FFI - April 2019

Learning high capacity models with little labeled data

A bag of tricks to deal with the small labeled dataset problems

Dr. Erlend Aune
Exabel & NTNU

Motivation

Why is this important?

- High capacity models (e.g. nn) can learn most functions with high precision.

- Caveat: We need a large labeled dataset

We want to be able to use the flexibility of high-capacity models in a small data regime

Large labeled datasets rarely available a priori

- If someone hasn't "solved" the problem before -> low probability for suitable (free) "pure" dataset

- 1 (3–4-lb.) corned beef brisket
- 2 lb. russet potatoes, scrubbed, quartered
- 1 1/4 cups extra-virgin olive oil, divided
- Kosher salt, freshly ground pepper
- 1 large green cabbage, cut through the core into 8 wedges
- 4 whole cloves

Recipe dataset

Want annotated data for each line of recipe

- 2 lb. russet potatoes, scrubbed, quartered
  - 2 (measurement)
  - lb (unit)
  - russet potatoes (ingredient)
  - scrubbed, quartered (description)

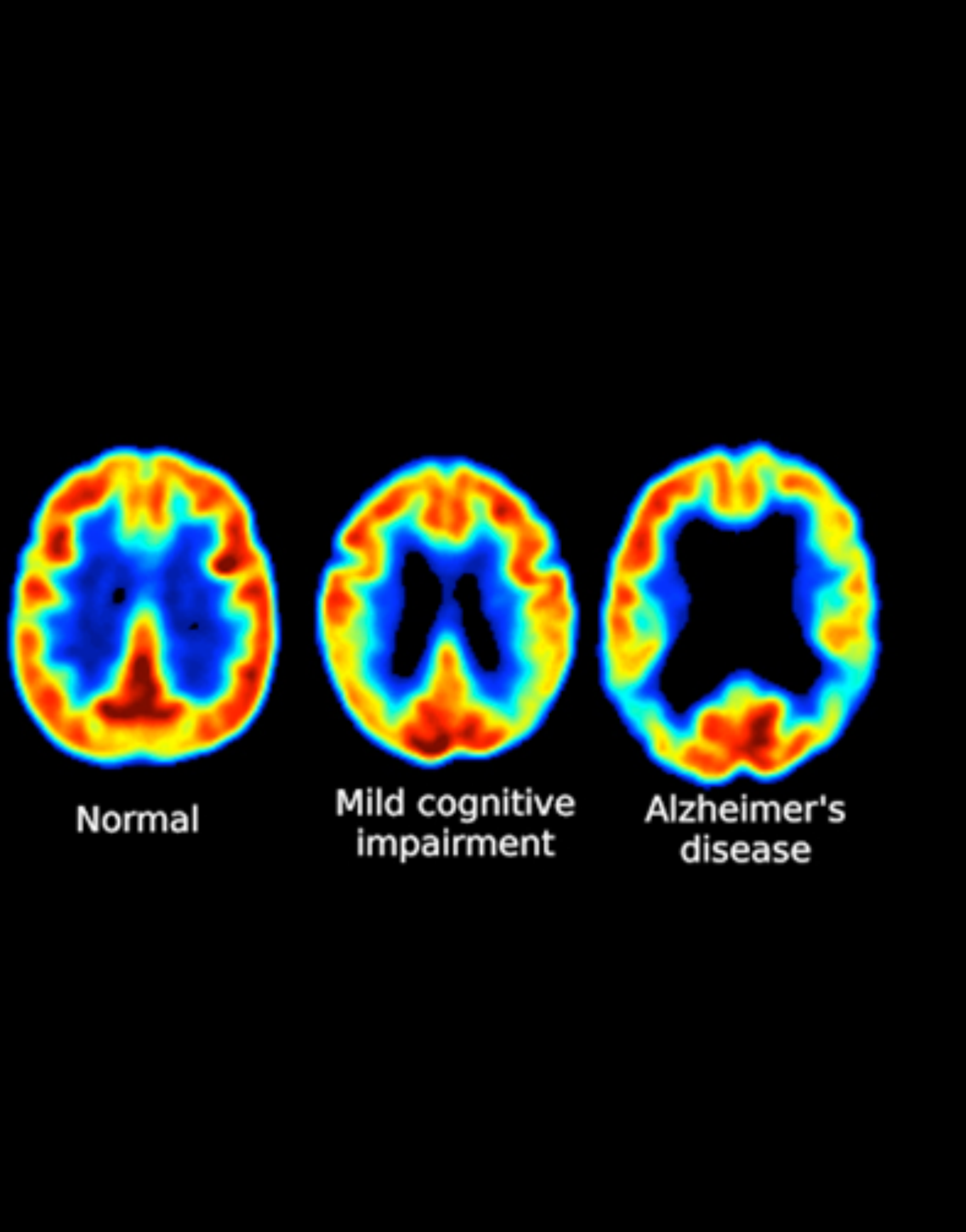
Is there data for such a task? Is it any good?

https://github.com/hhursev/recipe-scrapers

https://github.com/NYTimes/ingredient-phrase-tagger

Labeling may be expensive

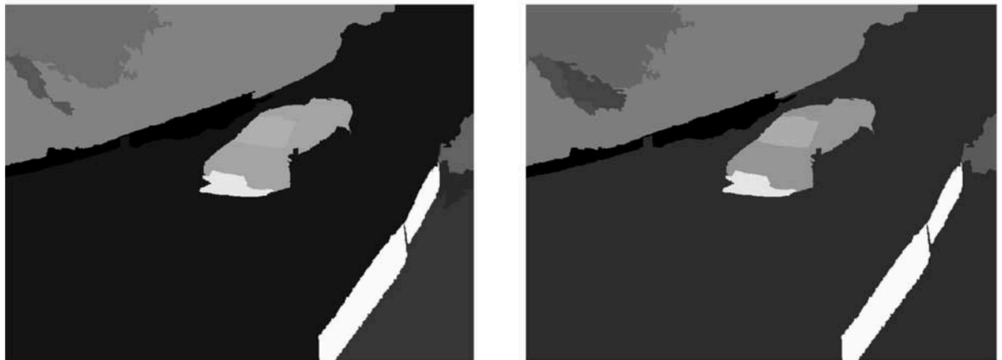
- Medical image annotation requires experts to annotate the images
- Images may be expensive to produce in the first place
- The process may be intrusive



Large dataset exists, but not good enough

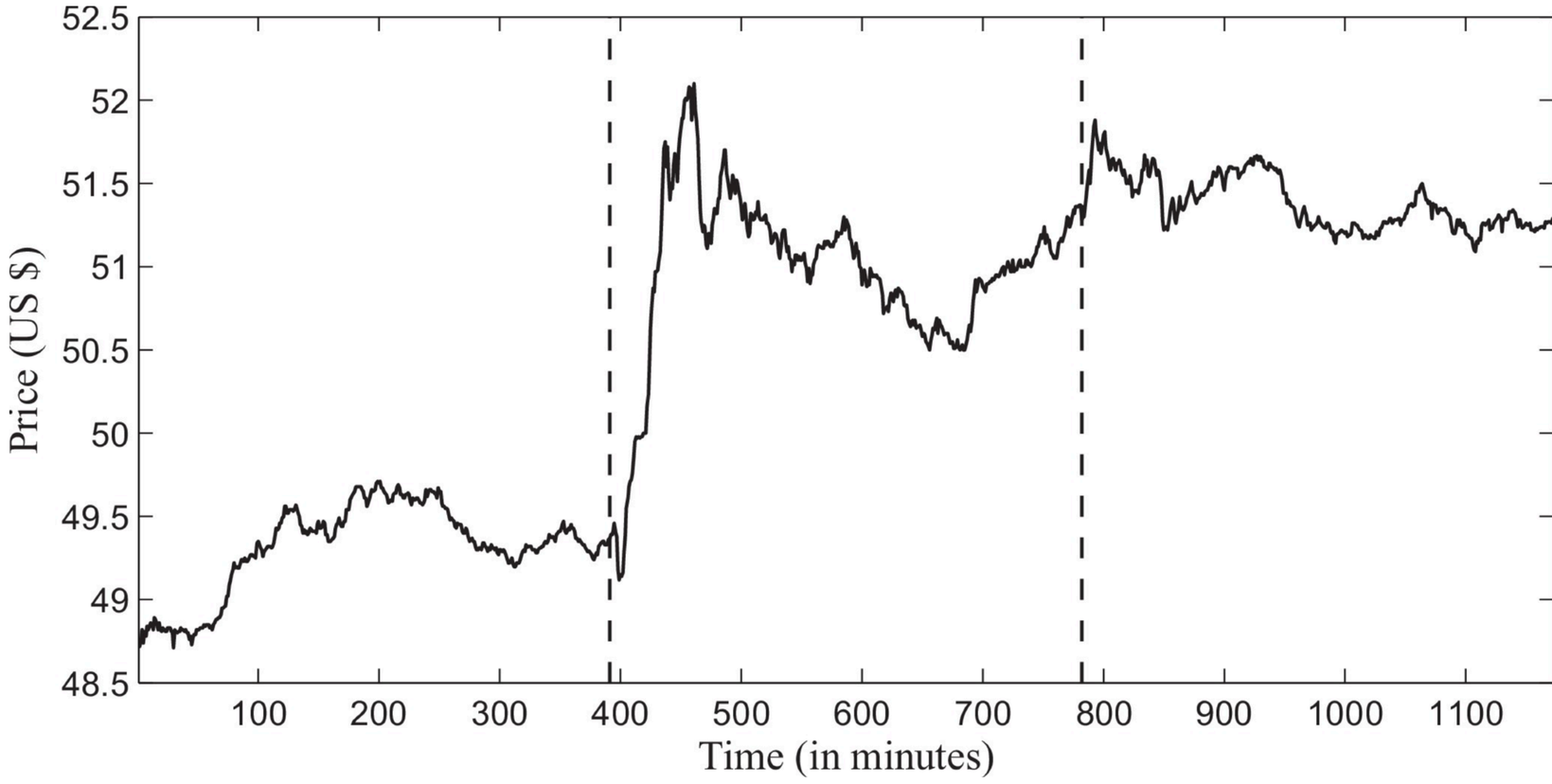
- Model gives reasonable results
- But not good enough for production systems
- *More of a modeling problem*

(a) (b)



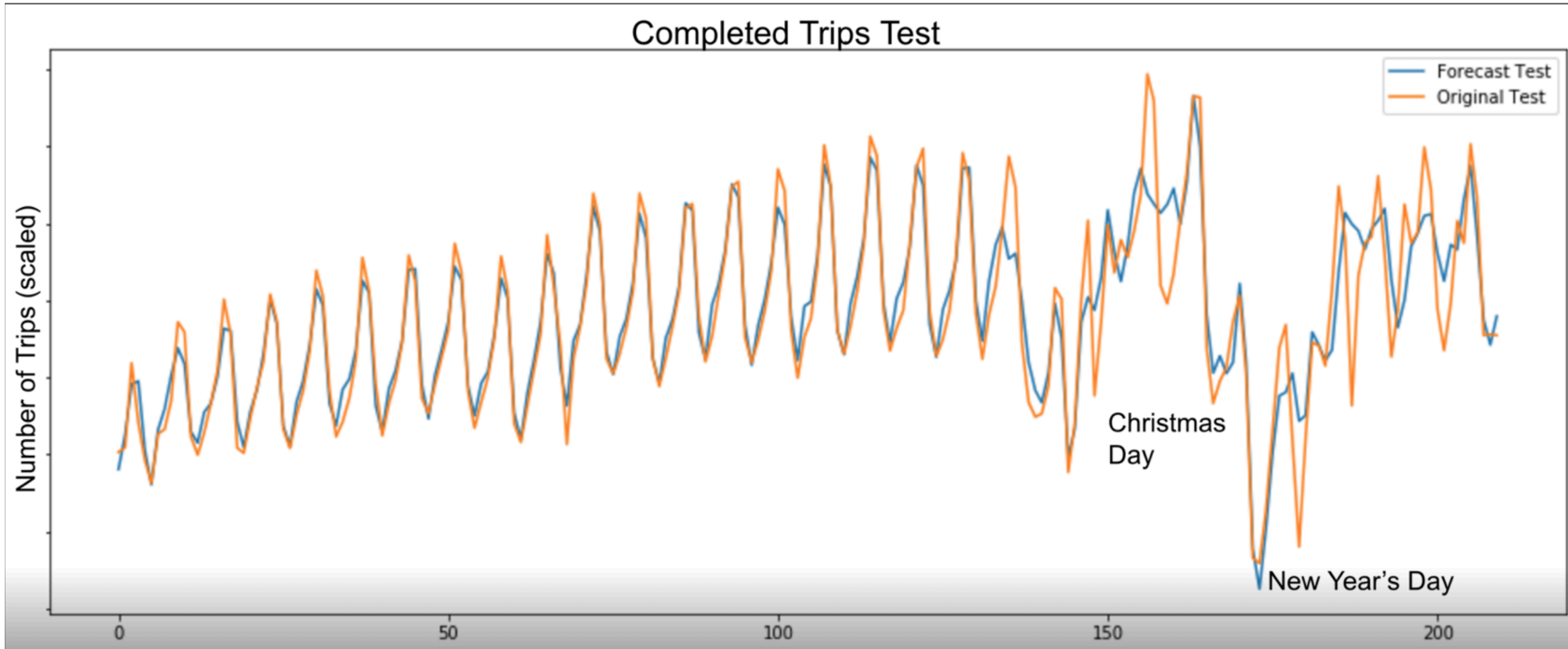
We only have a few positive samples

- Rare event detection
- Changepoints in time series



Know that some days are important

- Few such days
- How to use this information accross many time series?
- https://eng.uber.com/neural-networks/



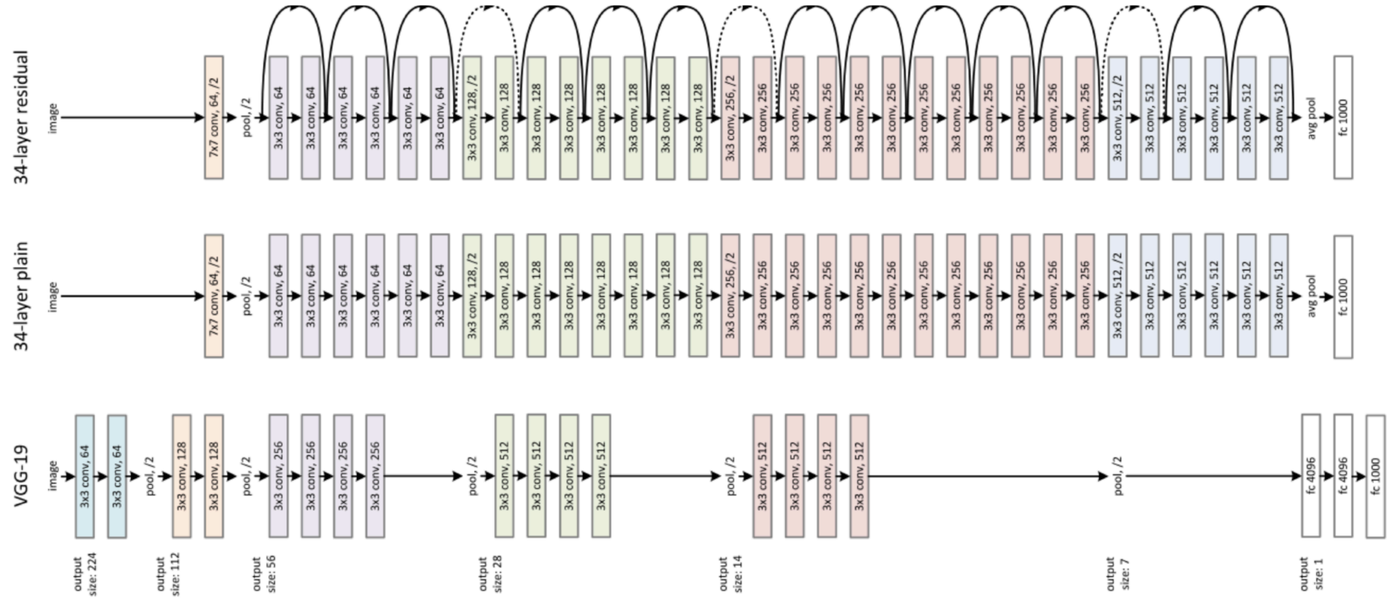
Plan for seminar

- High level overview of common ways of addressing little training data
- What approach is appropriate in different data scenarios
- Exercises to accompany some of the ideas

Good modeling is a must

Nothing beats good modeling! Good modeling: A prerequisite for making the most of these ideas.

I.e. If you are dealing with images, convnets *make sense*, but we may still have too little data.



But what if my inputs are sets, or sequences, etc...

Chances are that there is research that deals with the type of input you are interested in.

Example: What is the sum of...

https://arxiv.org/abs/1703.06114

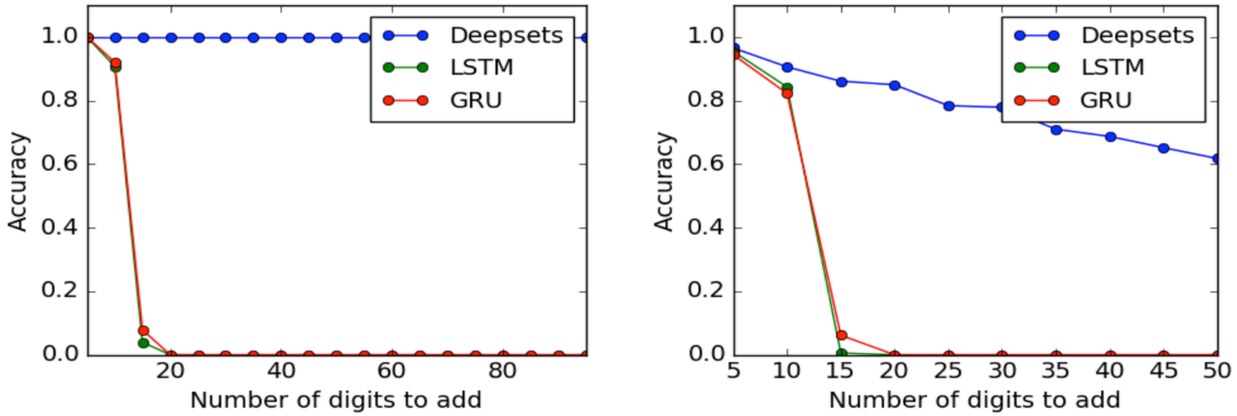
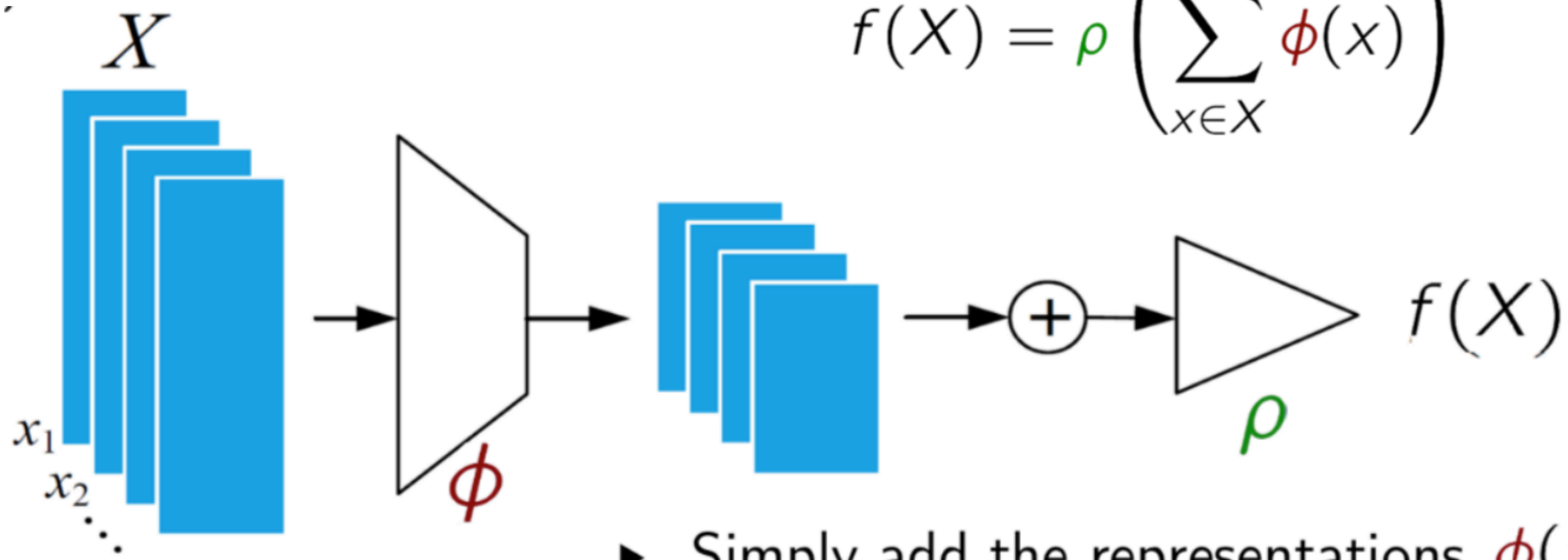


Figure 2: Accuracy of digit summation with text (*left*) and image (*right*) inputs. All approaches are trained on tasks of length 10 at most, tested on examples of length up to 100. We see that DeepSets generalizes better.

Deep sets

https://arxiv.org/abs/1703.06114

$$f(X) = \rho\left(\sum_{x \in X} \phi(x)\right)$$



- Simply add the representations $\phi(x)$
- Sum is processed by ρ network

Data scenarios

For small amounts of labeled data

First scenario

We only have a small amount of labeled data points

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

First scenario

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

Relevant questions:

1. Has anyone done research on a model that fits this type of data?

2. Is there a “similar” dataset out there?

Second scenario

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

$K \gg N$ unlabeled datapoints : $\{x_i^U, \ldots, x_K^U\} = U$

Second scenario

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

$K \gg N$ unlabeled datapoints : $\{x_i^U, \ldots, x_K^U\} = U$

Natural questions:

1. Can I use data in *U* to help modeling points in *L*?

2. If I want to extend *L*, how should I do that?

Third scenario

Streaming data:

New unlabeled data points arrives sequentially.

Example: Customer interactions

When should a chatbot hand over the interaction to an oracle (customer support agent)?

Ways to handle the scenarios

For small amounts of labeled data

First scenario

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

Two natural candidates

- Transfer learning/multitask learning
- Data augmentation

Transfer learning

A “hello world” of transfer learning

Train on imagenet...

...then fine-tune

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

Transfer learning code examples

Keras: https://keras.io/applications/#fine-tune-inceptionv3-on-a-new-set-of-classes

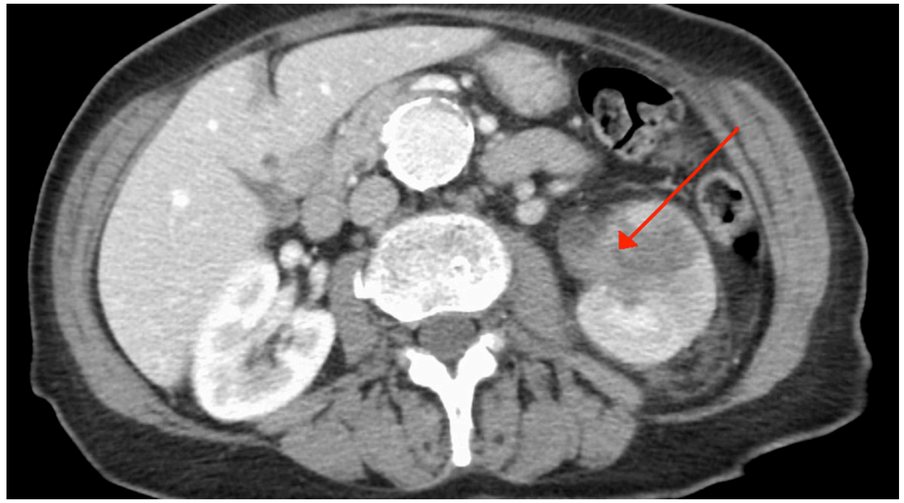
R: https://www.r-bloggers.com/transfer-learning-with-keras-in-r/

PyTorch: https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html

Main idea: Freeze most layers preceeding the input, then train the last parts. Unfreeze, and train with low learning rate.

A “hello world” of transfer learning???

Train on imagenet...



...then fine-tune

It doesn’t always work

I want to rank glyphs by aesthetic preferences.

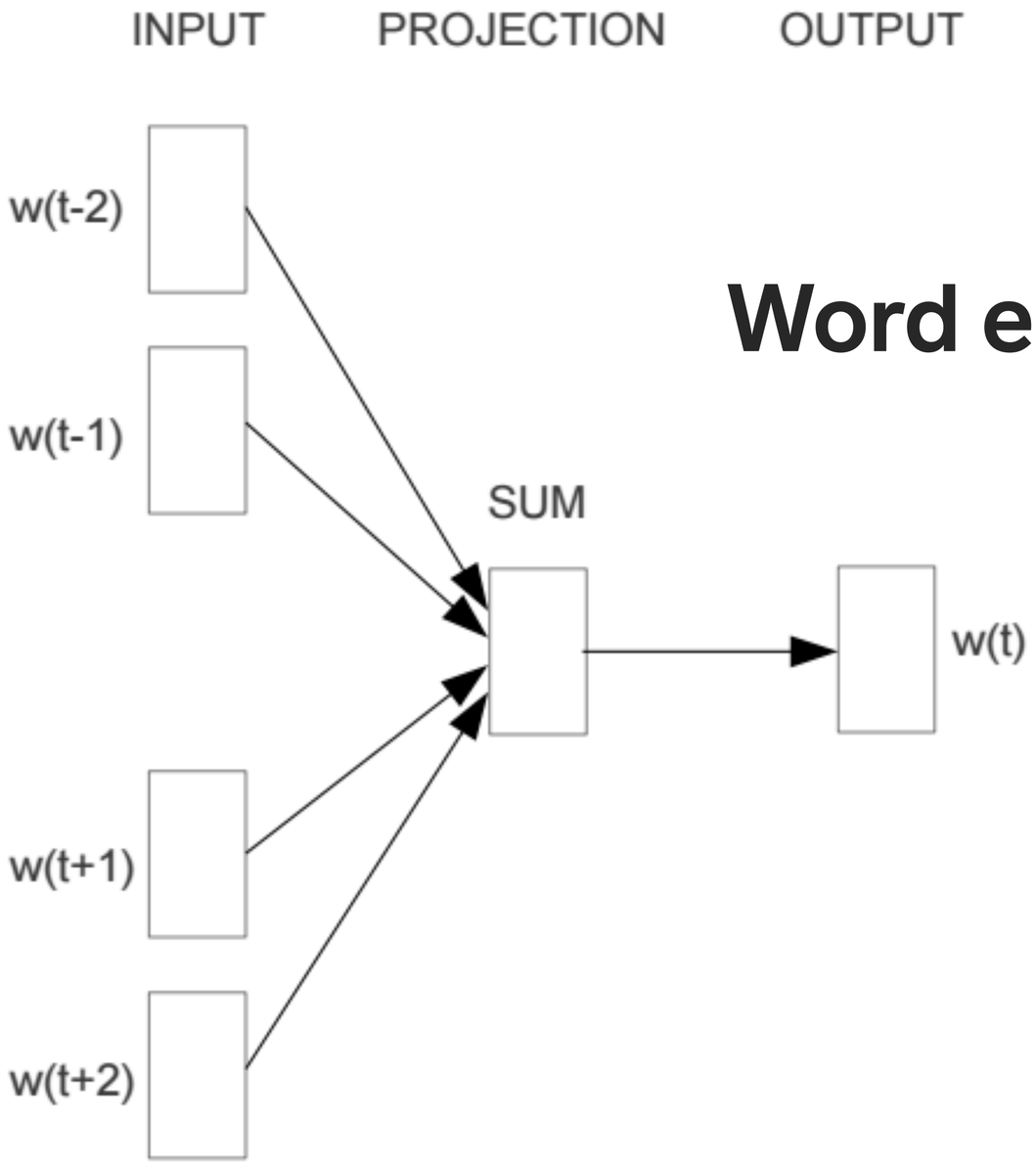
Training data: 1400 images

Can I use QuickDraw (15 M images)

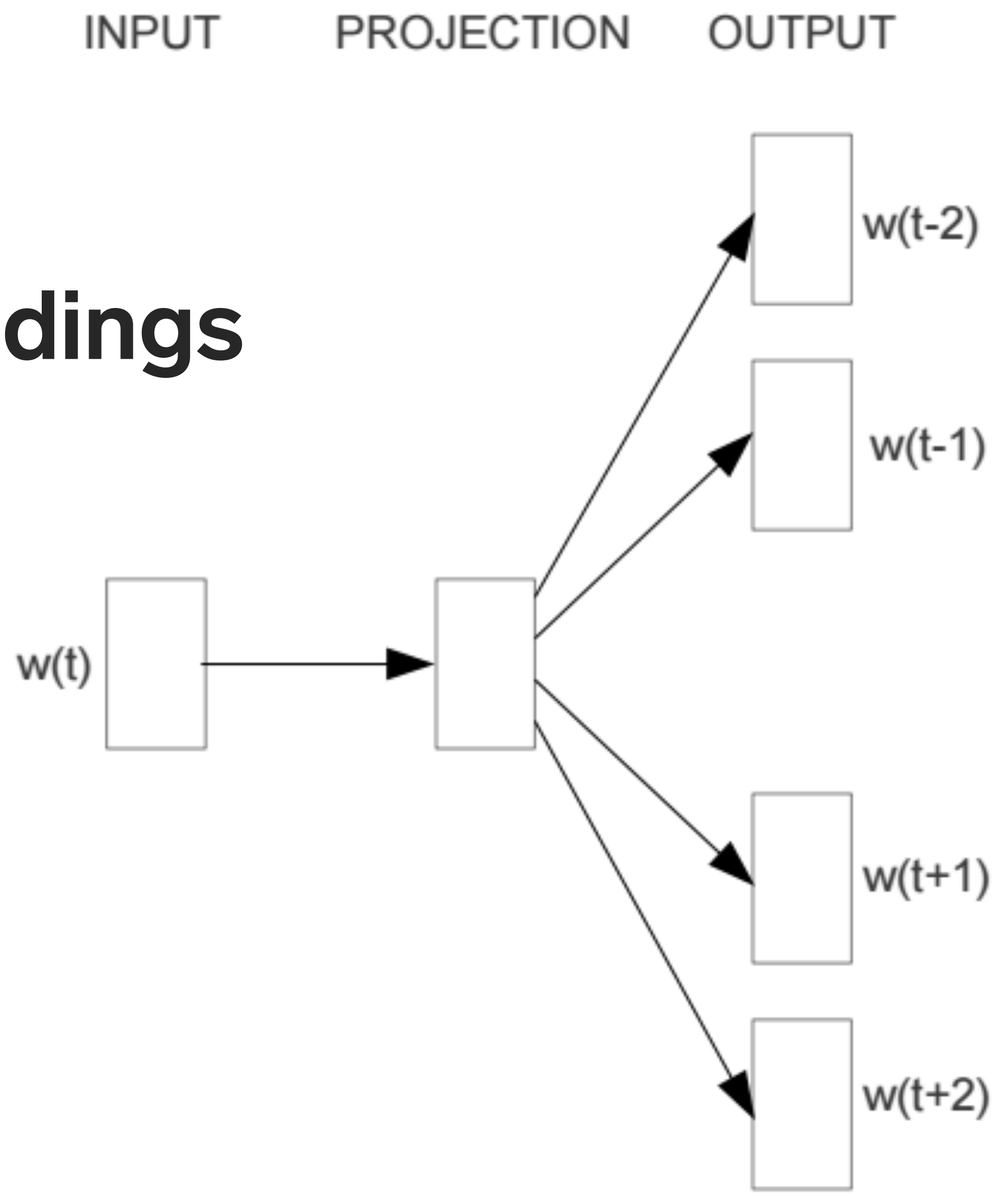
Does pre-training on QuickDraw help this task?

?

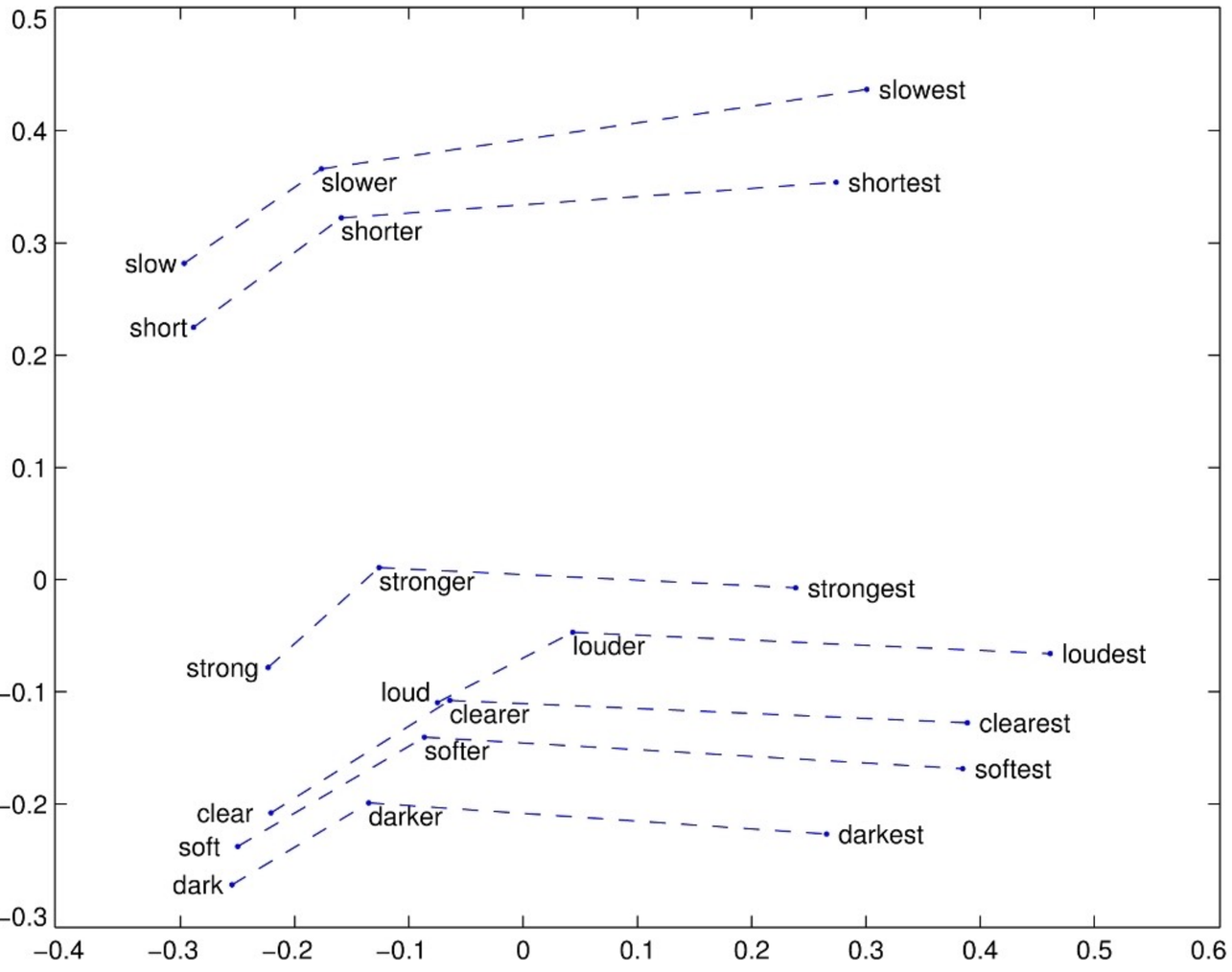
Word embeddings

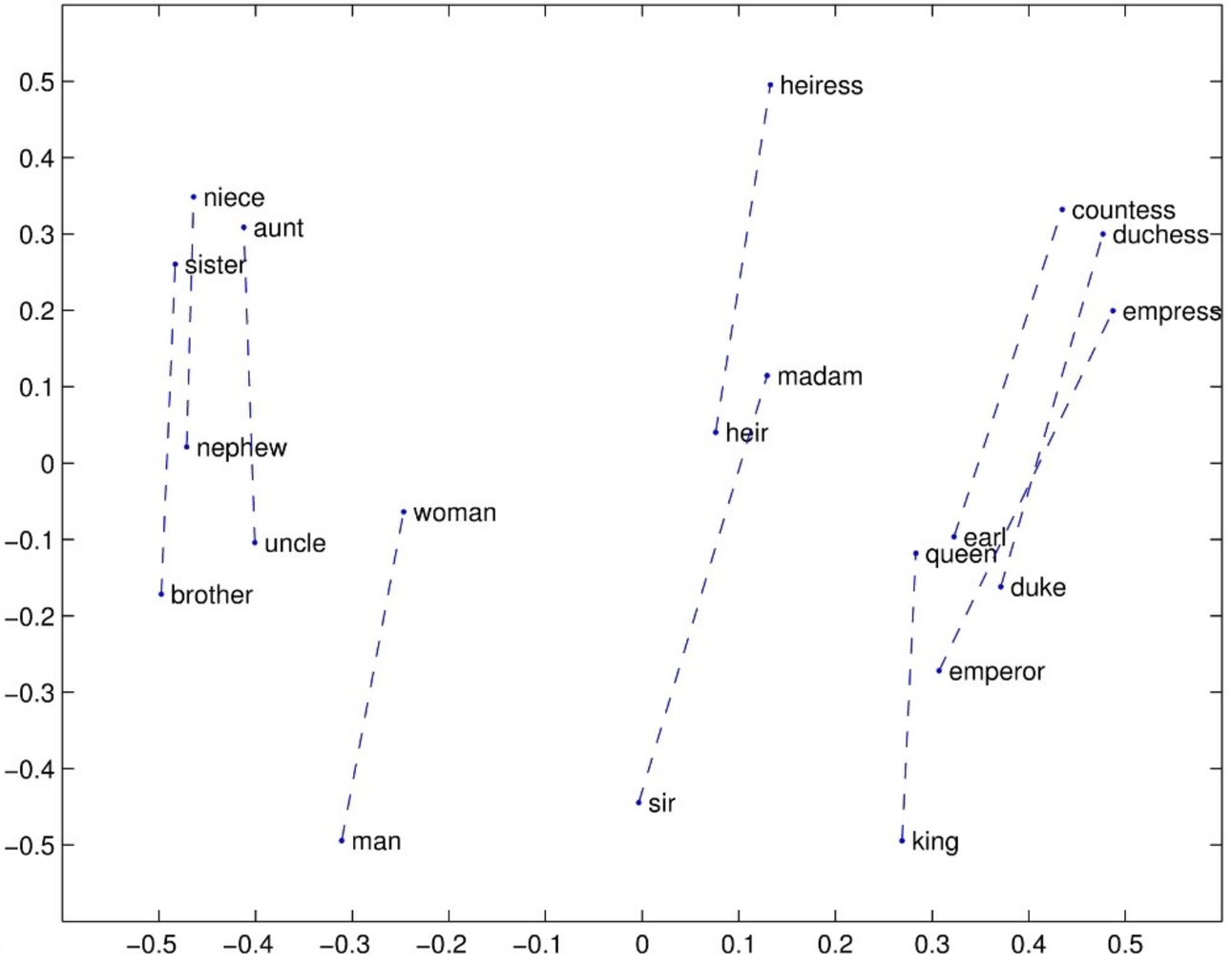


CBOW



Skip-gram


slowest
slower
shortest
shorter
slow
short
stronger
strongest
strong
louder
loudest
loud
clearer
clearest
softer
softest
clear
darker
soft
darkest
dark


heiress
niece
countess
aunt
duchess
sister
empress
madam
heir
nephew
woman
uncle
earl
queen
brother
duke
emperor
sir
man
king

Commonly used: Glove

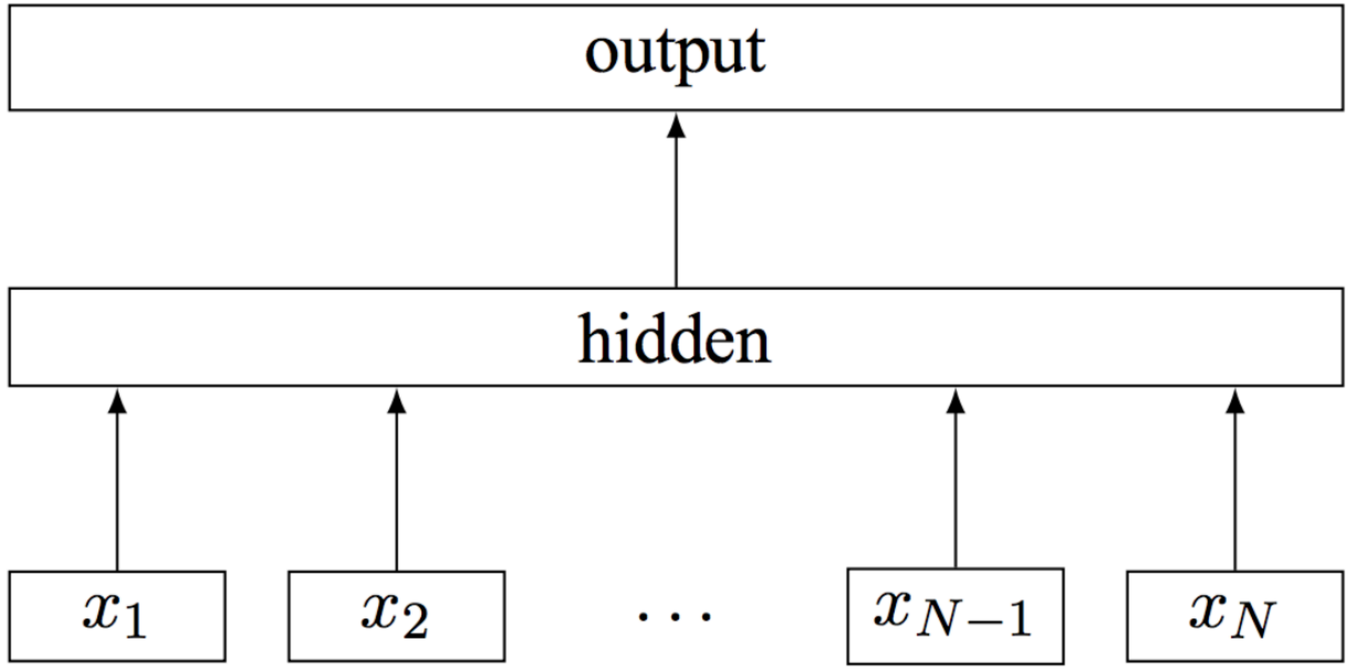
https://nlp.stanford.edu/projects/glove/

We initialise word embeddings with pre-trained weights.

Then use a standard architecture.

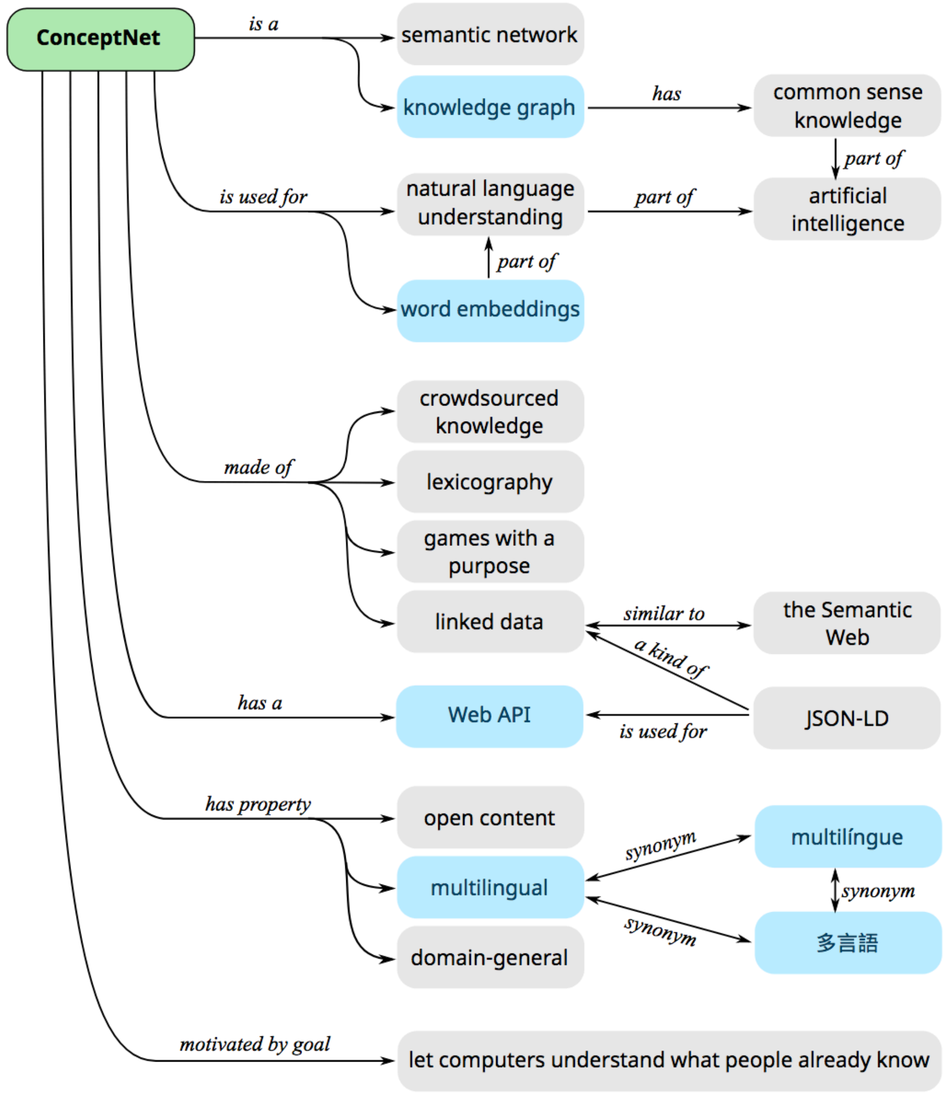
Russian word-vectors:

https://rusvectores.org/en/models/



Word embeddings

- Enriching Word Vectors with Subword Information
- Better embeddings with auxiliary information



ConceptNet → Enriching Word Vectors with Subword Information →

Transfer learning for text: BERT

BERT achieved SOTA on many NLP tasks using unsupervised pretraining.

https://arxiv.org/abs/1810.04805

https://github.com/huggingface/pytorch-pretrained-BERT

- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

Transfer learning recap

BERT

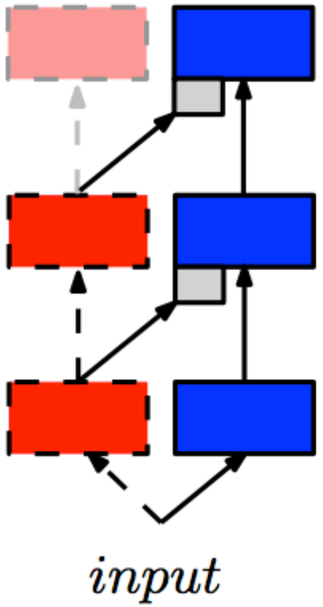
Transfer learning is

- Relatively easy

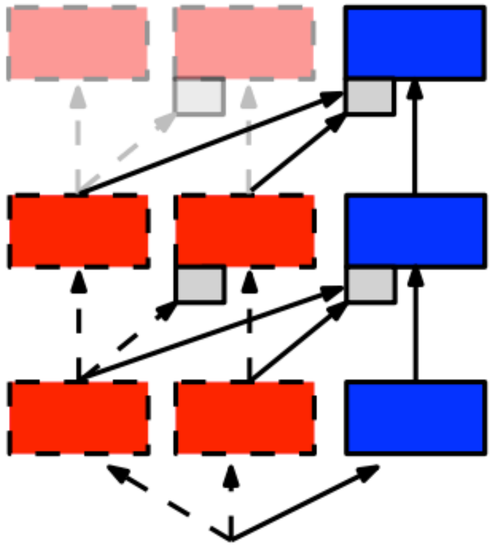
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

- Potentially boosts performance massively
- Creativity in creating the “auxiliary task” is important!
- The pre-trained model contains prior information about the world.

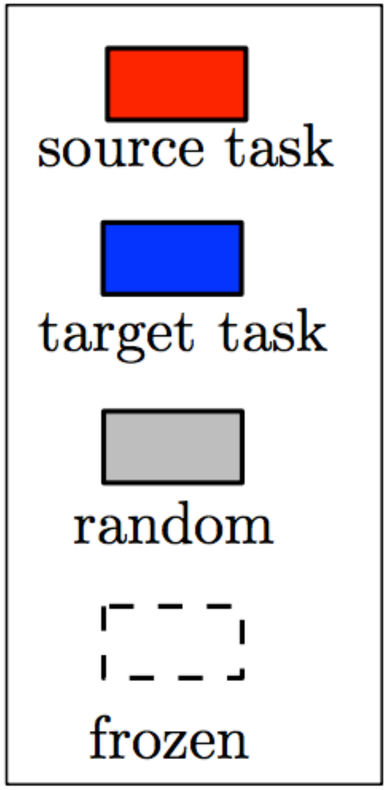
Adding capacity instead of fine-tuning



Progressive Net
2 columns



(6) Progressive Net
3 columns



Progressive Neural Networks →

Data Augmentation

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

Data Augmentation

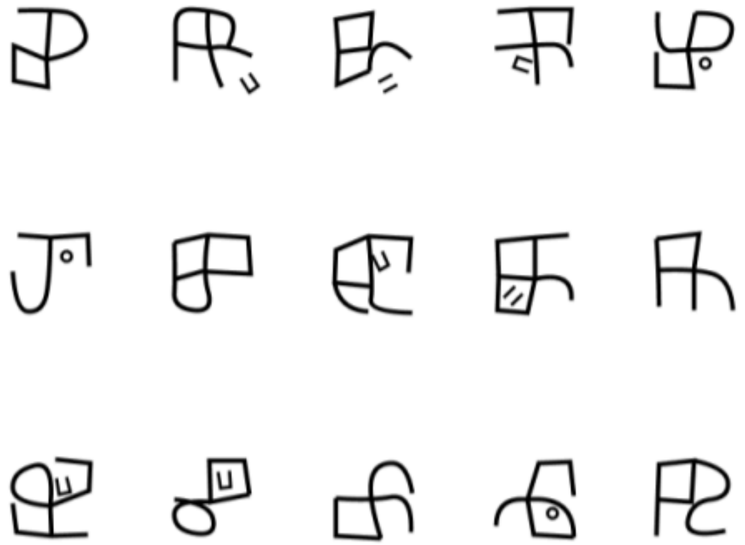
Create new datasets by bootstrapping/introducing noise in the primary dataset.

https://github.com/albu/albumentations

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

Data Augmentation

Important: Augmentations must not destroy features you want to detect.



Rotations != 90 degrees

-> bad results

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

Data Augmentation - text

Not many libraries out there.

Synonym replacement and sentiment libraries. https://wordnet.princeton.edu/

What about translating back and forth using google translate?

This film was just brilliant casting location scenery story direction everyone's really suited the part they played and you could just imagine being there robert. He is an amazing actor and now the same being director.

This film was just a wonderful scene scene scene scene where everyone fits the role they played and you can imagine that there is Robert. He is a wonderful actor and now he himself is a director.

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

Data Augmentation - using the web

What if you can link you texts/images to WikiPedia?

Image/text search on google...

Second scenario

Small labeled dataset, large unannotated dataset from same domain

Second scenario

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

$K \gg N$ unlabeled datapoints : $\{x_i^U, \ldots, x_K^U\} = U$

Natural questions:

1. Can I use data in *U* to help modeling points in *L*?

2. If I want to extend *L*, how should I do that?

Active Learning (pool based)

Labeling data from unannotated dataset

Active Learning

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

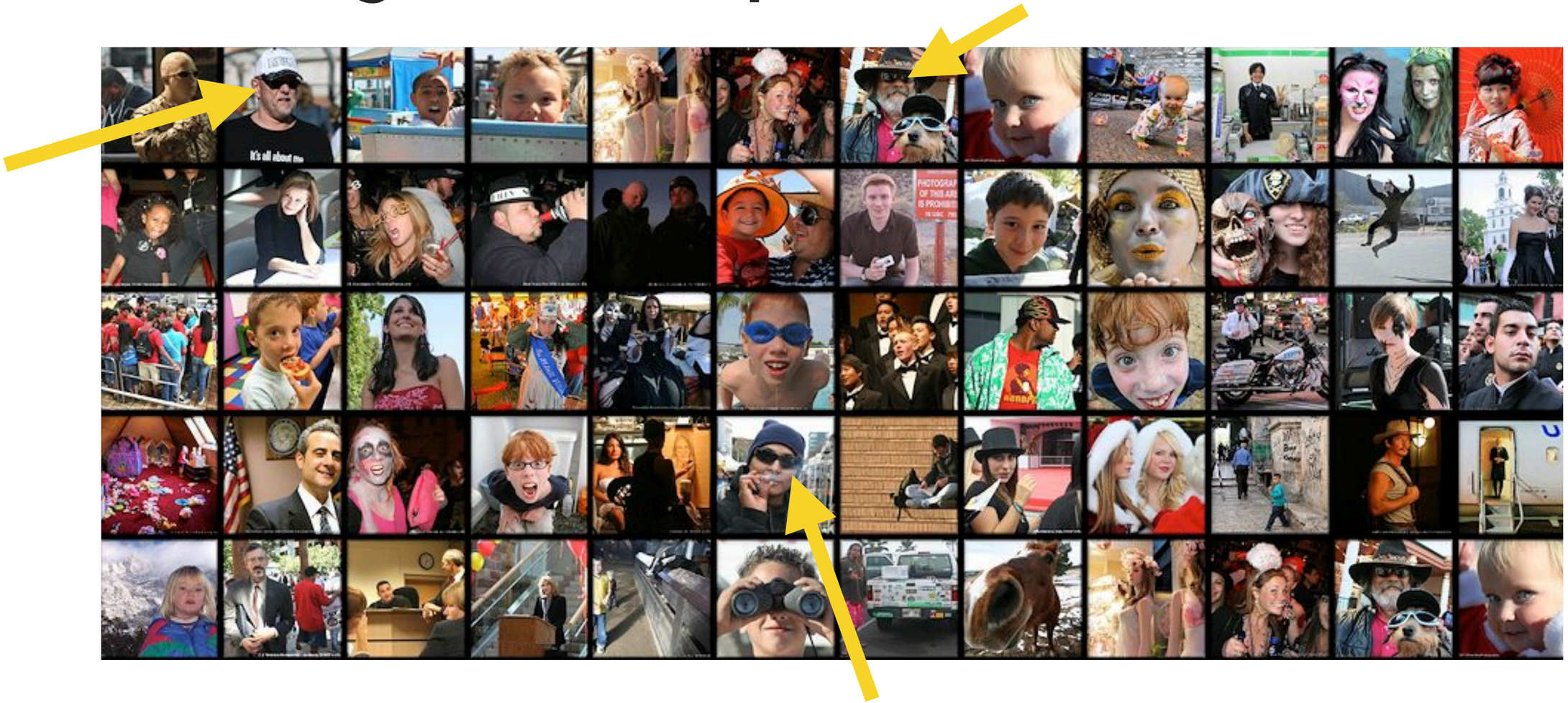
$K \gg N$ unlabeled datapoints : $\{x_i^U, \ldots, x_K^U\} = U$

Natural questions:

1. Can I use data in *U* to help modeling points in *L*?

2. If I want to extend *L*, how should I do that?

Choosing which samples to label



To identify people with sunglasses, which samples will yield the best model?

Active Learning - two scenarios

Scenario 1: Add labels from ***U*** to ***L.*** Pool based Active learning

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

$K \gg N$ unlabeled datapoints : $\{x_i^U, \ldots, x_K^U\} = U$

Scenario 2

New data arrives in a stream:

- Make a prediction
- Ask an oracle

Active Learning

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

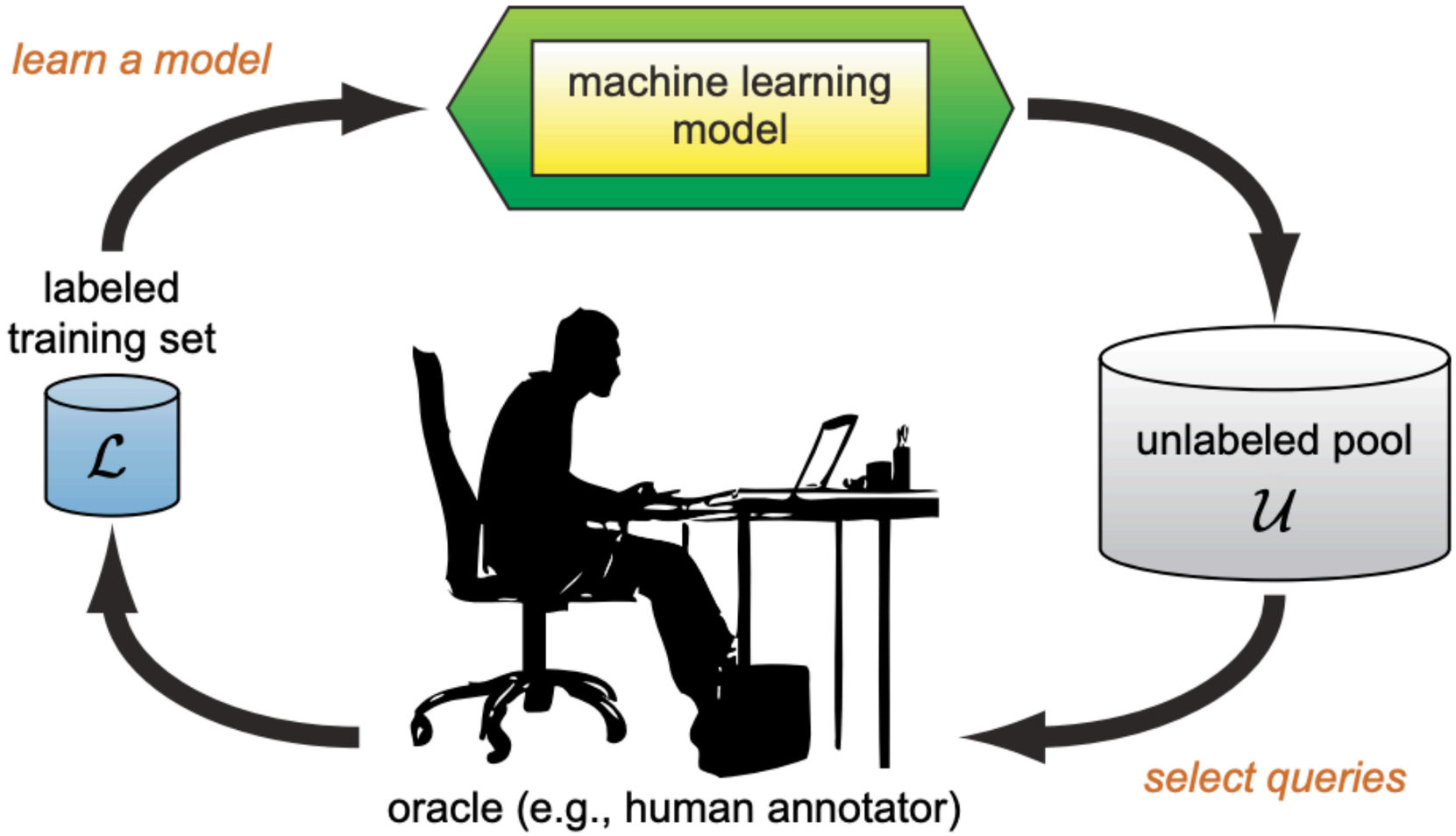
$K \gg N$ unlabeled datapoints : $\{x_i^U, \ldots, x_K^U\} = U$

What is the baseline?

Let $L_0 = L, U_0 = U$

Let $l \sim \text{Uniform}(0, \#U_j)$, $L_j = L_{j-1} \cup \{x_l^U\}$, $U_j = U_{j-i} - \{x_l^U\}$

Pool based AL



Pool based AL

- Let $\{x_j^i\} \subset f(U_i) \subset U_i$. f is then the query strategy
- Ask oracle for label for $\{x_j^i\} \forall j \in [1,k]$
- Let $L_i = L_{i-1} \cup \{x_j^i\}_{j=1}^k$
- Retrain model with $m_i = m(L_i)$

How does ***p(L$_i$)*** improve using that query strategy?

Performance measures

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}}$$

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}}$$

$$F_1 = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

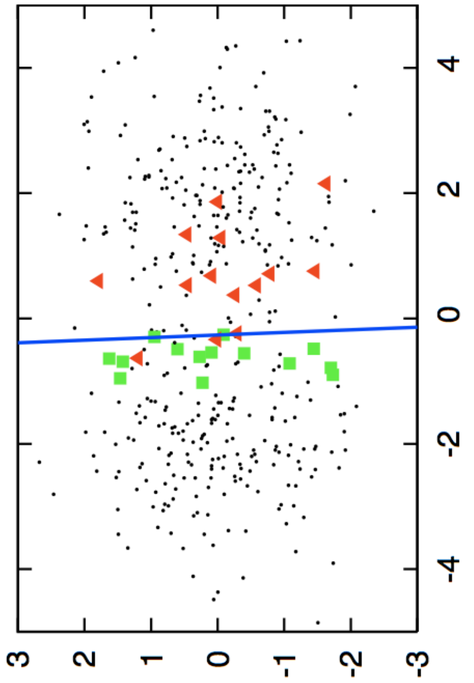
What do we want?

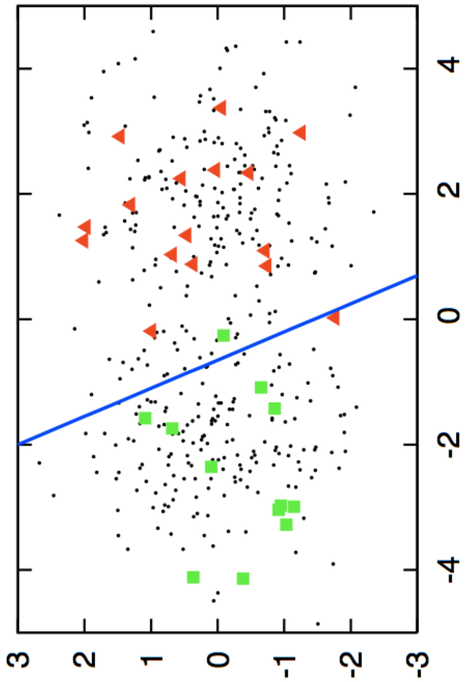
Let ***m(L)*** be a model with performance ***p(L)***, and let ***U*** is the set of unlabeled samples, and let

$$S_k^i \subset U \text{ with } \text{card}(S_k^i) = k, \cup_i S_k^i = U$$

Find

$$S_k^* = \arg\max_i p(L \cup S_k^i)$$





Questions

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

$K \gg N$ unlabeled datapoints : $\{x_i^U, \ldots, x_K^U\} = U$

- Can we find S_k^* ?
- What are useful criteria for adding a point to ***L***?

Reminder - motivation

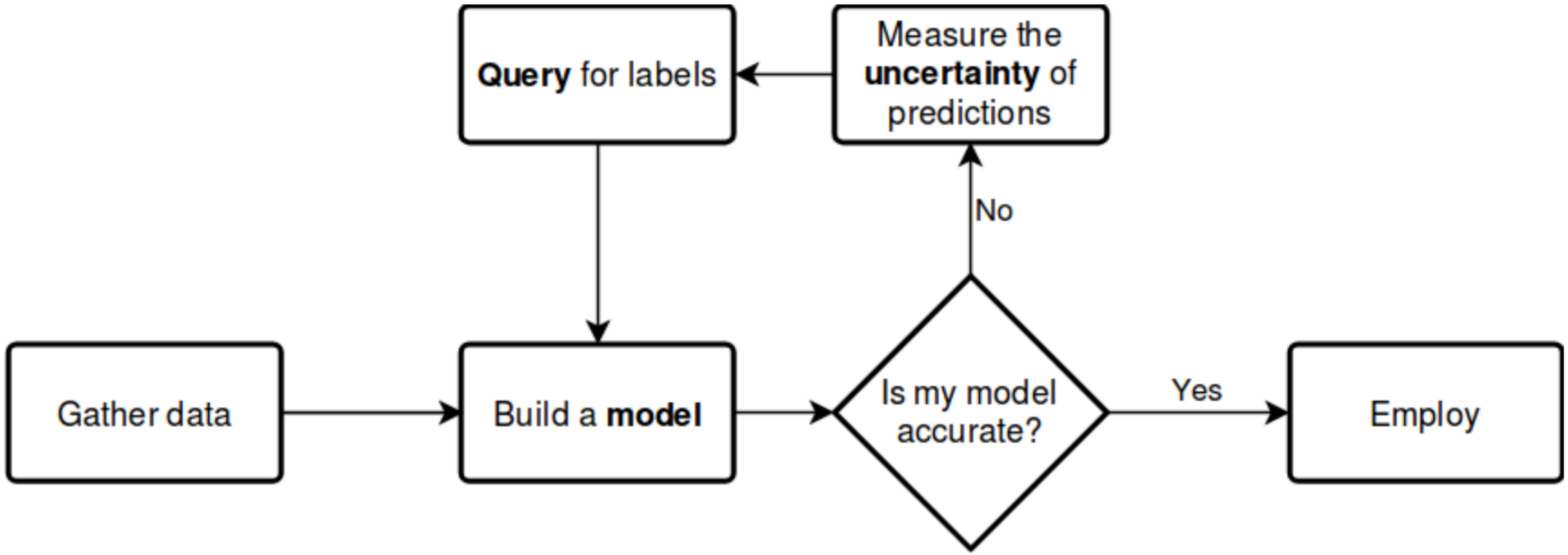
The government threw a wrench into Broadcom's plans for a hostile takeover of the U.S. tech giant Qualcomm on Sunday, pushing Qualcomm to delay a shareholder meeting and board of directors election this week. The Committee on Foreign Investment in the United States (CFIUS) is looking into whether the Broadcom's bid, which would be the largest tech deal in history if it goes through, threatens national security.

Time consuming and expensive to annotate

AL workflow

N labeled datapoints : $\{x_i^L, \ldots, x_N^L\} = L$

$K \gg N$ unlabeled datapoints : $\{x_i^U, \ldots, x_K^U\} = U$



Uncertainty sampling

Main idea:

Let $\{\hat{y}_j\} = \{m(x_j)\}_i \forall x_j \in U_i$

Choose to label ***k*** samples: the ***y***'s where ***m*** is most uncertain its prediction.

Classification: Two uncertainty measures

Choose the label where the model is least confident about the preferred class

$$\hat{y}_j = \arg\max_y P(y \mid x_j)$$

$$x_j^i = \arg \max_{x \in U_i - \{x_k\}_{k=0}^{j-1}} (1 - P(\hat{y} \mid x))$$

Alternative: Margin sampling

$$\hat{\hat{y}}_j = \arg \max_{y \in Y - \hat{y}} P(y \mid x_j)$$

$$x^* = \arg\min_x (P(\hat{y} \mid x) - P(\hat{\hat{y}} \mid x))$$

Exercise: Which one performs best on MNIST?

Take a subset of 20 digits of the digits 8, 3 and 9 in MNIST.
Use margin sampling or least confident sampling in https://github.com/modAL-python/modAL,

https://modal-python.readthedocs.io/en/latest/content/apireference/uncertainty.html

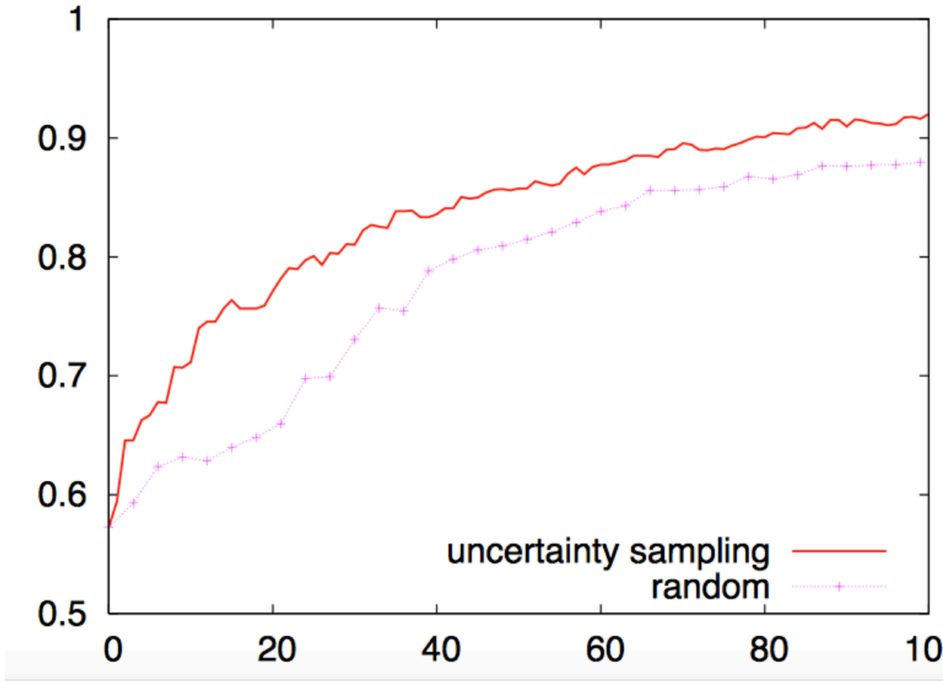
Which performs best?

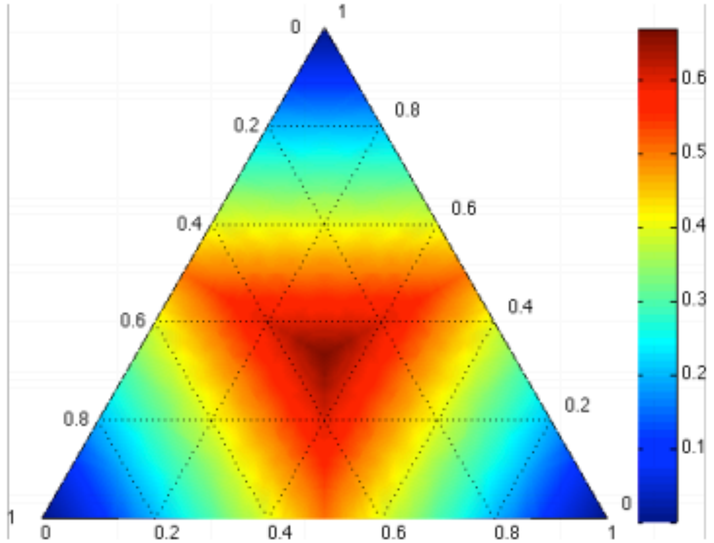
Entropy sampling

Use a weighted sum of all classification scores

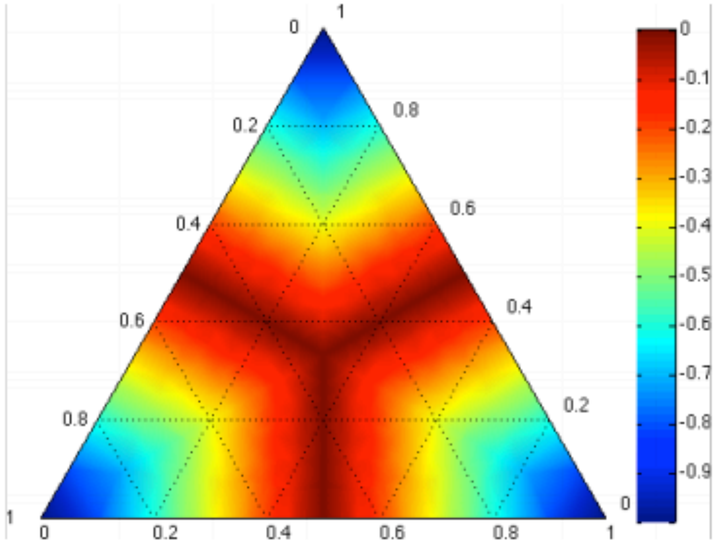
$$x_H^* = \underset{x}{\operatorname{argmax}} - \sum_i P_\theta(y_i|x) \log P_\theta(y_i|x),$$

This is a popular choice.

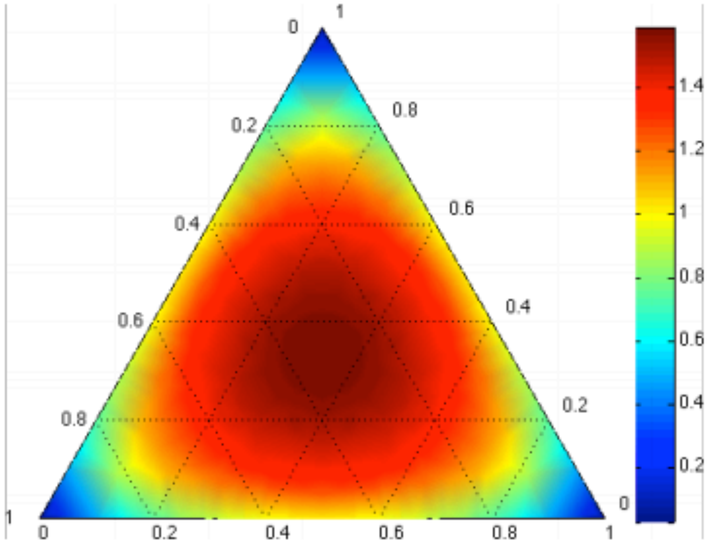




(a) least confident



(b) margin



(c) entropy

Query by Committee

Idea: Have ***n*** models, each performing predictions.

Models can be constructed in an arbitrary way (boosting, bagging, qualitative criteria, sampling)

Should be able to learn different aspects of data-label distribution.

Query by Committee

Metrics for model disagreement:

$$x^*_{VE} = \underset{x}{\text{argmax}} - \sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}$$

Expected Gradient Length

Finding the ***x*** that changes the model the most is an alternative strategy.

$$x^*_{EGL} = \underset{x}{\operatorname{argmax}} \sum_i P_\theta(y_i|x) \left\| \nabla \ell_\theta(\mathcal{L} \cup \langle x, y_i \rangle) \right\|$$

$$\nabla l_\theta(\mathscr{L} \cup \langle x, y_i \rangle) \approx \nabla l_\theta(\langle x, y_i \rangle)$$

What gradients to look at?

It may be advantageous to look at gradients at only parts of the model.

E.g. gradients of embedding layers in text problems.

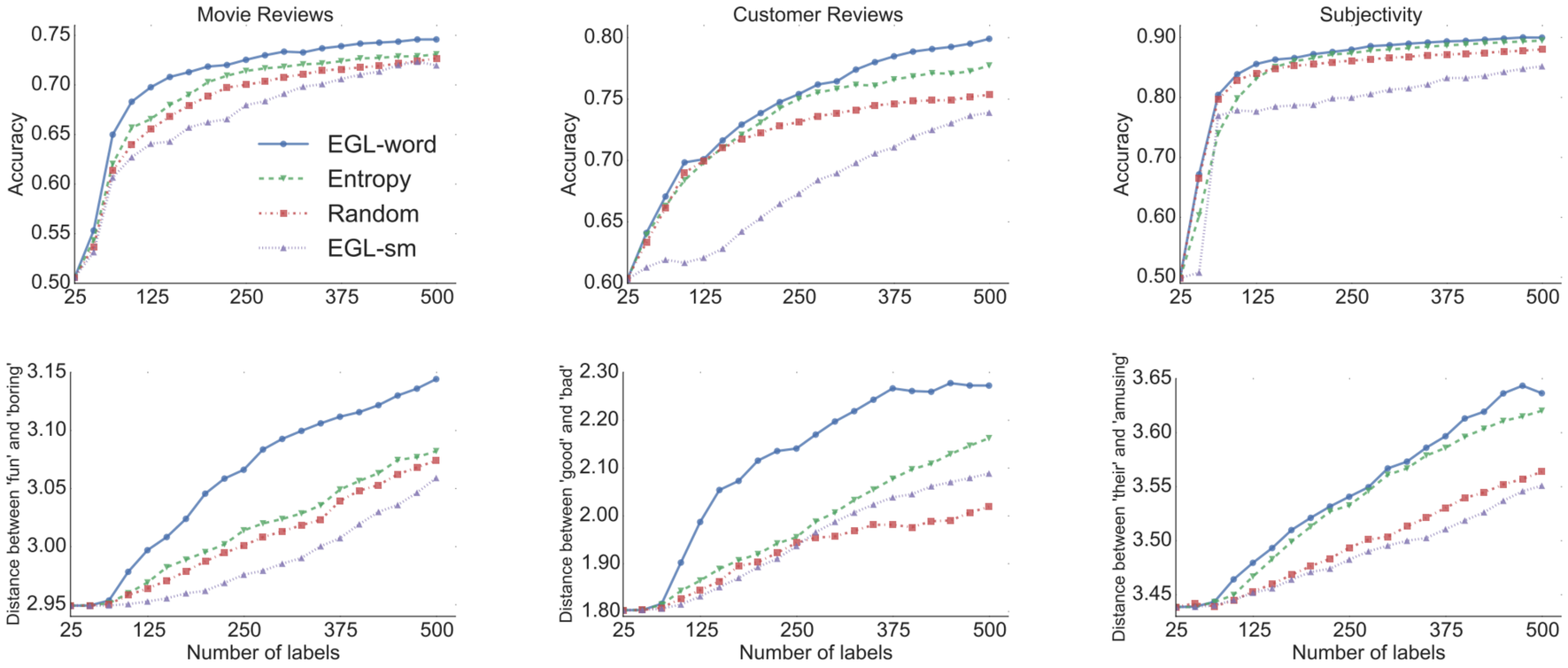
https://github.com/Froskekongen/MA8701/blob/master/gradients/keras_gradients.py

$$x^*_{EGL} = \underset{x}{\operatorname{argmax}} \sum_i P_\theta(y_i|x) \left\| \nabla \ell_\theta(\mathcal{L} \cup \langle x, y_i \rangle) \right\|$$

$$\nabla l_\theta(\mathscr{L} \cup \langle x, y_i \rangle) \approx \nabla l_\theta(\langle x, y_i \rangle)$$

Expected Gradient Length (EGL)

Maximize EGL in (word) embeddings



Active Discriminative Text Representation Learning →

Easy-to-use python library: modAL

We will go deeper into active learning later

https://github.com/modAL-python/modAL

Density weighting

- Main idea: Augment query strategy by representing the data distribution.

$$x_{ID}^{*} = \underset{x}{\operatorname{argmax}}\, \phi_A(x) \times \left(\frac{1}{U} \sum_{u=1}^{U} \operatorname{sim}(x, x^{(u)}) \right)^{\beta}$$

Discussion

What are the main issues with the strategies we have looked at?

- Training time?
- Data set sizes?

Does it make sense to always query the most uncertain sample?

Main challenges

- Querying outliers
- Intrinsic uncertainty in data
- Adding one label at the time before retraining

Batch-mode AL

Get more than one sample in query.

Retraining after one-sample query is potentially very different from querying for a large batch using the same criterion.

Question: How to induce diversity in a batch?

- Explicit modeling: https://papers.nips.cc/paper/3295-discriminative-batch-mode-active-learning.pdf

- Heuristics

Diverse mini-batch AL

Simple strategy in: https://arxiv.org/abs/1901.05954

Let s_i be the query score for sample i.

Minimize k-means objective:

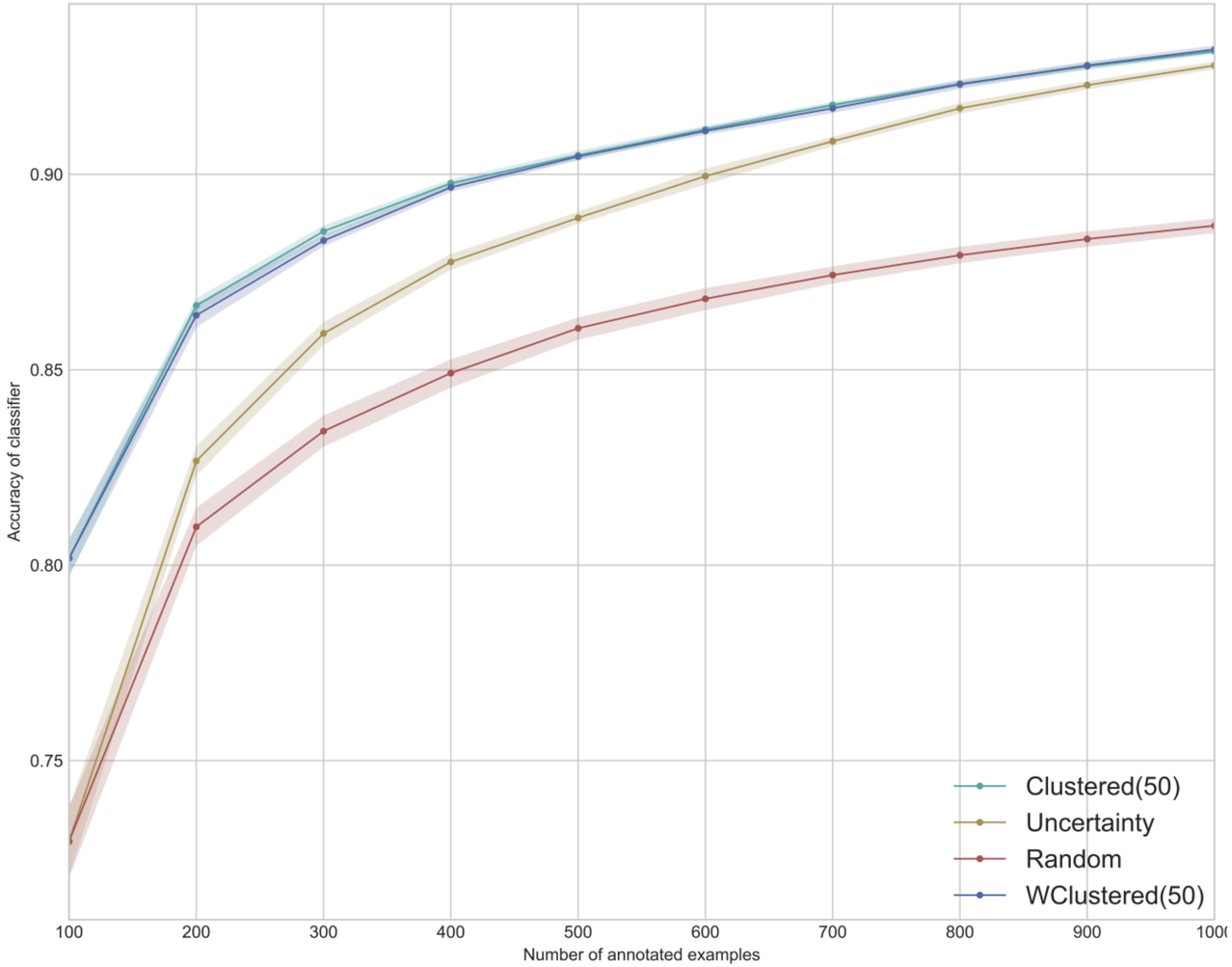
$$\sum_{x_i \in \mathcal{X}^U} z_{i,k} s_i \|x_i - \mu_k\|^2 \to \min$$

Algorithm 1 Diverse mini-Batch Active Learning (DBAL)

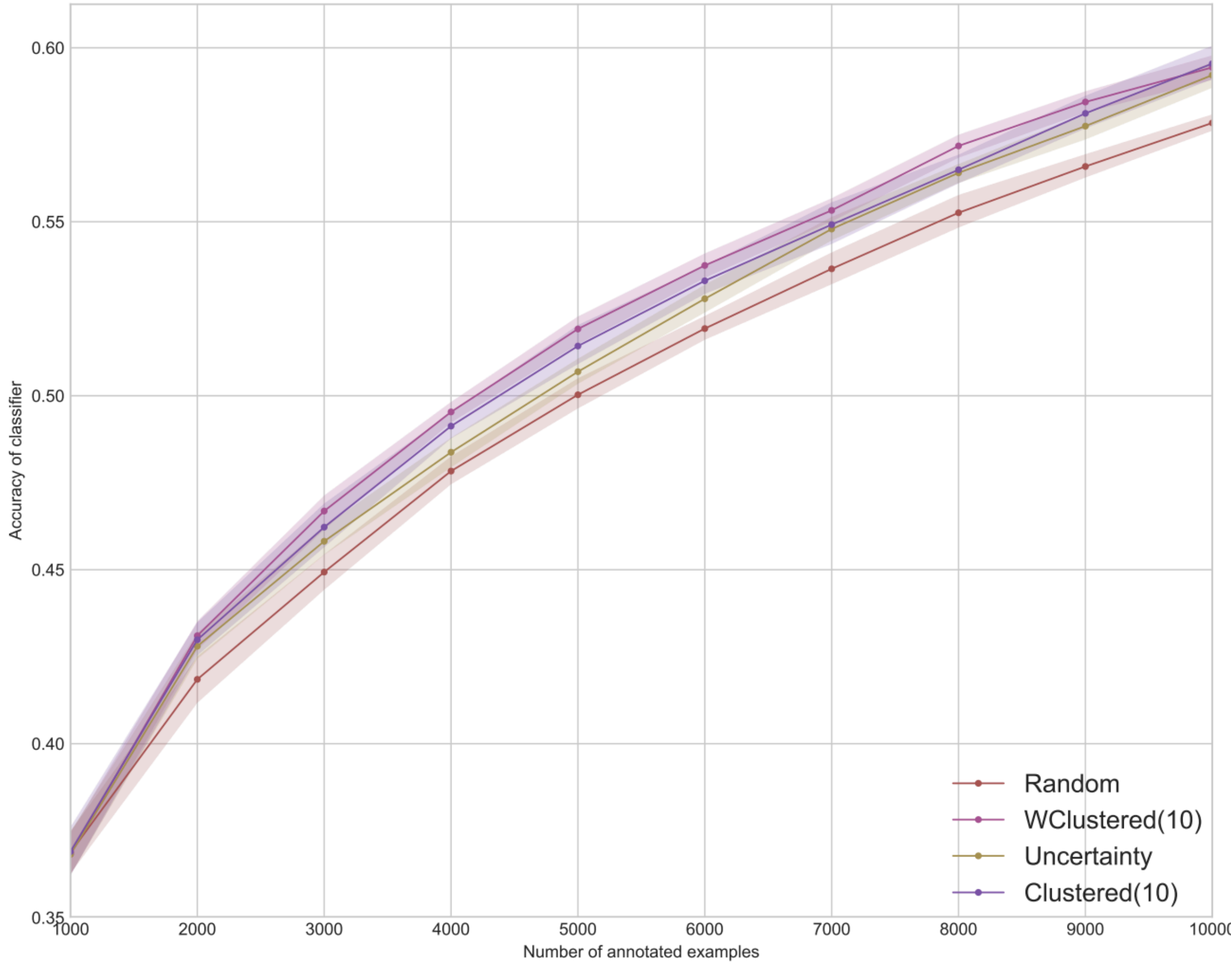
Input: dataset of examples x_i, budget B, batch-size k, pre-filter factor β
Select first k examples randomly, obtain labels for these examples
repeat
  Train classifier on all the examples selected so far
  Get informativeness for every unlabeled example
  Prefilter to top βk informative examples
  Cluster βk examples to k clusters with (weighted) K-means
  Select k different examples closest to the cluster centers, obtain labels for these examples
until Budget B is exhausted

Diverse mini-batch AL

MNIST



CIFAR-10



Observation noise

The observations (labels) are often noisy.

Can lead to information collapse. What happens if there is more noise in parts of the data space?

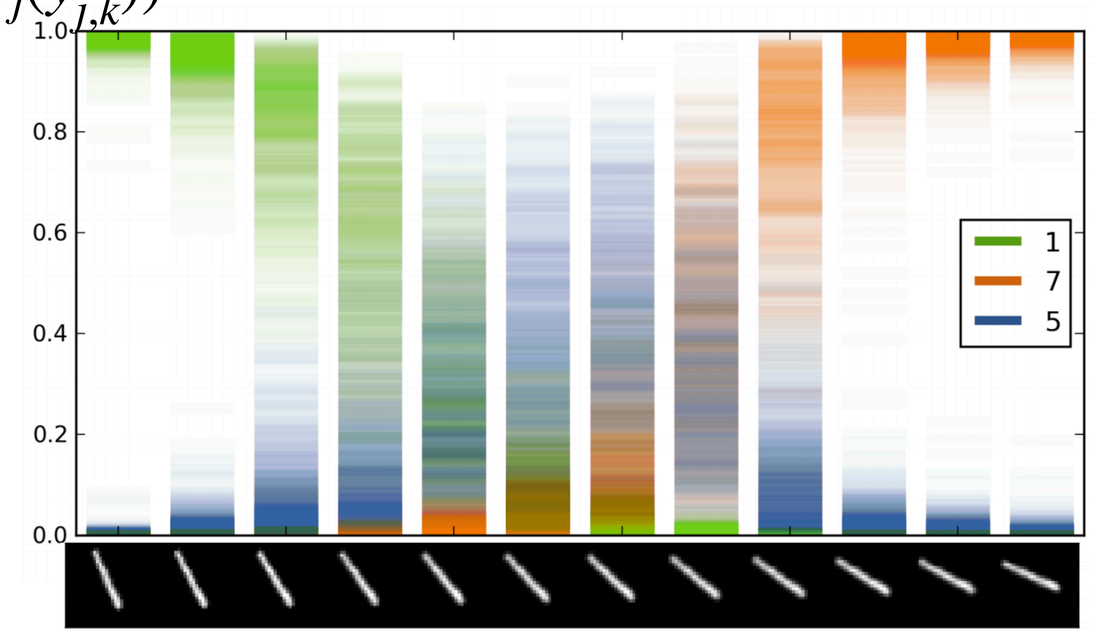
However: Easy to simulate effects! https://modal-python.readthedocs.io/en/latest/content/examples/active_regression.html

Dropout uncertainty

What if we use Dropout ***n*** times in the forward pass of a neural network? (https://arxiv.org/abs/1506.02142)

Uncertainty:

$$\text{Unc}(x_k) = \sum_{i=1}^{L} (\text{Var}_j(\hat{y}^i_{j,k}))$$



Deep Bayesian AL with Image Data

Bayesian NN: Prior on weights.

Approximate posterior using Dropout or other regularization.

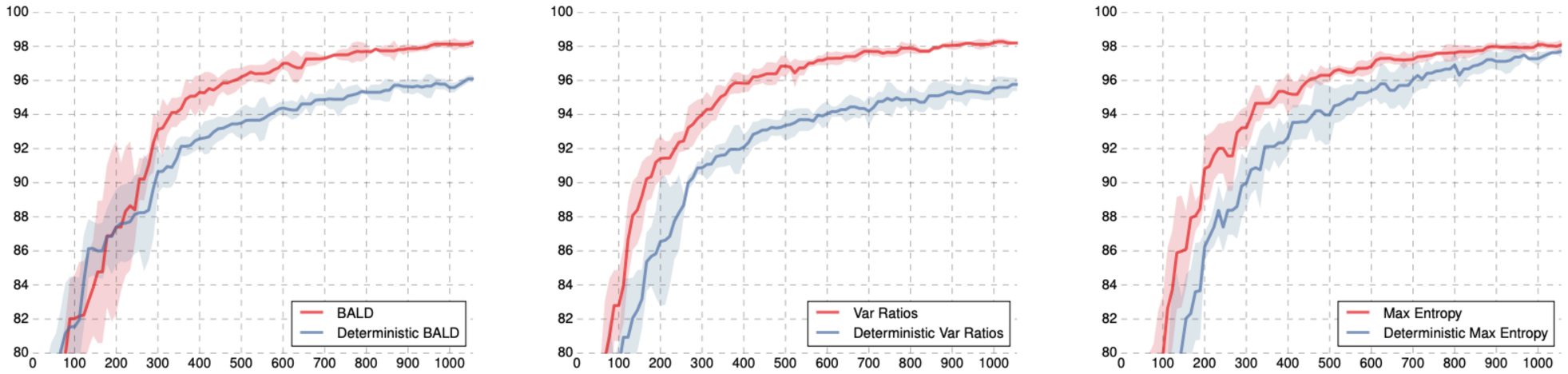
Main takeaway:

- Using Bayesian posterior approximation yields better query strategy

https://arxiv.org/abs/1703.02910

Deep Bayesian AL with Image Data

https://github.com/Riashat/Deep-Bayesian-Active-Learning



(a) BALD (b) Var Ratios (c) Max Entropy

Figure 2. Test accuracy as a function of number of acquired images for various acquisition functions, using both a **Bayesian CNN (red) and a deterministic CNN (blue)**.

Semi-supervised learning

Using unannotated data to inform learning process

Semi-supervised learning

Similar to transfer learning, but dataset (from same domain) does not have labels.

The litterature on semi-supervised learning is huge!

http://www.acad.bg/ebook/ml/MITPress-%20SemiSupervised%20Learning.pdf

A more recent study of approaches can be found here:

https://arxiv.org/abs/1804.09170

https://github.com/brain-research/realistic-ssl-evaluation

Idea 1: Using predictions as pseudo-labels

Use model to predict on ***U***.

If model is sure about predictions on $S \subset U$, add ***S*** to ***L***.

This is called *pseudo-labeling*.

Idea 2: Perturbation on inputs

(Small) Perturbation on inputs should not affect predictions to much.

Example: Obtain predictions, $\hat{y}_i = p(x_i^U) \forall x_i^U \in U$

Train model on $\hat{y}_i$ using a strongly regularized model. E.g. heavy dropout and noise.

What does this mean?

Idea 2: Perturbation on inputs

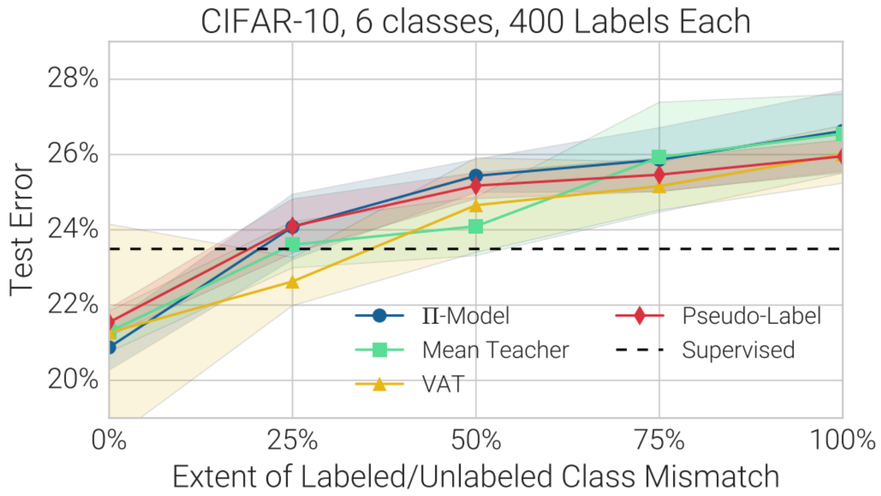
Added simple Virtual Adversarial Training example to repository:

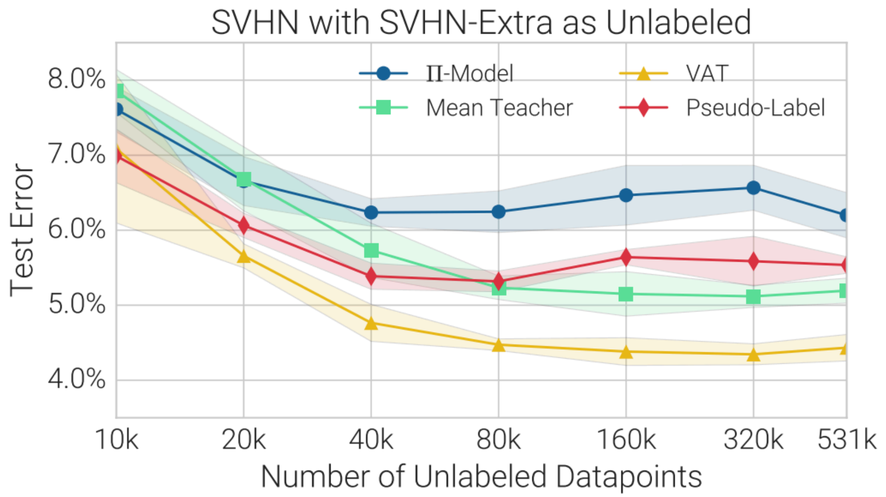
https://github.com/Froskekongen/MA8701/blob/master/semisupervised/vat_example.py

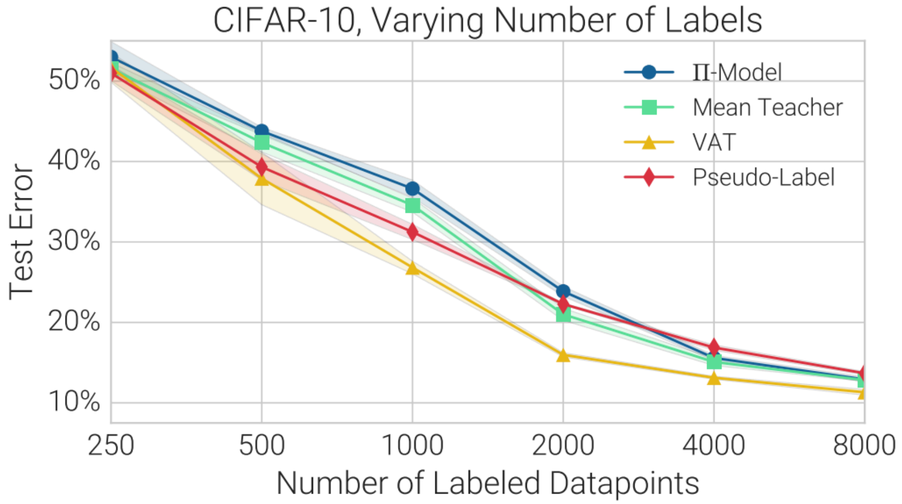
Needs modification to be used in semi-supervised setting. The paper is easy-to-read:

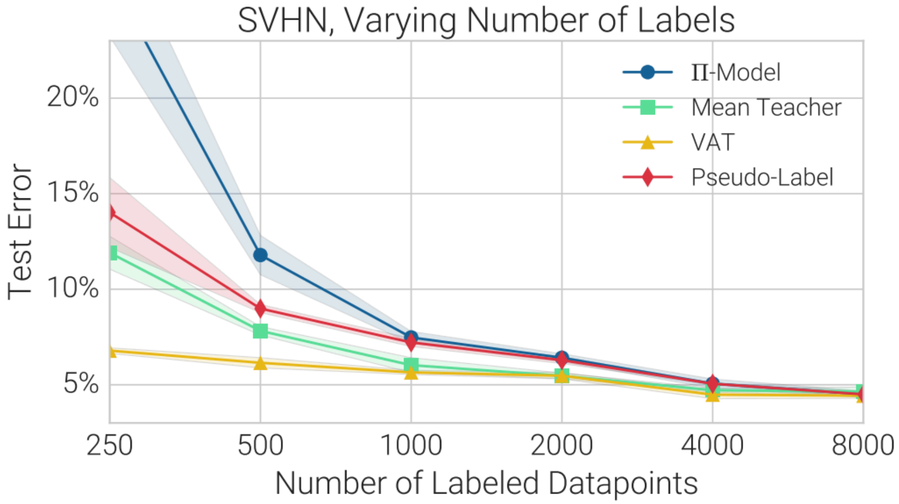
https://arxiv.org/abs/1704.03976

Idea 2: Perturbation on inputs









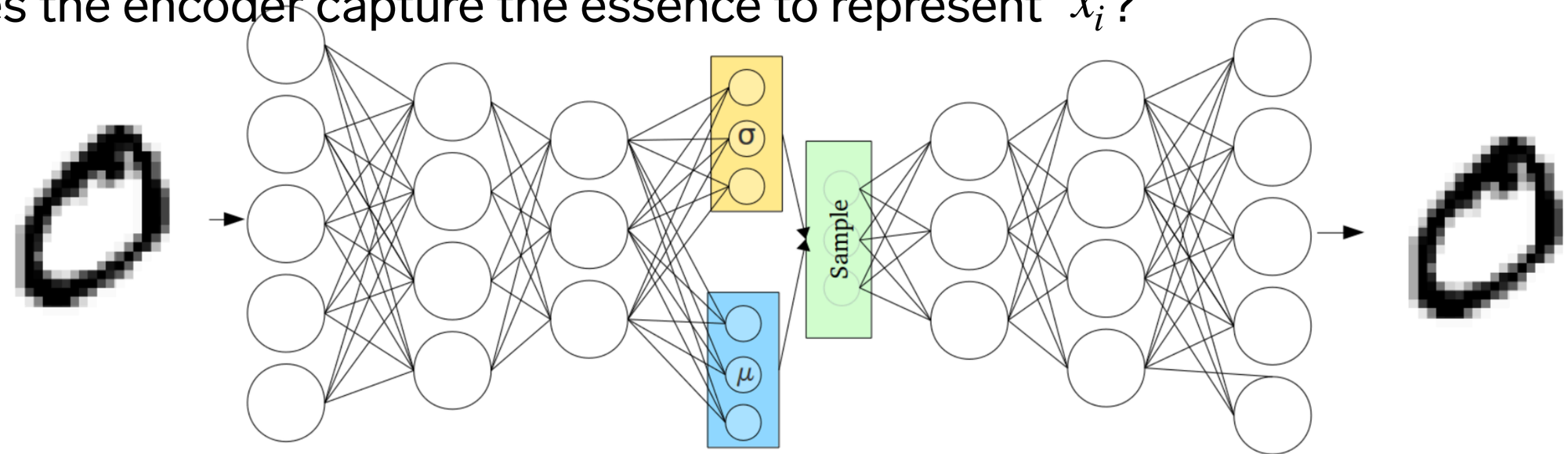
Idea 3: Use autoencoders

What is an autoencoder?

We have an encoder $z_i = f(x_i)$

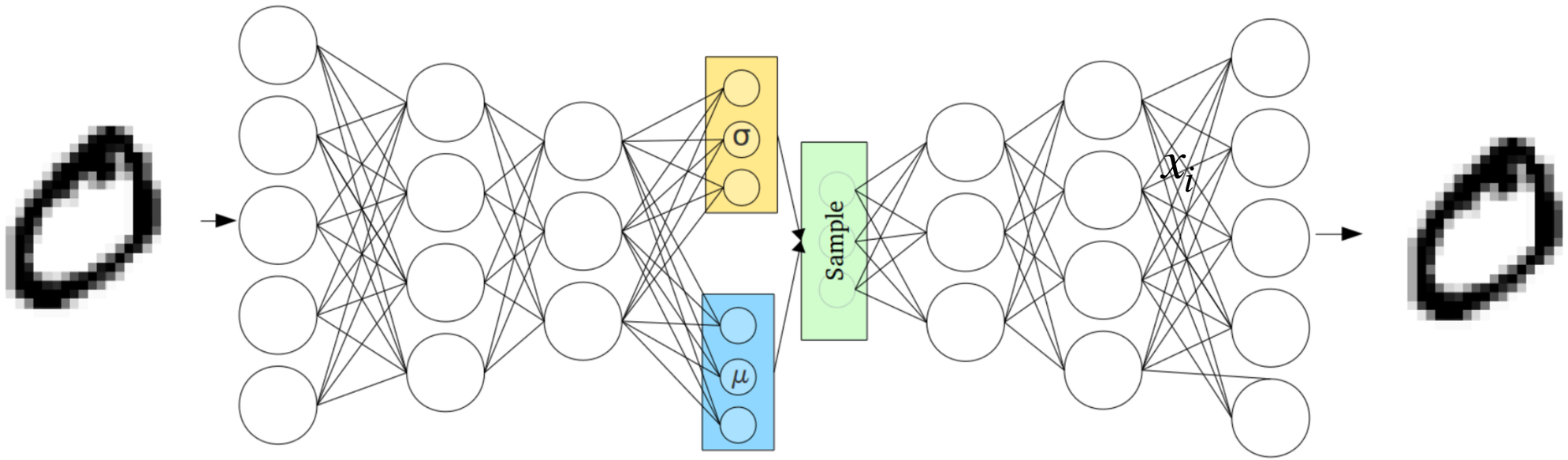
And a decoder $g(z_i) \approx x_i$

Does the encoder capture the essence to represent x_i?



Idea 3: Use autoencoders

Hard to make semi-supervised learning with autoencoders work well!

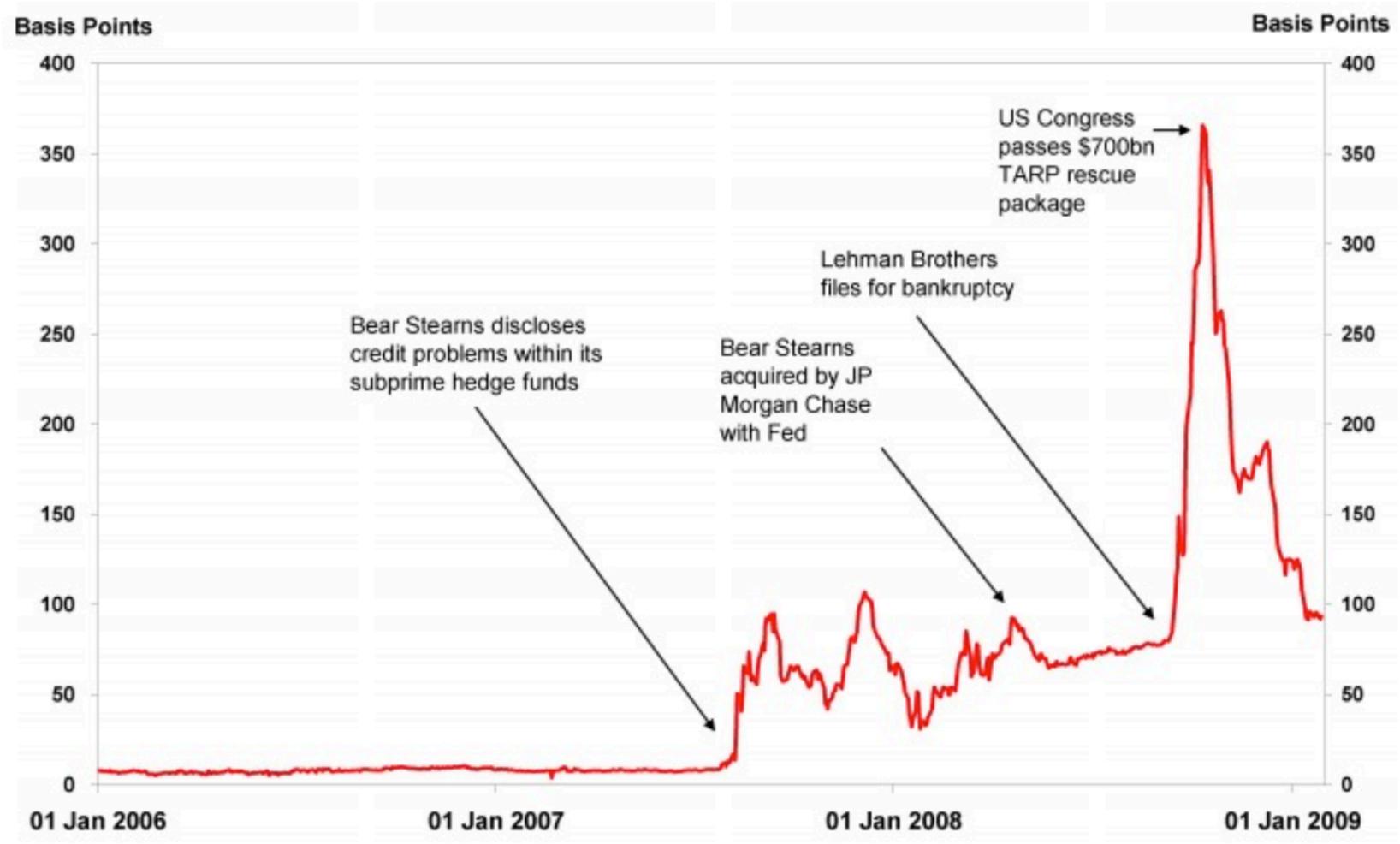


One-shot learning

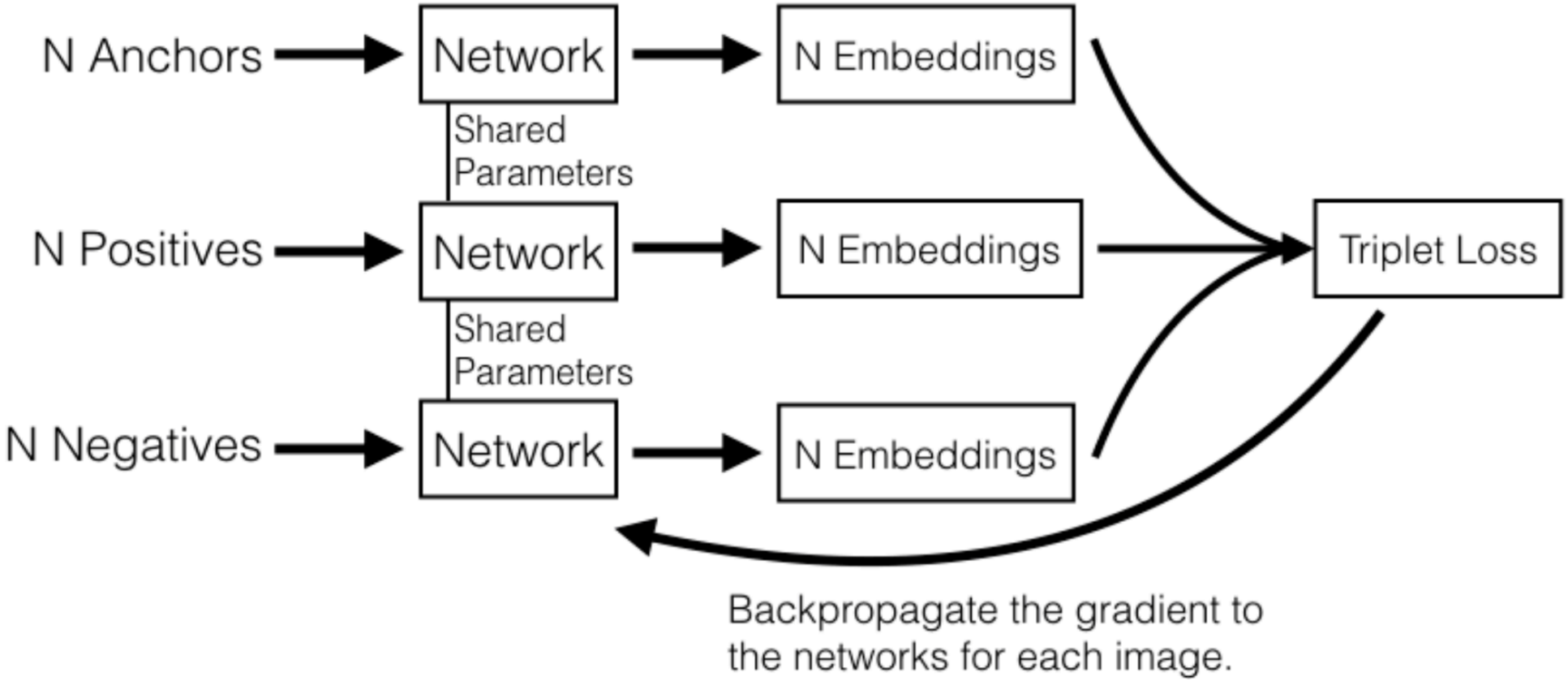
We see a class only once (or a few times) during training

One-shot learning

We want to characterise the important aspects from few observations



Siamese networks



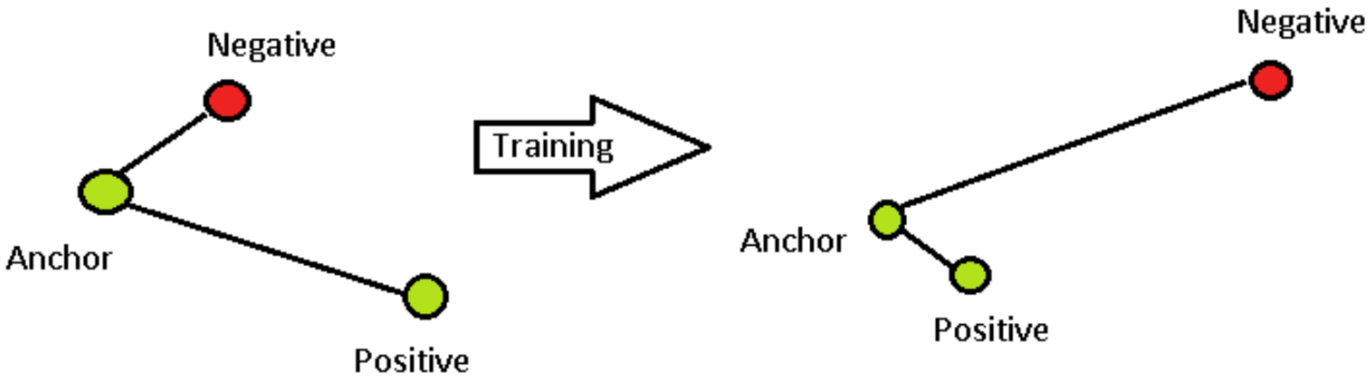
$$Loss = \sum_{i=1}^{N} \left[\| f_i^a - f_i^p \|_2^2 - \| f_i^a - f_i^n \|_2^2 + \alpha \right]_+$$

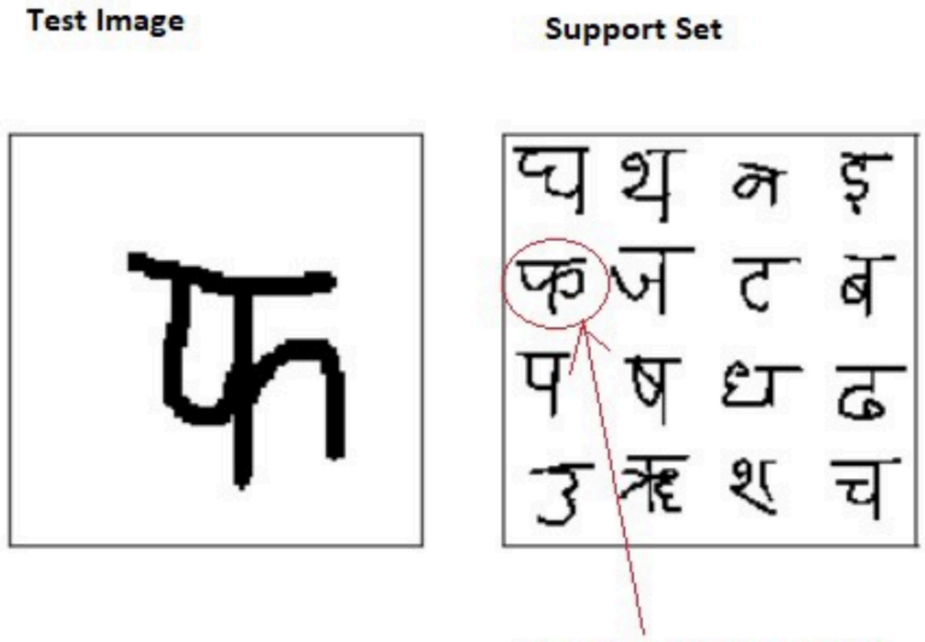
Siamese networks

Challenge

Choose three good data points: Anchor, positive and negative examples.

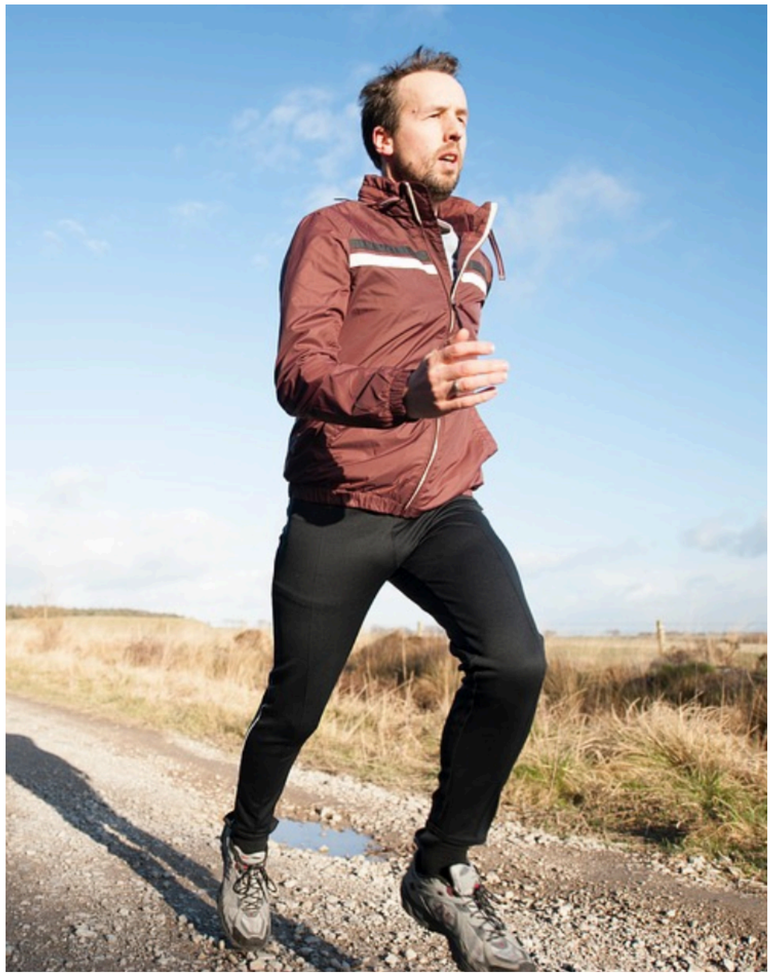
https://arxiv.org/abs/1503.03832





Siamese networks

Is there an easier version of the problem?



vs.

Combining small-data techniques

AL and semi-supervised learning

- Use both information from ***U*** for strengthening the model AND choosing samples.

Exercise: Combine pseudo-labeling with entropy strategy: Pseudo label the least-entropic labels, and query the most entropic. (Subsample 200 imgs from CIFAR-10 and MNIST as a starting point for ***L***)

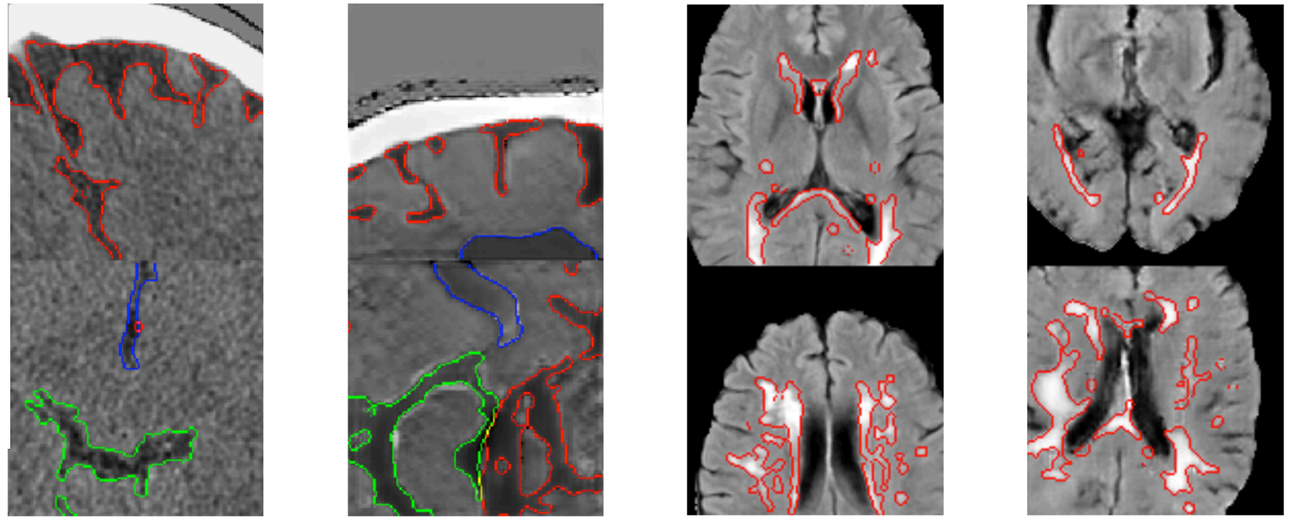
Does this strategy work?

AL data augmentation

- Recent research tries using GANs for data augmentation (https://arxiv.org/abs/1810.10863)

- Can we combine AL with such data augmentation to increase robustness? What synthetic data should be labeled? What if we also have a pool of unlabeled data?

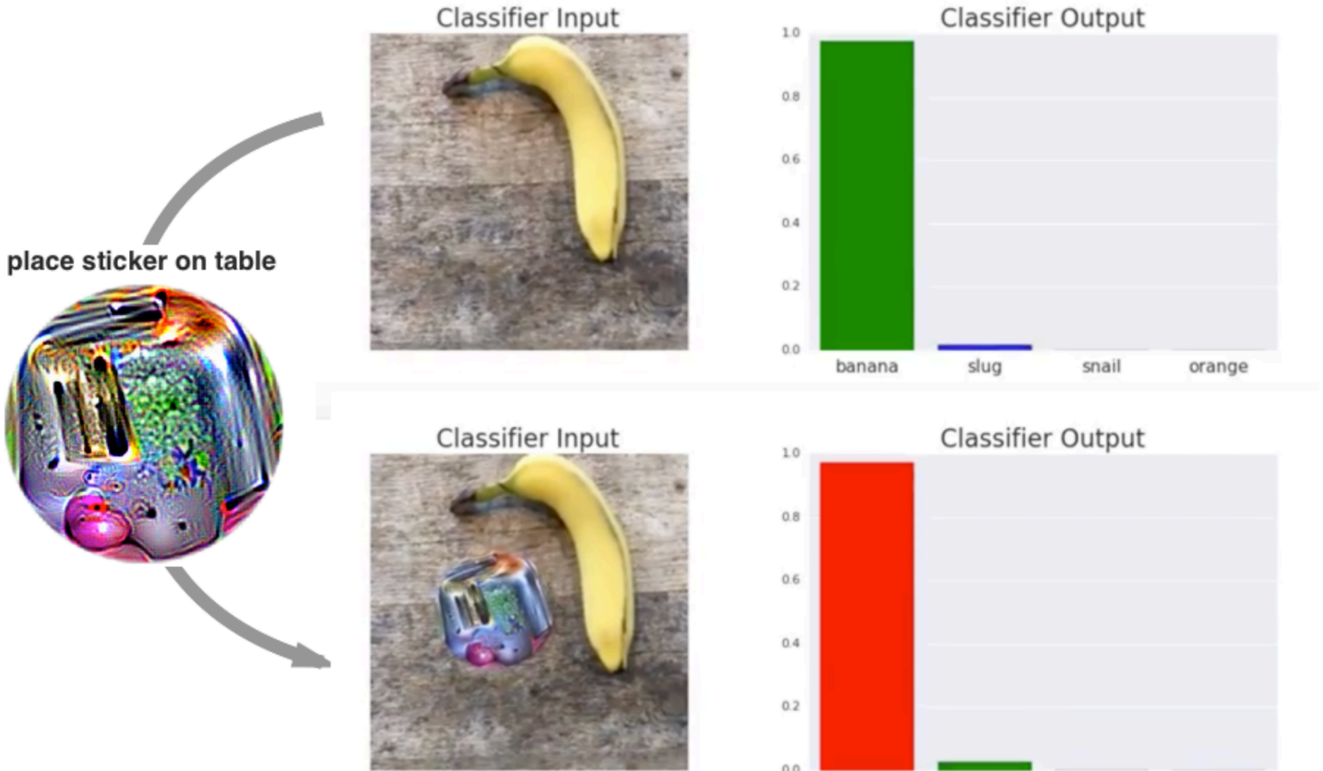
	Available data		
	100%	50%	10%
No augmentation	88.1 (0.32)	85.0 (0.58)	75.1 (0.60)
GAN augmentation	88.4 (0.41)	85.6 (1.33)	76.3 (1.77)
Rotation augmentation	**88.9** (0.51)	**86.0** (0.50)	**76.9** (0.58)
GAN + Rotation augmentation	**89.3** (0.39)	**86.9** (0.36)	**78.4** (0.99)

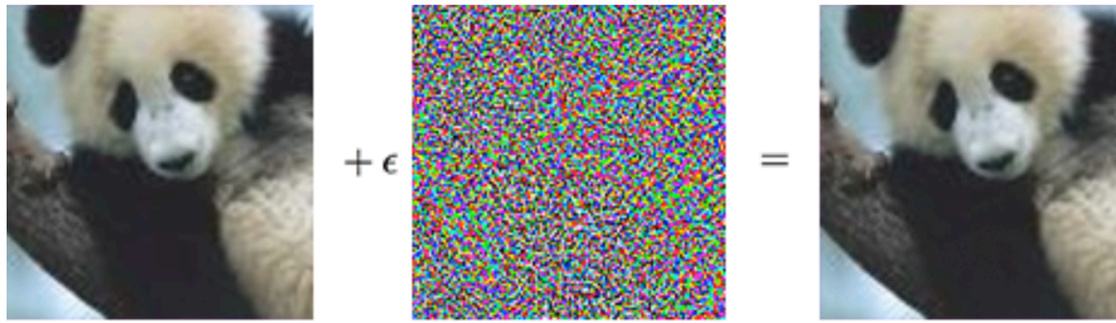


(a) Real CT (b) Synthetic CT (c) Real MRI (d) Synthetic MRI

AL with adversarial examples

- Adversarial examples can “attack” models (https://openai.com/blog/adversarial-example-research/, https://arxiv.org/abs/1712.09665)





“panda”

57.7% confidence

“gibbon”

99.3% confidence

AL with adversarial examples

Same challenge applies for text. E.g. spam-filter fooling, ads, etc...

Question: Can an annotator/process help making a model robust against adversarial attacks? What would a natural query strategy be?

AL with transfer learning

Use a pre-trained model as baseline.

Exercise: Use a model pretrained on ImageNet, and take a seed set of 100 images from CIFAR-10. Use entropy-sampling. How does this compare with no pretraining?

The major question

What would you do ***in practice*** if you were given an large unlabeled dataset?

Example: Classifying the what cuisine a dish is from based on an image.

Exercises

The only way to learn this stuff is trying it out yourself!

Suggestion 1: Active Learning

Take the bag-of-words example from Ben.

Extract a subset of size N (for example 1000), and leave a large “unlabeled” dataset

Use https://github.com/modAL-python/modAL to add samples from the unlabeled dataset.

Does it work better than sampling at random?

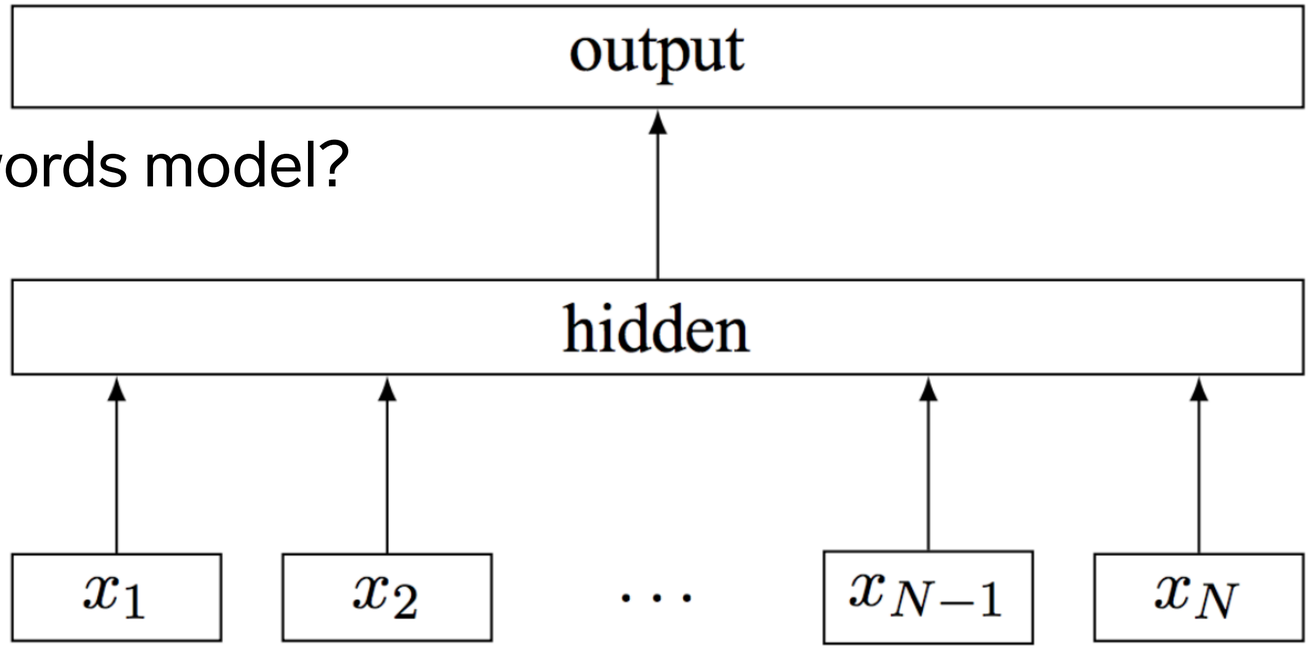
$$u_i = \sum_k P(y_i = k \,|\, x_i) \log P(y_i = k \,|\, x_i)$$

Suggestion 2: Transfer learning

Take russian word embeddings: https://rusvectores.org/en/models/.

Train a continuous bag of words model on this: That is use average of all word embeddings in a text, rather than one-hot encodings.

Does it work better than bag-of-words model?



Suggestion 3: Semi-supervised learning

Take a subset of 1500 images from MNIST (or CIFAR-10) - use 500 images as validation set

Use pseudo-labeling on full dataset to see how much you can improve validation error on the validation set.

Alternatively, modify code on https://github.com/Froskekongen/MA8701/blob/master/semisupervised/virtual_adv_training_baseline.py for semi-supervised learning

Suggestion 4: Siamese networks

Take a subset of 1500 images from MNIST (or CIFAR-10) - use 1000 images as validation set

Use siamese network to classify images into the different categories

Starting point: https://github.com/Goldesel23/Siamese-Networks-for-One-Shot-Learning

Note: This is a more challenging task than the others.