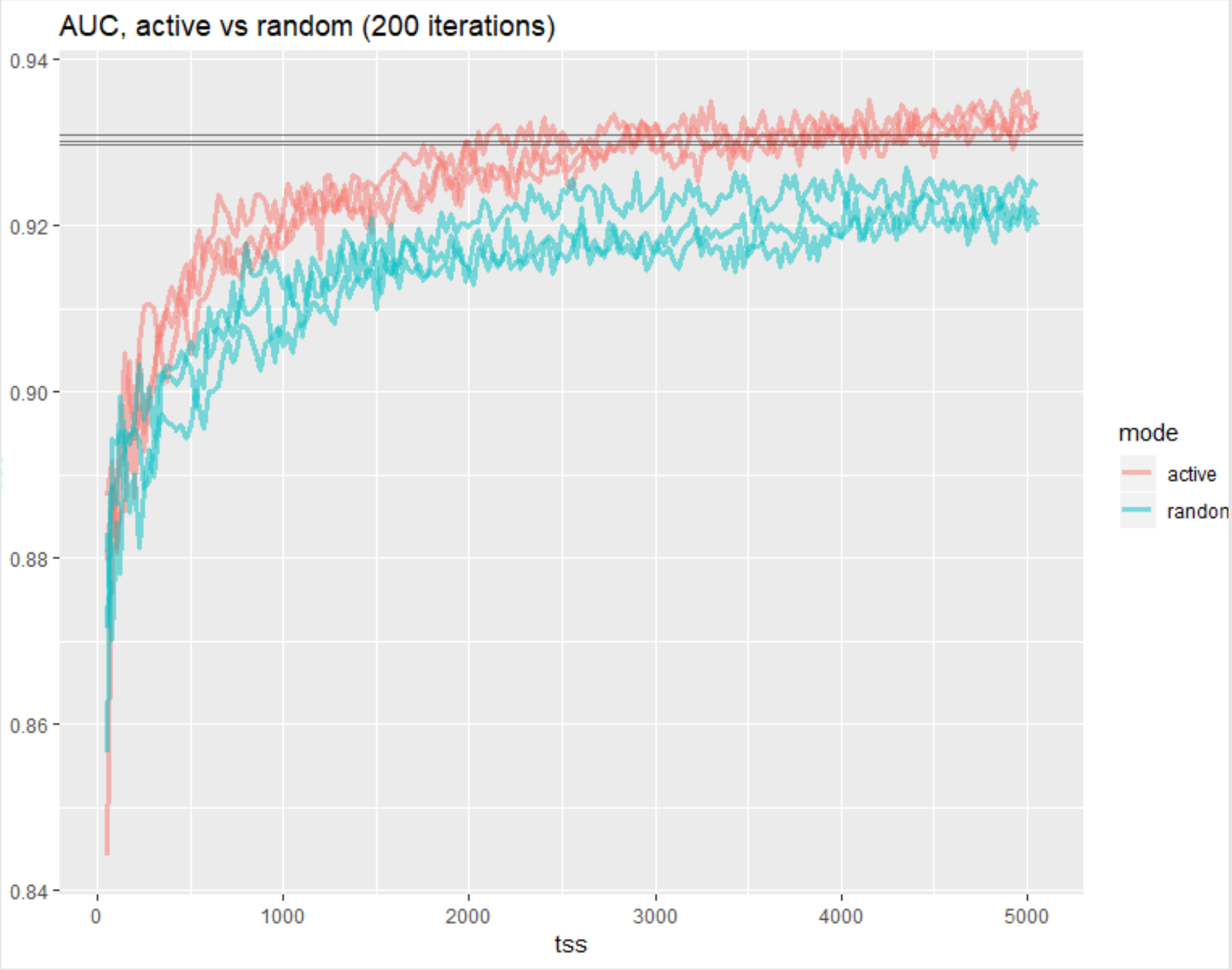
Given an initial model, *M* and unlabeled set of samples *U:*

1. **Using the current model *M* make class *c* likelihood predictions, *P(c | U) = M(U)*.**
2. **Select a set to label *L* (possibly one) from *U,* based on *P(c | U).***
3. **Update *M* with the training set *T' <- T + L***

Repeat until model improvement / labeling cost < threshold

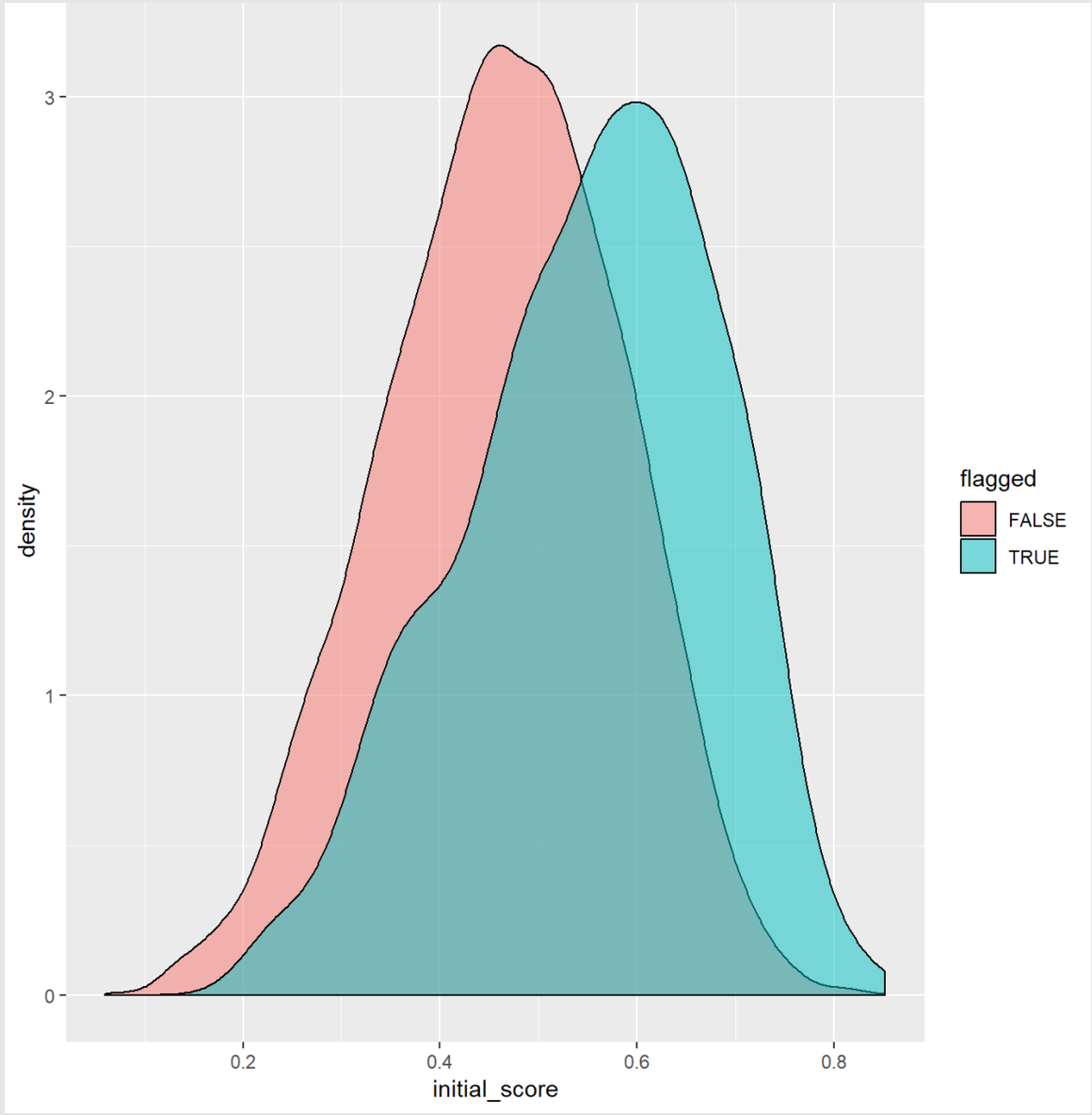
Active Learning methods differ by the way they use *P(c | U)* to search over the set *U*.

Learning curves show expected gains from Active Learning



Uncertainty Sampling

- The intuition is that unlabeled samples that the model predicts with greater uncertainty are more likely to be informative.
- This implies a strategy to select samples assigned the most uninformative probability by the current class *c* likelihood *P(c | U)*.



Three uncertainty-sampling sample selection methods

Methods differ by how *P(c | U)* is used to select from *U*.

- Query Synthesis [Anguin, 1988]
- Selective Sampling [Atlas et al. 1989]
- Pool-based Active Learning [Lewis & Gale, SIGIR 1994] [Text classification - Lewis & Gale ICML 1994]

Query Synthesis

- Generate a query of where to look in the feature space *x* to select items to label. This applies with feature space representations where a "sample" could mean generating an *x ab initio* rather than selecting from an existing set *U*.
- For example *x* could be an chemical synthesis, or a synthetic image whose outcome is passed to the model learner.
- There's a rough analog to the generative step in current DNN adversarial networks.

Selective Sampling

- Choose a region in feature space to focus on that is predicted to have the greatest uncertainty or information gain.
- This applies best when gaining new samples is passive or free, such as when selecting from streaming samples.
- Samples are selected sequentially from the stream.
- Unlike with *Query Synthesis*, samples are guaranteed to represent the actual distribution of the data, *P(U)*.
- For example, generate image samples by aiming a camera at areas that need clarification.

Pool-based Sampling

- Choose greedily among the existing set of unlabeled samples *U* by an uncertainty measure applied to each element in the set.
- Batch sampling: one or more samples may be selected at each stage.
- When labelling costs vary among samples they may also be considered along with information gain.
- The examples in this tutorial will demonstrate Pool-based Sampling.

Activity

Active learning for text classification

https://github.com/Azure/active-learning-workshop/tree/master/text_classification

Hyperparameter Tuning Using mmlspark

Activity

https://github.com/Azure/active-learning-workshop/blob/master/text_classification/tuning/find-best-model.ipynb

Serving the Model Using mmlspark

Activity

https://github.com/Azure/active-learning-workshop/blob/master/deployment/TextClassificationWithMMLSpark.ipynb

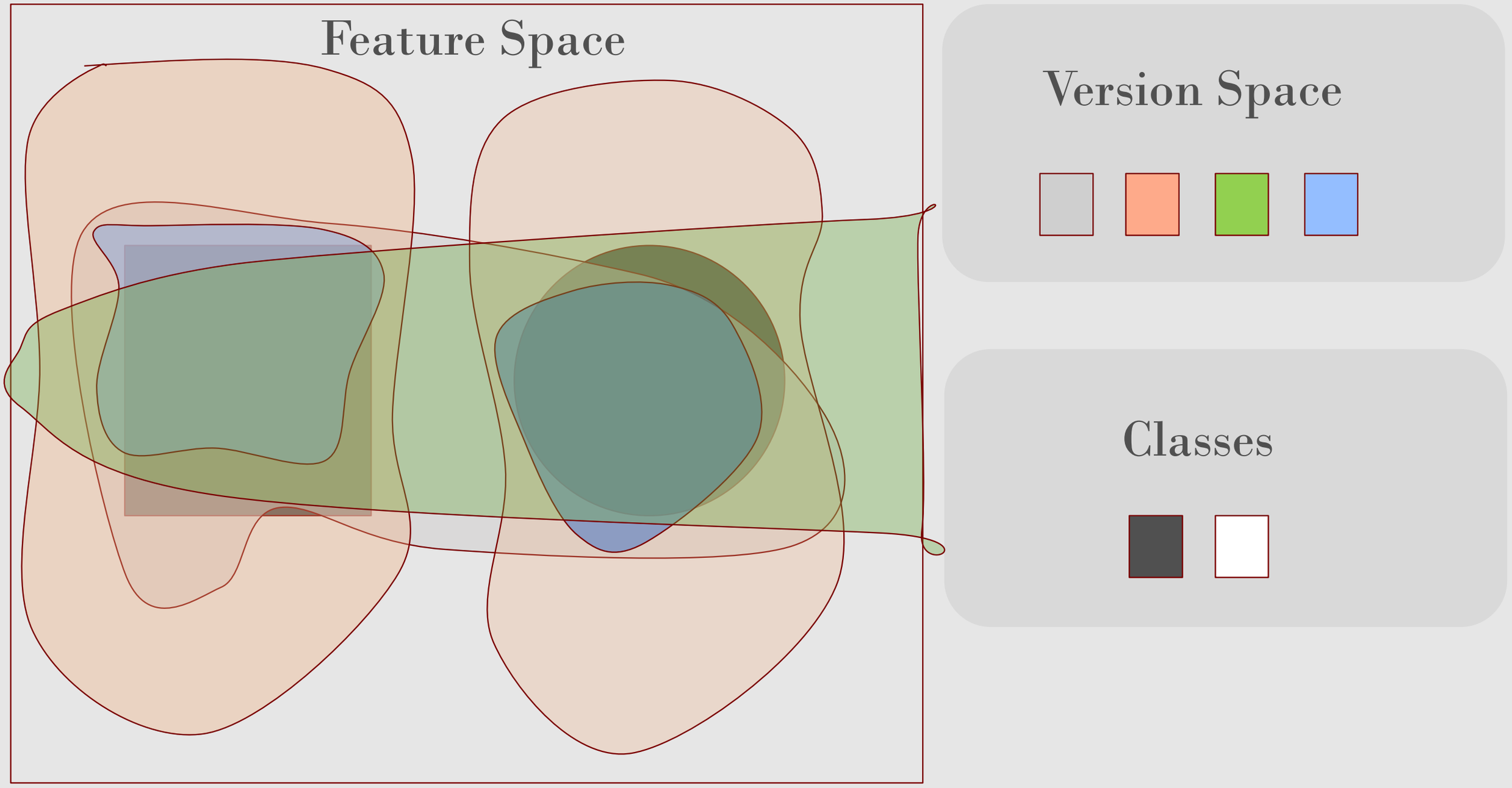
How Uncertainty Sampling Fails

How Sampling Fails

- *P(c | u)* often picks samples that are irrelevant, since there are areas of the sample space that are uncertain but do not help distinguish classes.
- Example: outliers may be highly uncertain but uninformative.
- Better to combine the class likelihood *P(c | u, x)* with the areas of feature space likely to distinguish known classes.
- Learners that generate margins for the class separators can find unlabeled samples both uncertain and that discriminate strongly between classes.

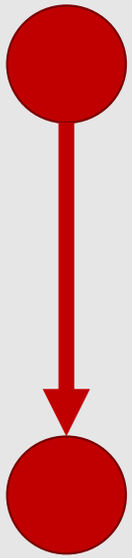
Working in the Version Space

- The space of all hypotheses is called the *version space*. Think all possible separators for a linear classifier.
- The version space is the dual to the feature space. Active Learning can be posed as maximizing the reduction in the version space by choice of samples.
- For instance, a sample that eliminates half the version space would best reduce model uncertainty.
- When model predictions are uncertain there are also Bayesian interpretations of the version space.


Feature Space
Version Space
Classes

Uncertainty about uncertainty

- 1. Classifiers all consider a sample uncertain.
- 2. Classifiers consider a sample certain, but disagree, so on average the sample is uncertain.
- Case 2. is more informative, since the "second order" uncertainty is at the version space level.



Distribution over version spaces

Predicted class given the version space

INTERMISSION

Use Case 2: Building a custom image classifier for wood knots

Note: please, download assets for exercise

- URLs: are in the tutorial readme

https://github.com/Azure/active-learning-workshop/

- Unzip

KDD 2018 Hands-on Tutorial: Active learning and transfer learning at scale with R and Python

Instructions

Provision an Ubuntu Linux Data Science Virtual Machine; the size "Standard_DS12_v2" works well
(**Note: at the start of the tutorial, credentials for pre-provisioned VMs will be handed out**):
https://azuremarketplace.microsoft.com/en-us/marketplace/apps/microsoft-ads.linux-data-science-vm-ubuntu

Log in to JupyterHub by pointing your web browser to https://hostname:8000 (be sure to use https, not http, and replace "hostname" with the hostname or IP address of your virtual machine). Please disgregard warnings about certificate errors.

Open a bash terminal window in JupyterHub by clicking the New button and then clicking Terminal.

In the terminal, run these four commands:

```
cd ~/notebooks

git clone https://github.com/Azure/active-learning-workshop.git

cd active-learning-workshop

source startup.sh
```

You can now log in to RStudio Server at http://hostname:8787 (unlike JupyterHub, be sure to use http, not https).

To provision many Data Science Virtual Machines using automation, see the scripts and the README file in
https://github.com/Azure/active-learning-workshop/blob/master/automation_scripts

Download assets for image labeling actvity:

1. Download release package (zip) of Visual Object Tagging Tool (VOTT)
2. Dowload images pre-labeled by Active Learning pipeline from here.

Domain: Wood Knots and Lumber Grading

- In the sawmill industry lumber grading is an important step of the manufacturing process.
- Improved grading accuracy and better control of quality variation in production leads directly to improved profits.
- Grading has traditionally been done by visual inspection, in which a (human) grader marks each piece of lumber as it leaves the mill, according to a factors like size, category, and position of knots, cracks, species of tree, etc.
- A number of automated lumber grading systems have been developed which aim to improve the accuracy and the efficiency of lumber grading.

Data: pictures of boards

Types of wood knots

Sound knot: A knot grown firmly into the surrounding wood material and does not contain any bark or signs of decay. The color may be very close to the color of sound wood.

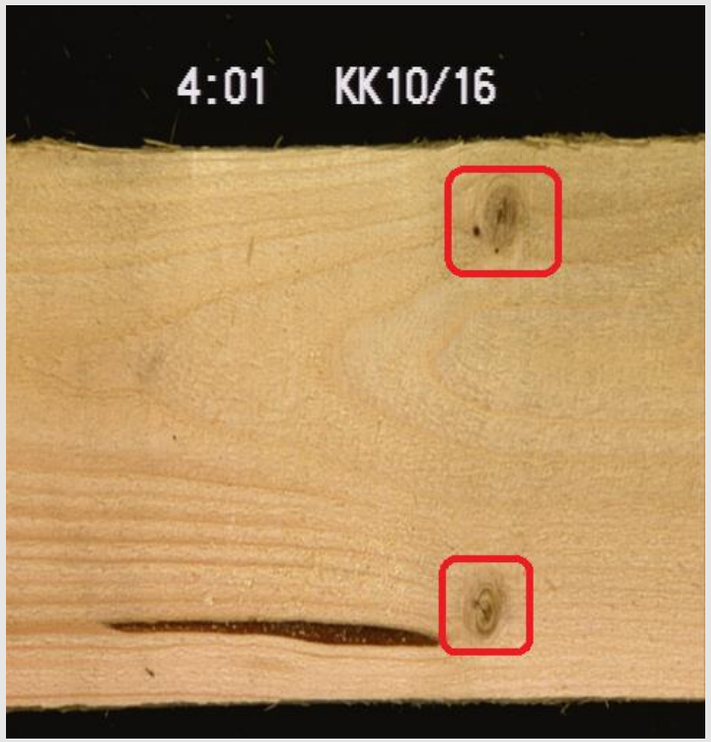
Dry knot: A firm or partially firm knot, and has not taken part to the vital processes of growing wood, and does not contain any bark or signs of decay. The color is usually darker than the color of sound wood, and a thin dark ring or a partial ring surrounds the knot.

Encased knot: A knot surrounded totally or partially by a bark ring. Compared to dry knot, the ring around the knot is thicker.

Active Learning for Object Detection

Two phase approach

Step1: prepare data for classification (do not need lumber expert)
Step2: classify wood knots images (need lumber expert to label data



Sound knot?

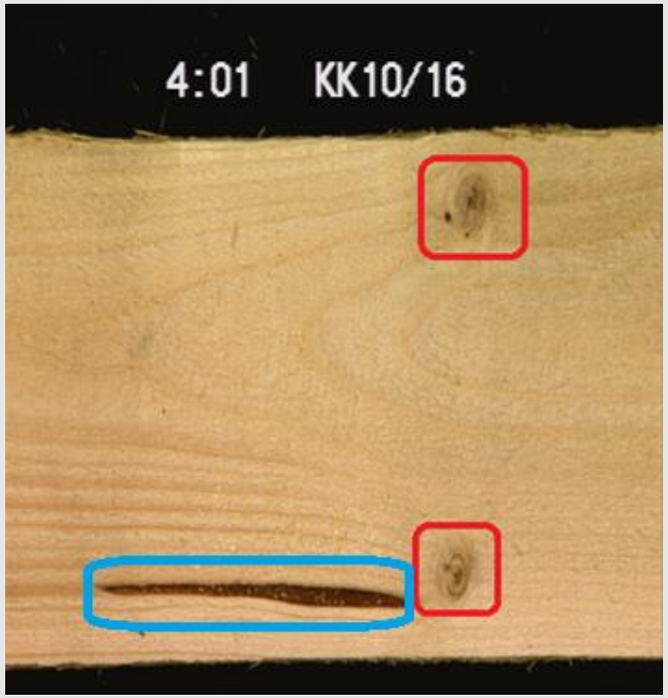
Dry knot?

Encased knot?

DNN Based Object detection

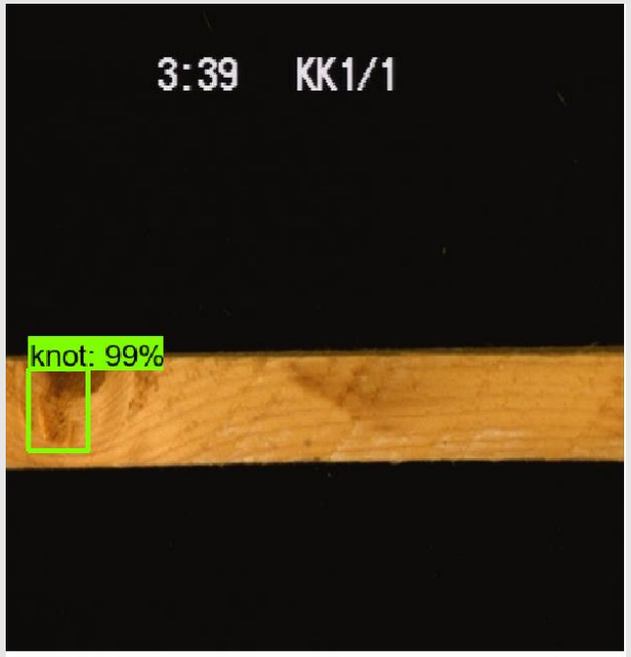
Rather than manually cropping knot images from boards train Object Detection model predict knots bonding boxes. Human will verify predictions and assign labels (bboxes).

Label knots and defects



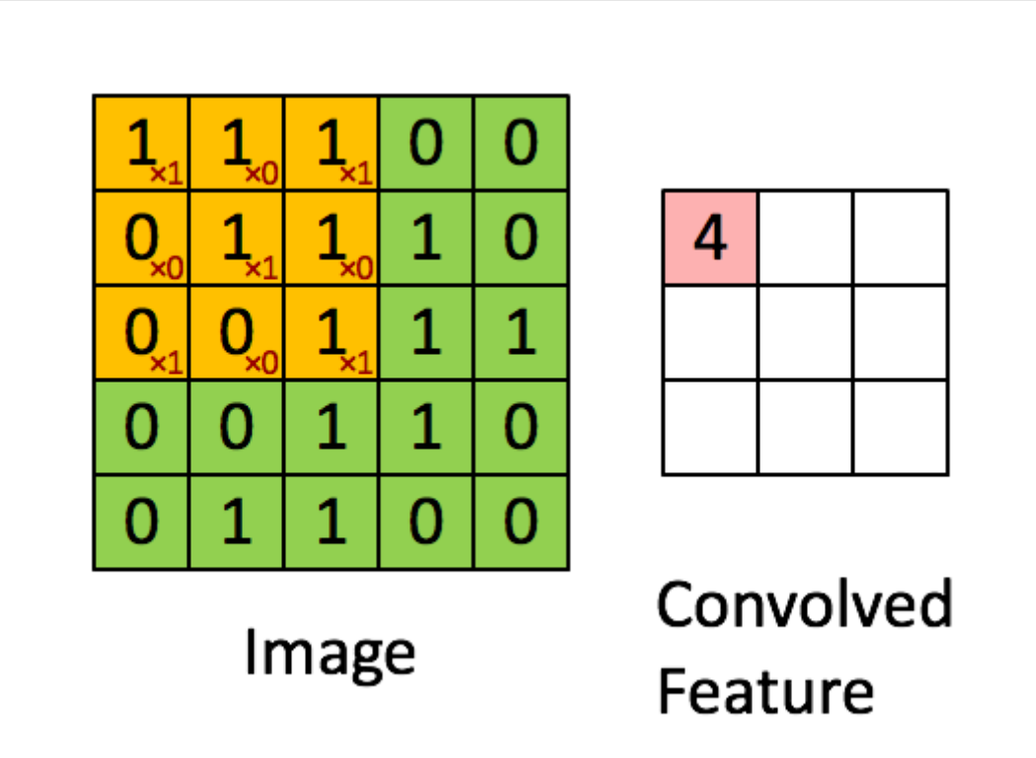
Train Object Detection Model

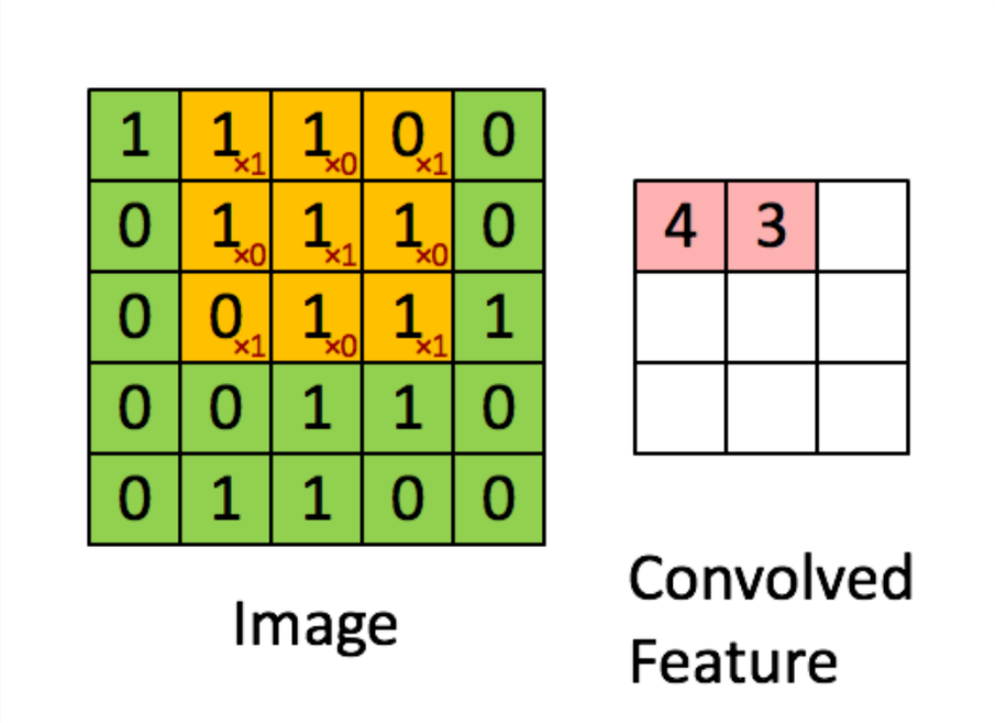
Verify predicted bboxes



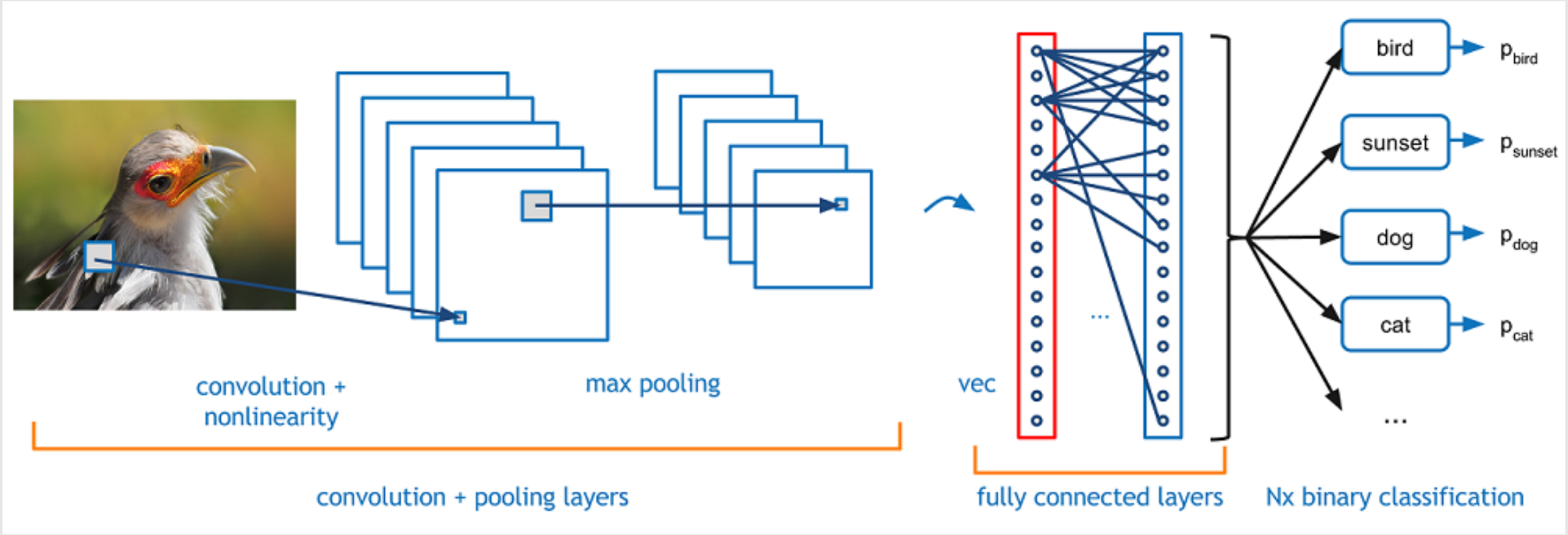
Intro to Object Detection: Convolutions

- Sliding window function applied to a matrix
- Element wise multiplication
- Learns image features (spatial relationship between pixels is preserved)

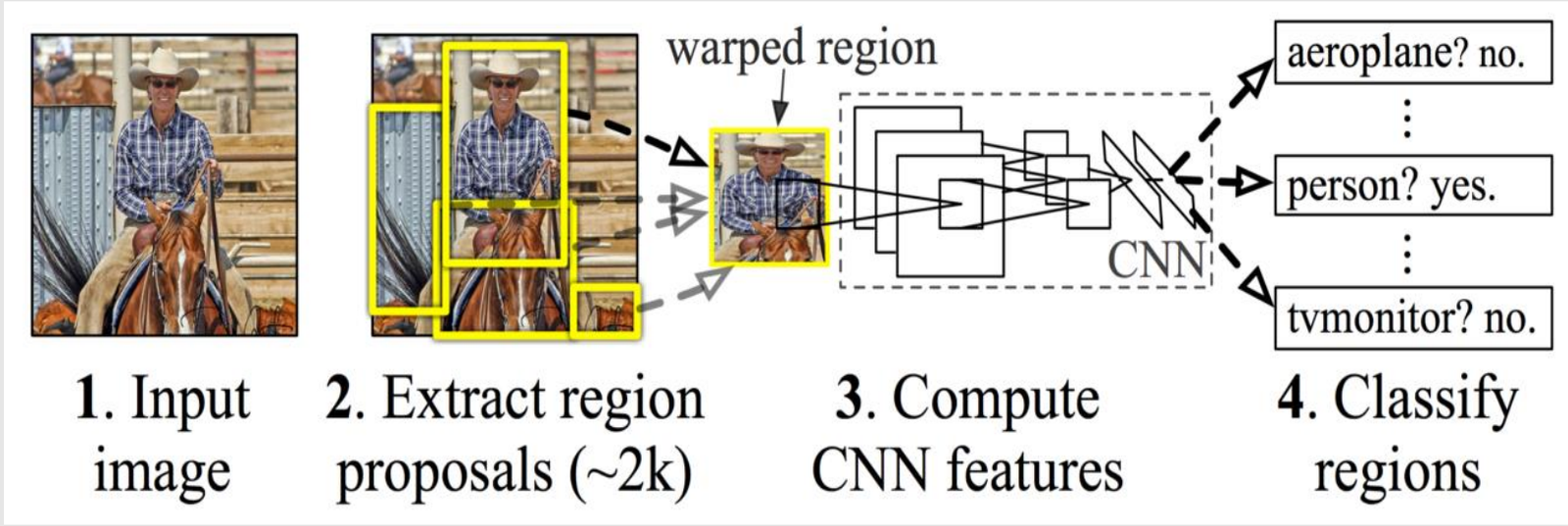


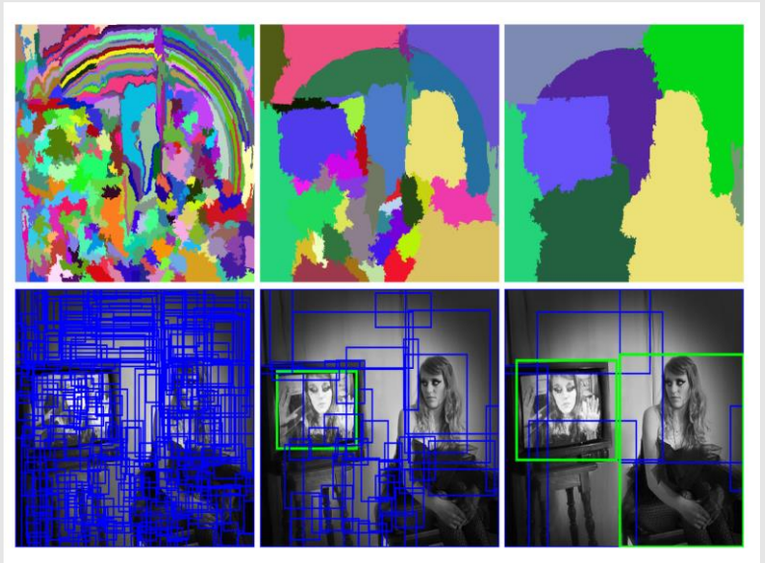


Intro to Object Detection: CNN



Intro to Object Detection: Region-based CNNs





Selective search

CNN network

SVM

Tensorflow Object Detection API

https://github.com/tensorflow/models/blob/master/research/object_detection

- Variety of powerful pre-trained models :
  - Faster RCNN with ResNet 50, 101, 152;
  - RFCN with Resnet 101
  - SSD
- Rich config files: tune learning rate, optimizer, data augmentation

Pipeline for data labeling

https://github.com/olgaliak/active-learning-detect

Label initial dataset and get predictions

- Spend 1-2 hours
- Train simple model
- Select set of images where model was less certain
- Review predicted bounding boxes for knots

Train model #2 and get predictions

- Model #2 will have more data
- May need to train longer (monitor train\validation loss)
- Get newer set of predicted bounding boxes for knots
- …. [repeat]

Activity

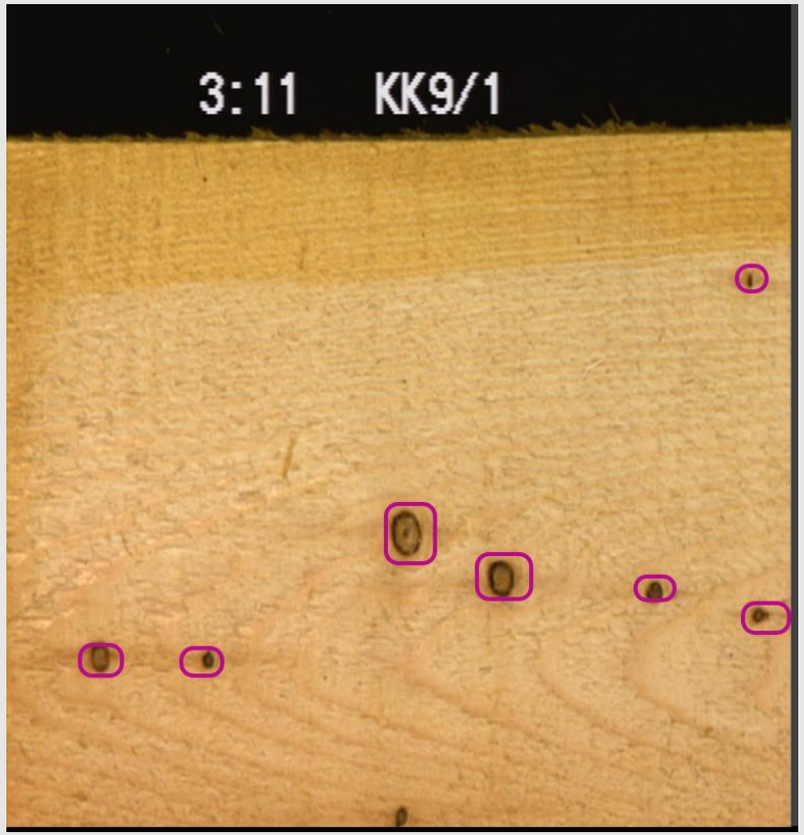
Review pre-labeled images

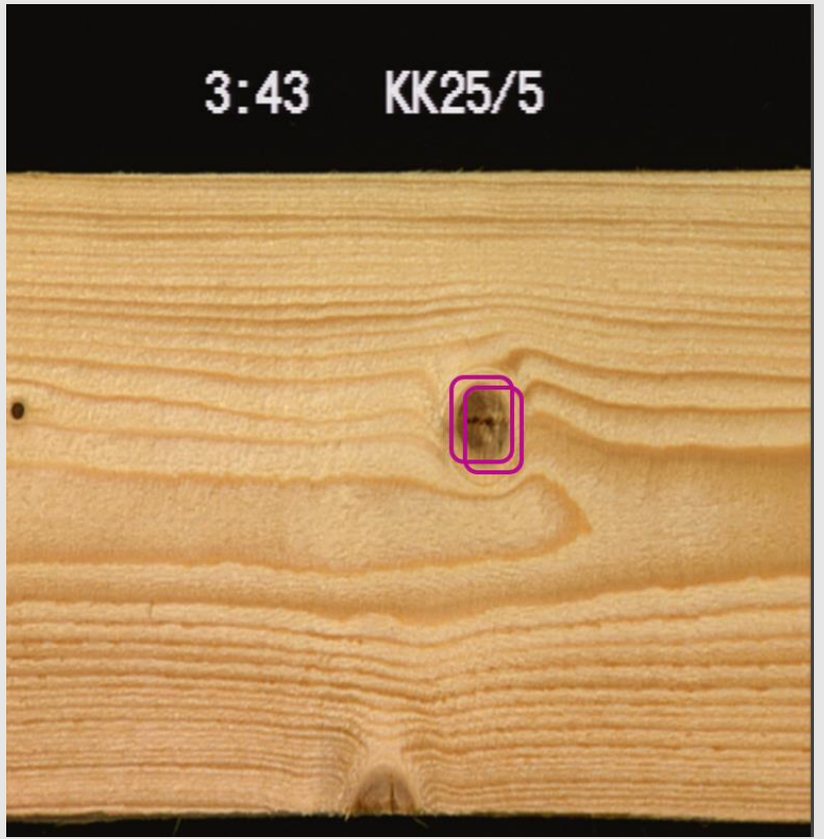
Review pre-tagging results (model 1)

- Model trained only on 100 images and
- Only 30 images have predictions with confidence > 70
- Already able to find some notes
- Model is not certain: multiple bounding boxes over one knot

Review pre-tagging results (model 2)

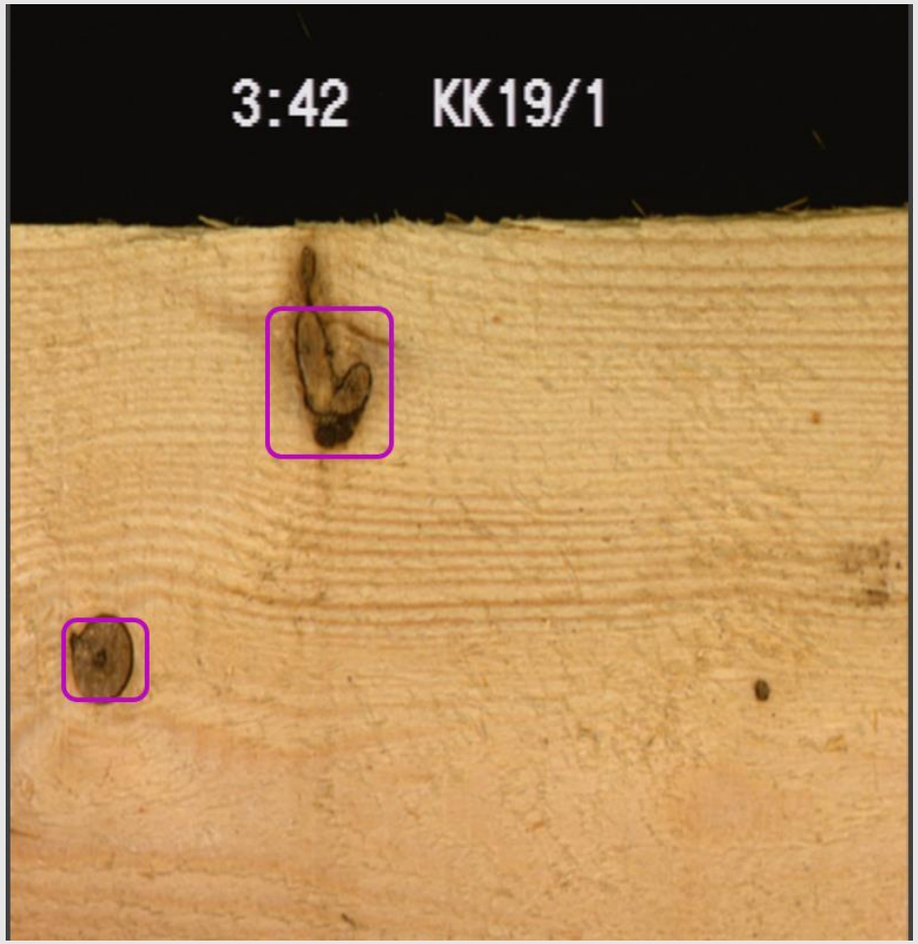
- Model trained on 100+30 = 130 images
- 100 images with predictions with confidence > 70 were reviewed
- Better performance
- Models is not certain: multiple bounding boxes over one knot

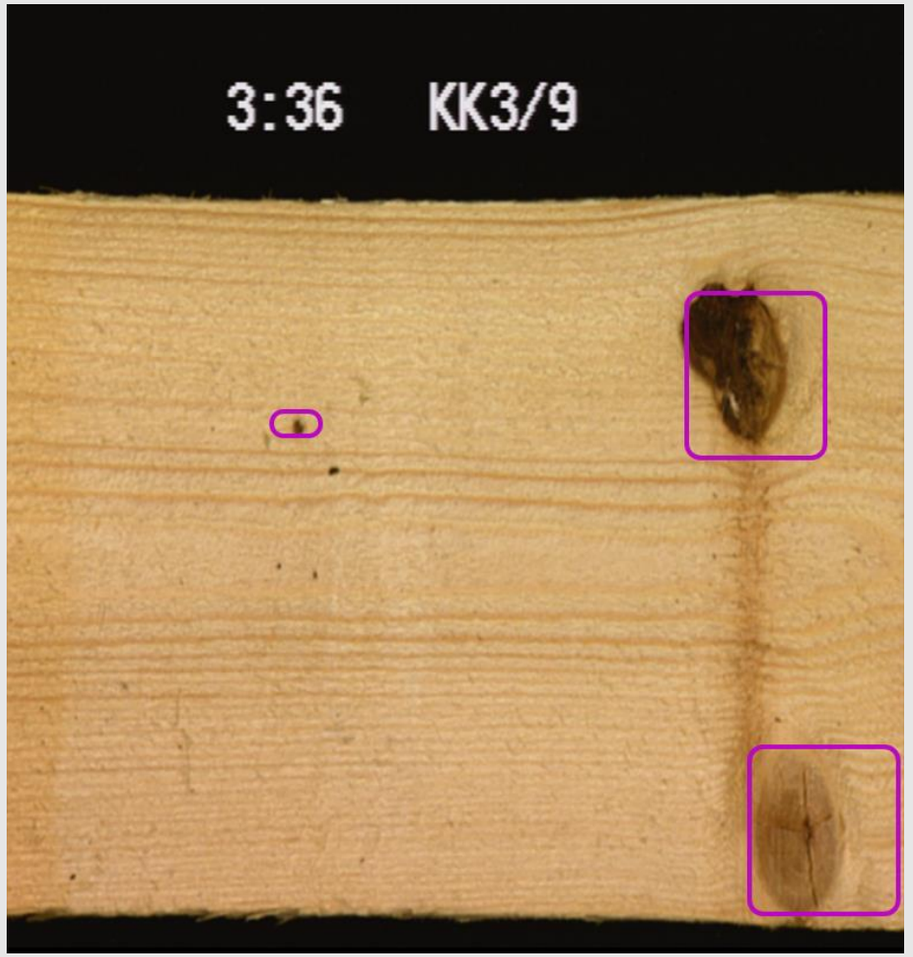




Review pre-tagging results (model 3)

- Model trained on 100+30 +100= 230 images
- Much less “duplication” of bboxes over one knot



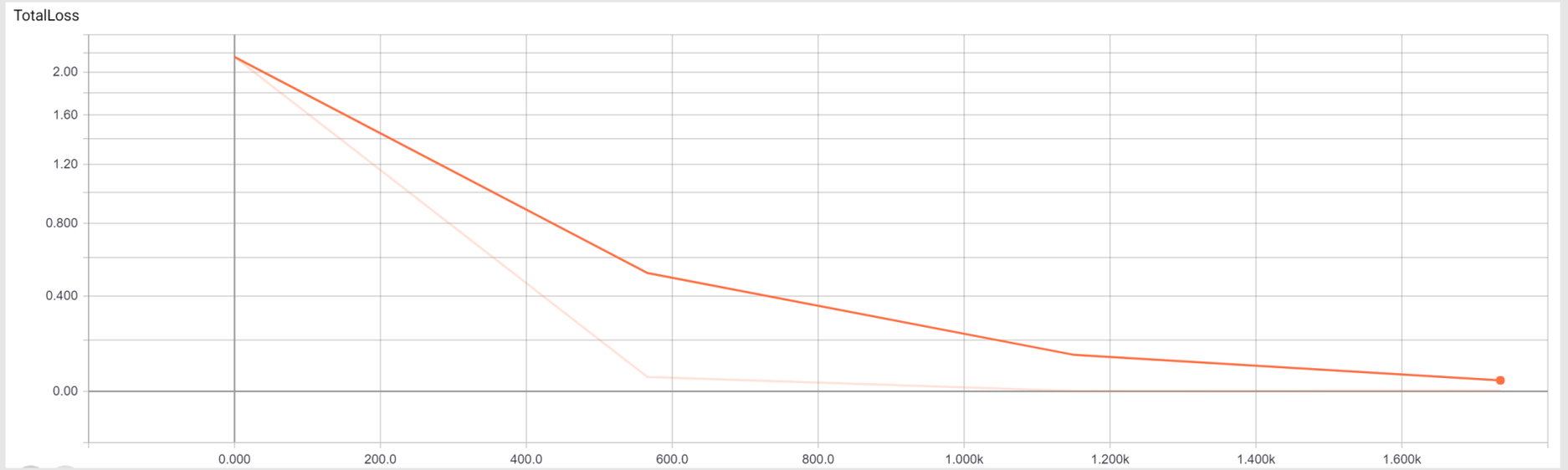


Training loss

Model 1:

Trained on 100 images

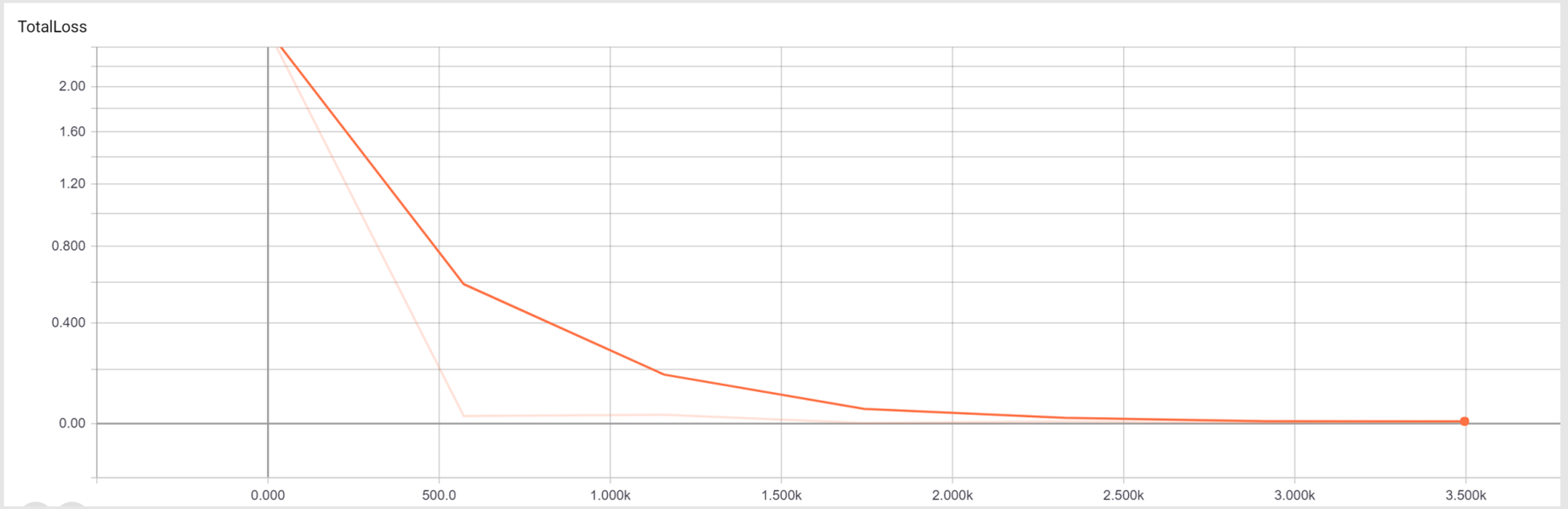
2k iterations



Model 4:

Trained on 330 images

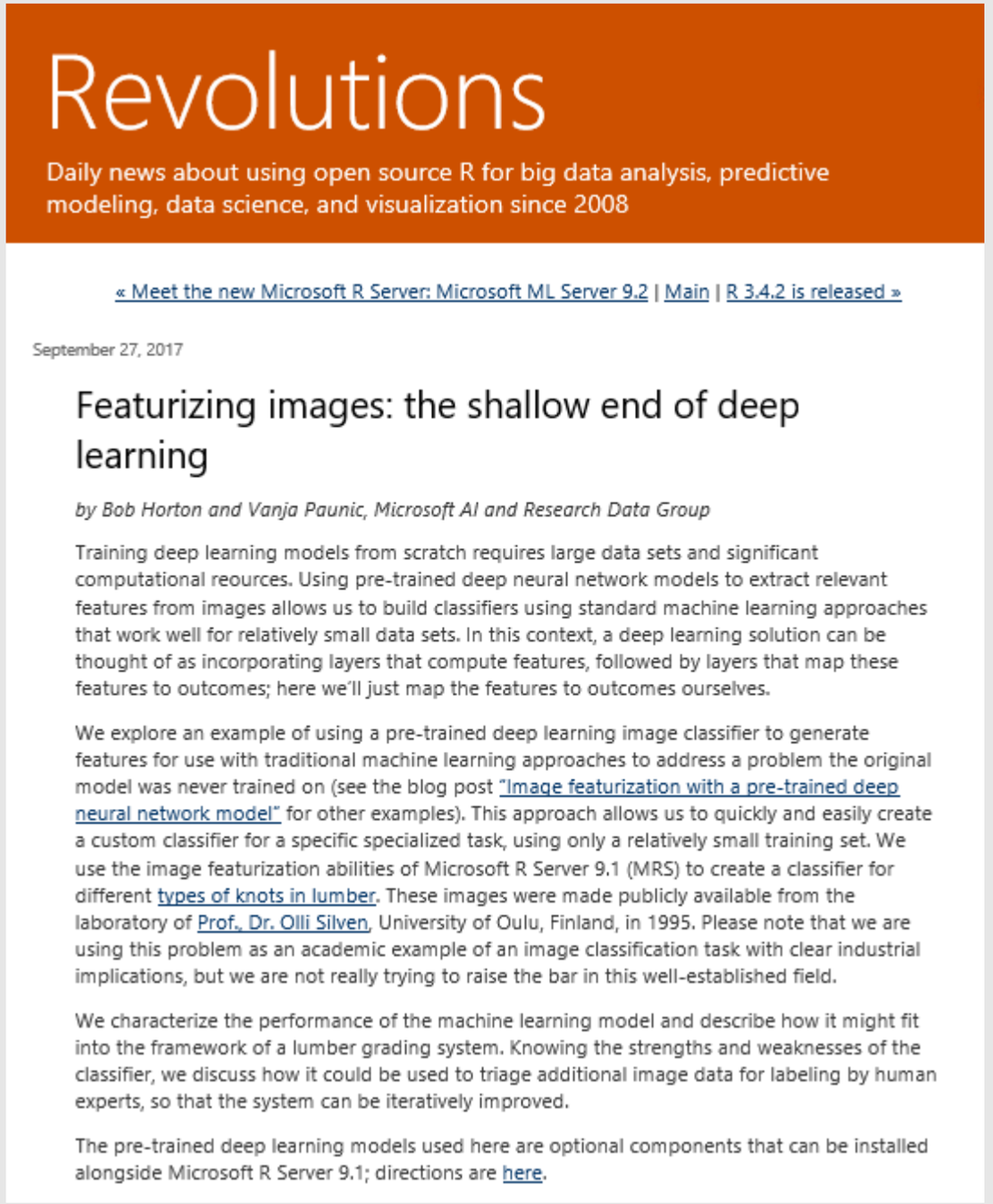
4k iterations



Featurizing images at scale for building custom image classifier

Featurizing images: the shallow end of deep learning

- http://blog.revolutionanalytics.com/2017/09/wood-knots.html



Revolutions

Daily news about using open source R for big data analysis, predictive modeling, data science, and visualization since 2008

« Meet the new Microsoft R Server: Microsoft ML Server 9.2 | Main | R 3.4.2 is released »

September 27, 2017

Featurizing images: the shallow end of deep learning

by Bob Horton and Vanja Paunic, Microsoft AI and Research Data Group

Training deep learning models from scratch requires large data sets and significant computational reources. Using pre-trained deep neural network models to extract relevant features from images allows us to build classifiers using standard machine learning approaches that work well for relatively small data sets. In this context, a deep learning solution can be thought of as incorporating layers that compute features, followed by layers that map these features to outcomes; here we'll just map the features to outcomes ourselves.

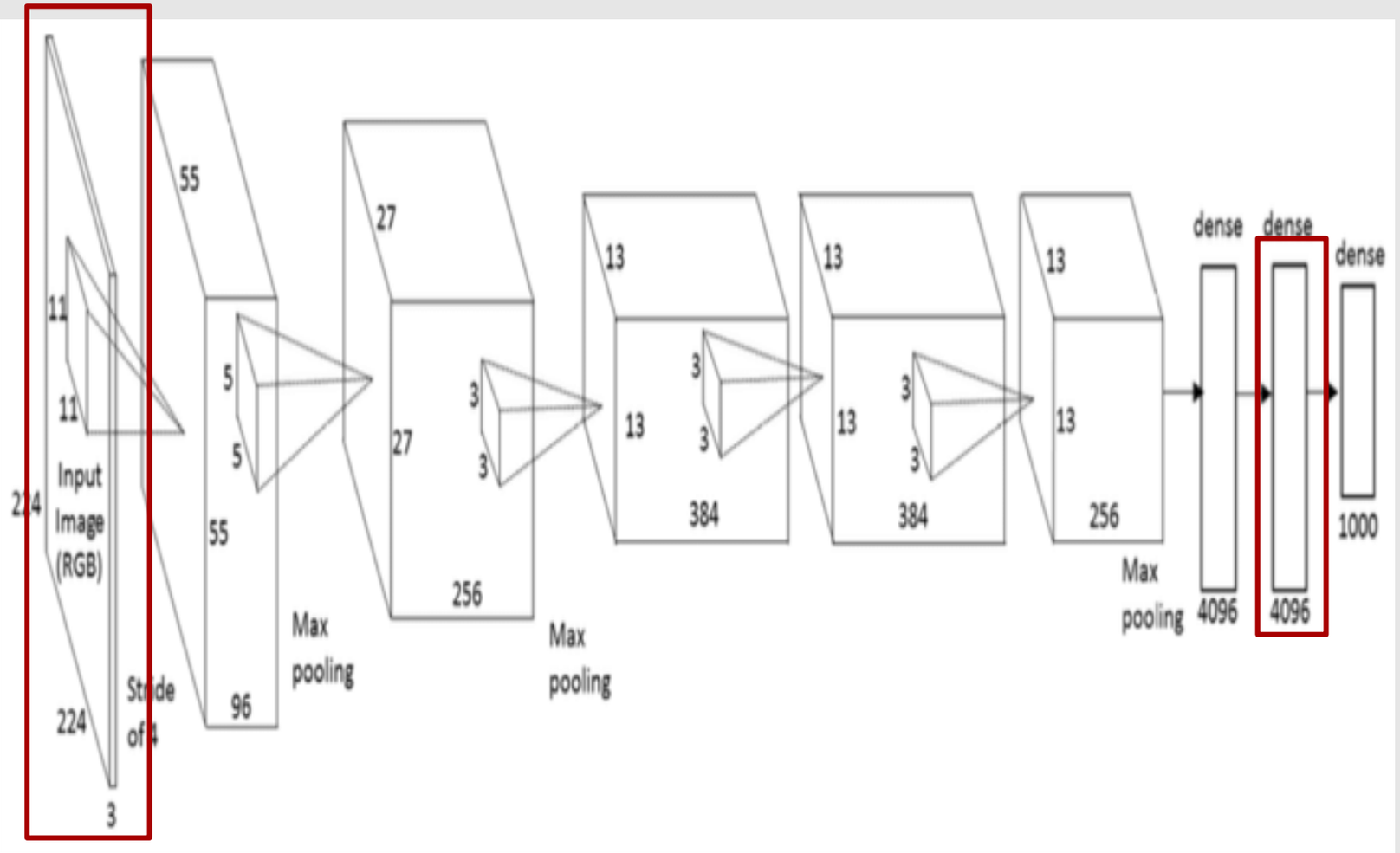
We explore an example of using a pre-trained deep learning image classifier to generate features for use with traditional machine learning approaches to address a problem the original model was never trained on (see the blog post "Image featurization with a pre-trained deep neural network model" for other examples). This approach allows us to quickly and easily create a custom classifier for a specific specialized task, using only a relatively small training set. We use the image featurization abilities of Microsoft R Server 9.1 (MRS) to create a classifier for different types of knots in lumber. These images were made publicly available from the laboratory of Prof., Dr. Olli Silven, University of Oulu, Finland, in 1995. Please note that we are using this problem as an academic example of an image classification task with clear industrial implications, but we are not really trying to raise the bar in this well-established field.

We characterize the performance of the machine learning model and describe how it might fit into the framework of a lumber grading system. Knowing the strengths and weaknesses of the classifier, we discuss how it could be used to triage additional image data for labeling by human experts, so that the system can be iteratively improved.

The pre-trained deep learning models used here are optional components that can be installed alongside Microsoft R Server 9.1; directions are here.

Image Featurization from a Pre-trained Model

Input image



Output features

Activity

Featurizing Images at Scale

Activity

Active learning on images

Microsoft