
Learning with Small Samples

Including zero-shot learning

Nour Karessli
DSR 2018



Nour Karessli

Computer vision engineer, EyeEm
Master in computer science, Saarland University
Bachelor in software engineering, Damascus University

nour.karessli@gmail.com
[LinkedIn](#)

About this tutorial

- A basic understanding of zero-shot and low-shot learning
- Get to know state of the art approaches
- Hands-on experience with zero-shot image classification
- Hands-on experience with training image classifier with small set



Prerequisites

- Basic math e.g. matrix operations, derivatives,.., etc
- Basic understanding of ML concepts e.g. classifier, loss function,.., etc.
- Basic DL concepts e.g. MLP, CNN, LSTM, ..etc
- Python, Keras

Structure

- Introduction & motivation
- Zero-shot learning
 - Definition
 - Side information
 - Zero-shot learning models
 - Exercise
- Low-shot learning
 - Definition
 - Low-shot learning models
- Tips & tricks
- Exercises

Structure

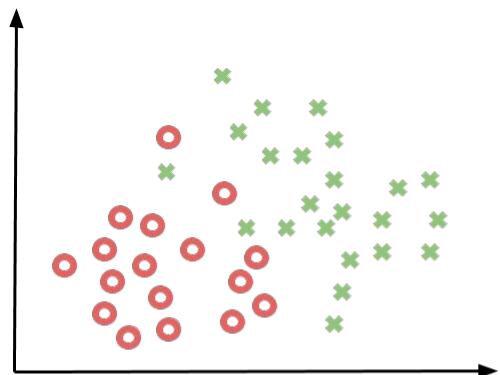
- Introduction & motivation
- Zero-shot learning
 - Definition
 - Side information
 - Zero-shot learning models
 - Exercise
- Low-shot learning
 - Definition
 - Low-shot learning models
- Tips & tricks
- Exercises



Introduction

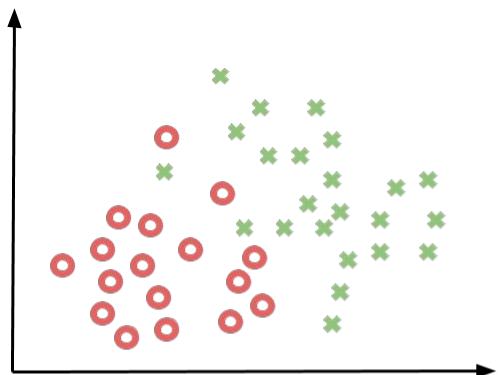
Learning

Supervised

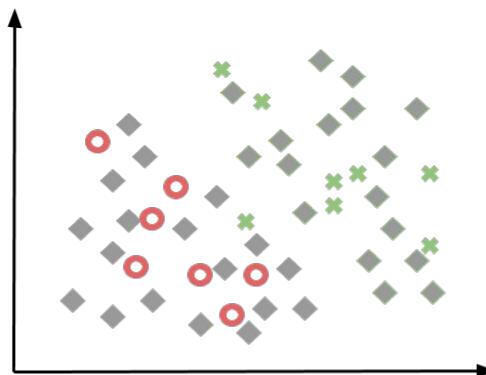


Learning

Supervised

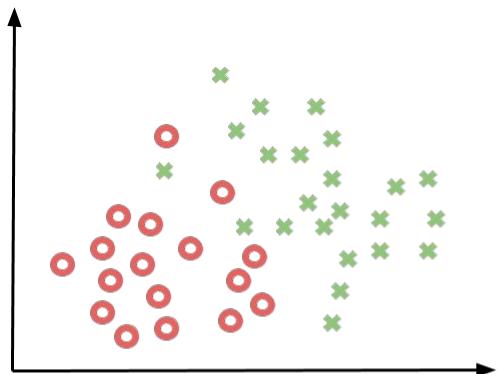


Semi-supervised

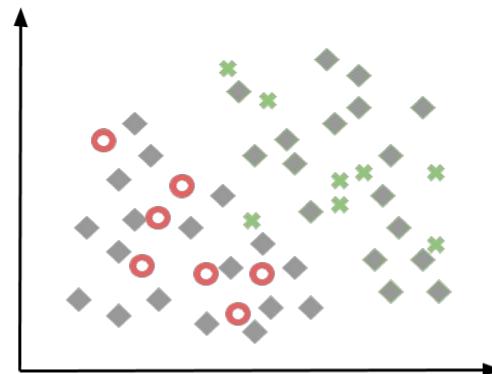


Learning

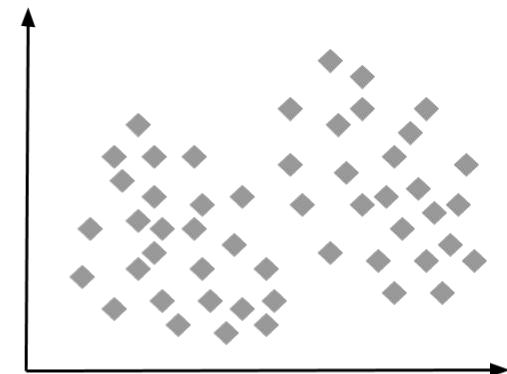
Supervised



Semi-supervised

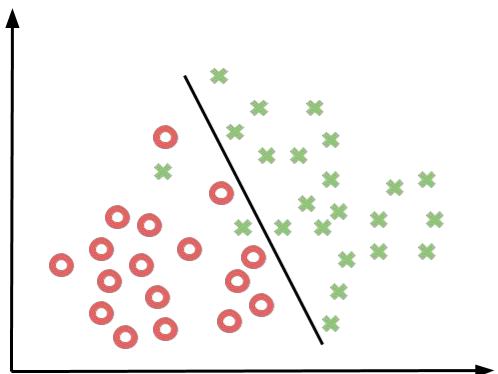


Unsupervised



Learning

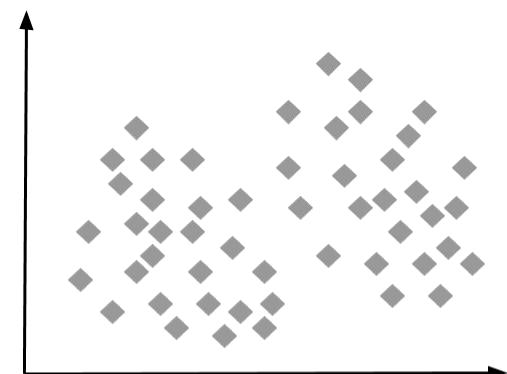
Supervised



Semi-supervised

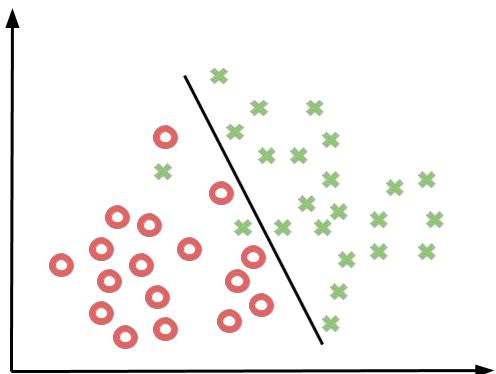


Unsupervised

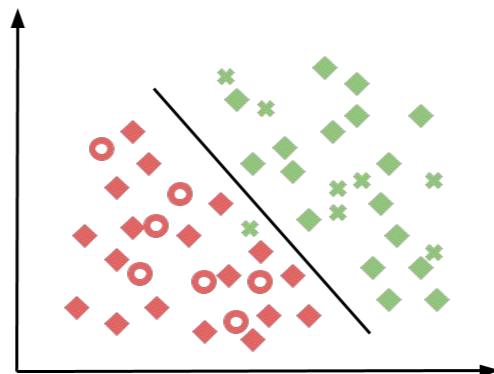


Learning

Supervised



Semi-supervised

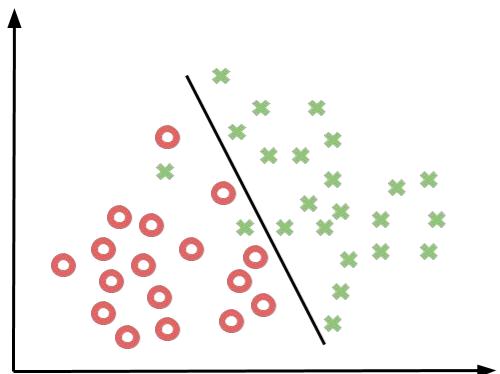


Unsupervised

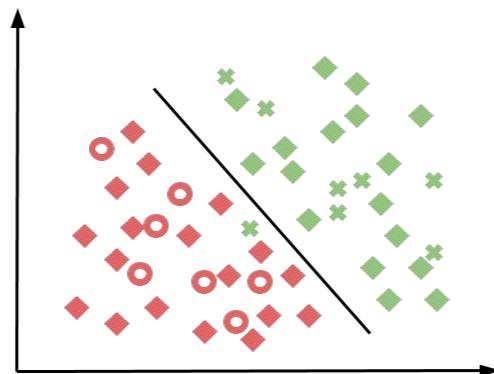


Learning

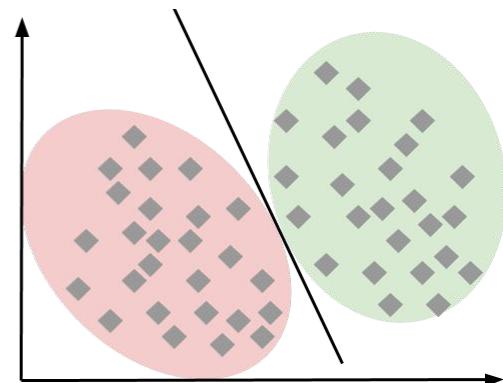
Supervised



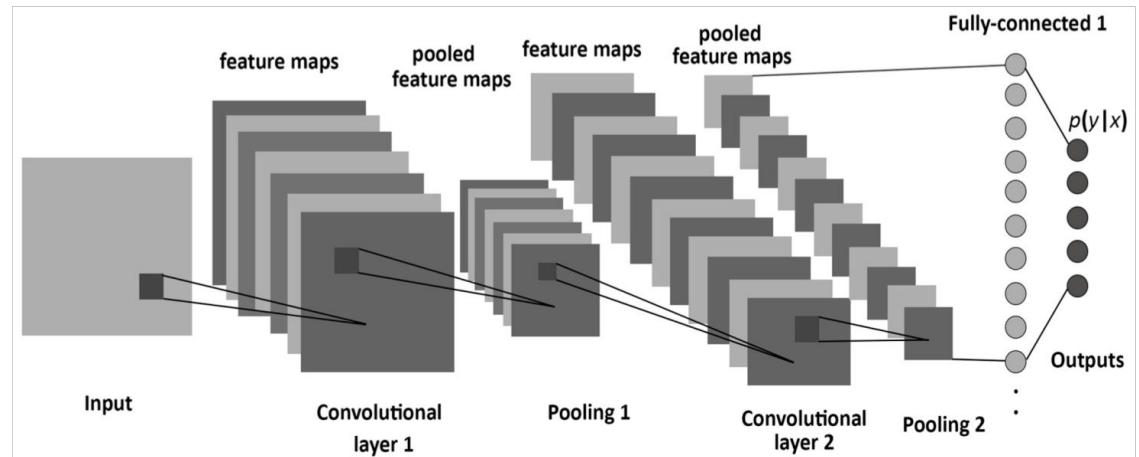
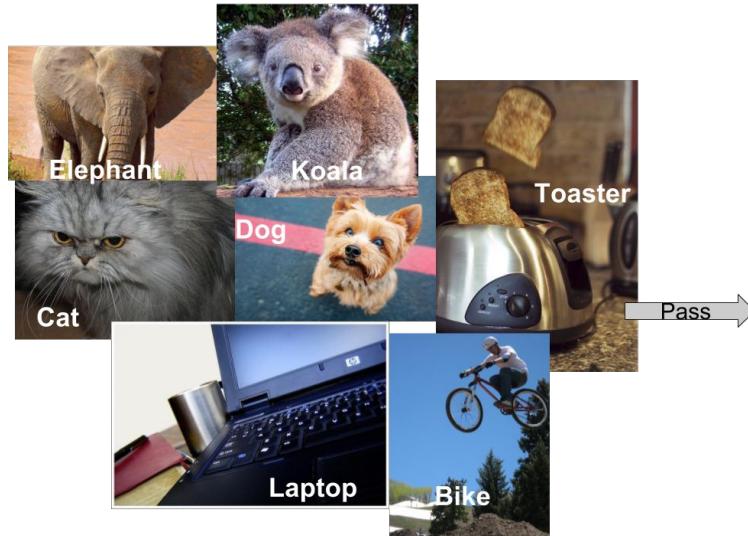
Semi-supervised



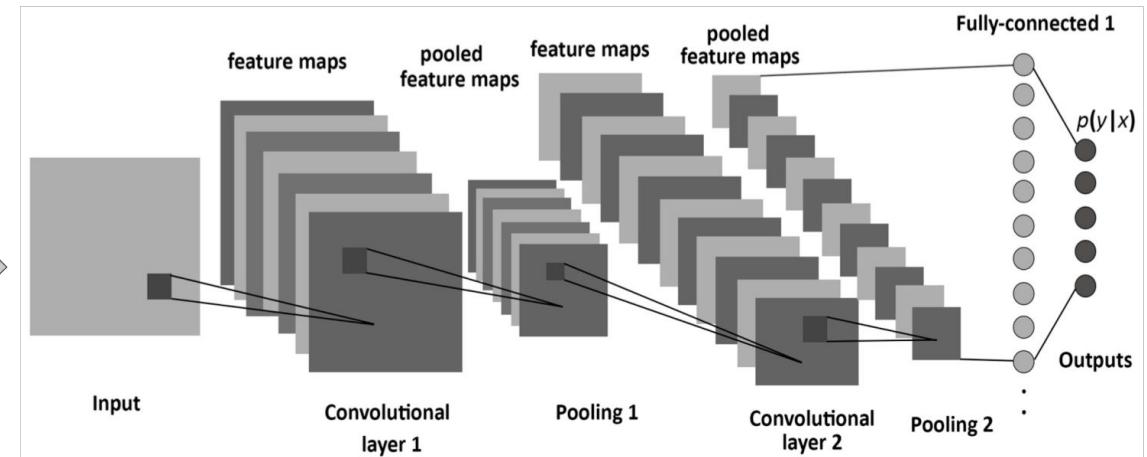
Unsupervised



Many-shots supervised learning



Many-shots supervised learning



Annotation Effort

IMAGENET



14M images, 21K categories

[Deng, et al CVPR2009]

Annotation Effort

- OpenImages
9M images, 6K categories
- COCO
330K images
- MIT Places
2.5M images



Task complexity

- Classification
Single labeling is relatively easy



Capybara

Task complexity

- Classification
Single labeling is relatively easy
- Localization
Requires precise bounding boxes



Task complexity

- **Classification**

Single labeling is relatively easy

- **Localization**

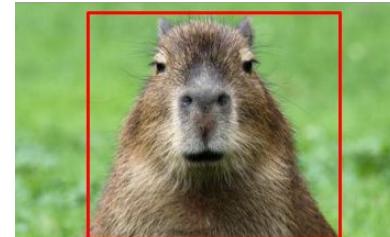
Requires precise bounding boxes

- **Captioning**

Many other possibilities!

“Surprised capybara looking at the camera”

“Portrait photo of a capybara”



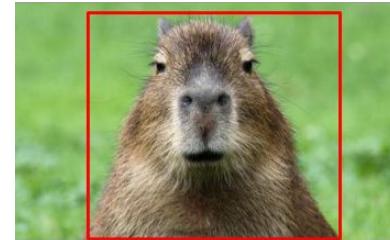
Task complexity vs. annotation

- Classification
ImageNet 14M labeled images



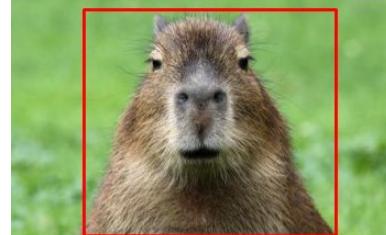
Task complexity vs. annotation

- Classification
ImageNet 14M labeled images
- Localization
openImages 3.6M bounding boxes



Task complexity vs. annotation

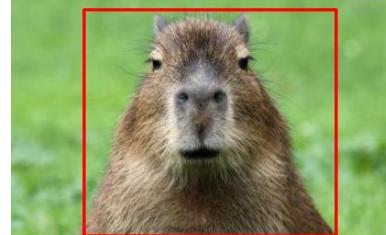
- Classification
ImageNet 14M labeled images
- Localization
openImages 3.6M bounding boxes
- Captioning
COCO 300K captioned images



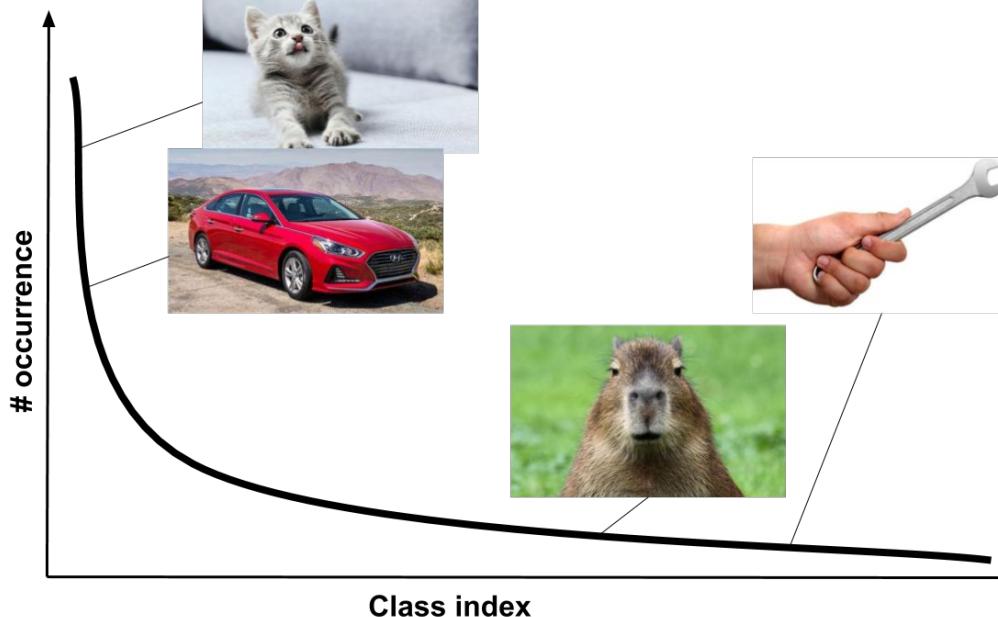
Task complexity vs. annotation

- Classification
ImageNet 14M labeled images
- Localization
openImages 3.6M bounding boxes
- Captioning
COCO 300K captioned images

Less annotations for more complex tasks!



Tail distribution



<https://www.cars.com/>

<http://www.foxnews.com/lifestyle/2017/11/09/how-to-keep-cat-from-scratching-your-sofa-to-shreds.html>

<https://www.livescience.com/55223-capybara-facts.html>

<https://www.indiamart.com/proddetail/hand-wrench-13045857897.html>

Fine-grained categories

Hard: subtle differences



Oxford Pet dataset

Fine-grained categories

Hard: subtle differences



Cars dataset

Fine-grained categories

Hard: subtle differences



FGVC-Aircraft dataset

Fine-grained categories

Hard: subtle differences



60 classes of Caltech Birds dataset

Fine-grained categories

Novice annotator



Fine-grained categories

Bird expert → expensive





Zero-shot Learning

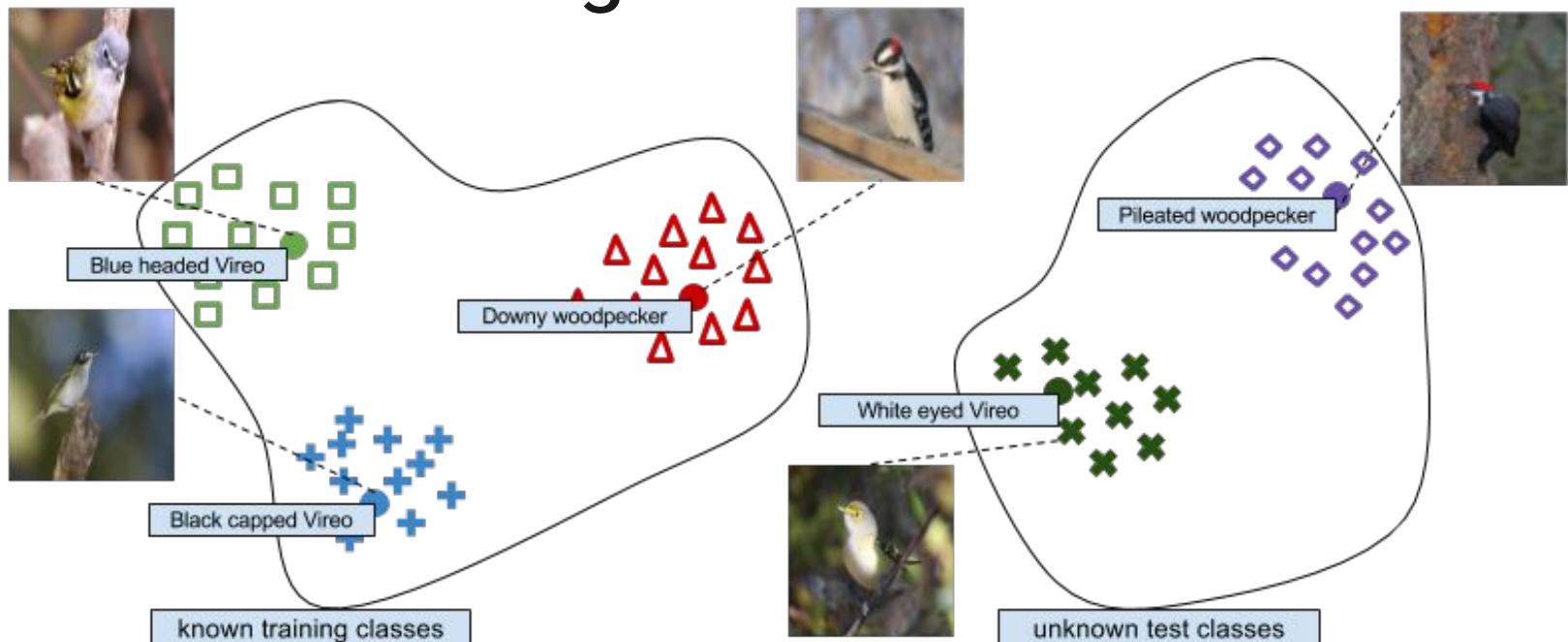
Structure

- Introduction & motivation
- Zero-shot learning
 - Definition
 - Side information
 - Zero-shot learning models
 - Exercise
- Low-shot learning
 - Definition
 - Low-shot learning models
- Tips & tricks
- Exercises

Zero-shot learning

- Extreme case of scarce training data
- Disjoint sets of training and test classes
- New class examples appear after training stage

Zero-shot learning





Zero-shot learning

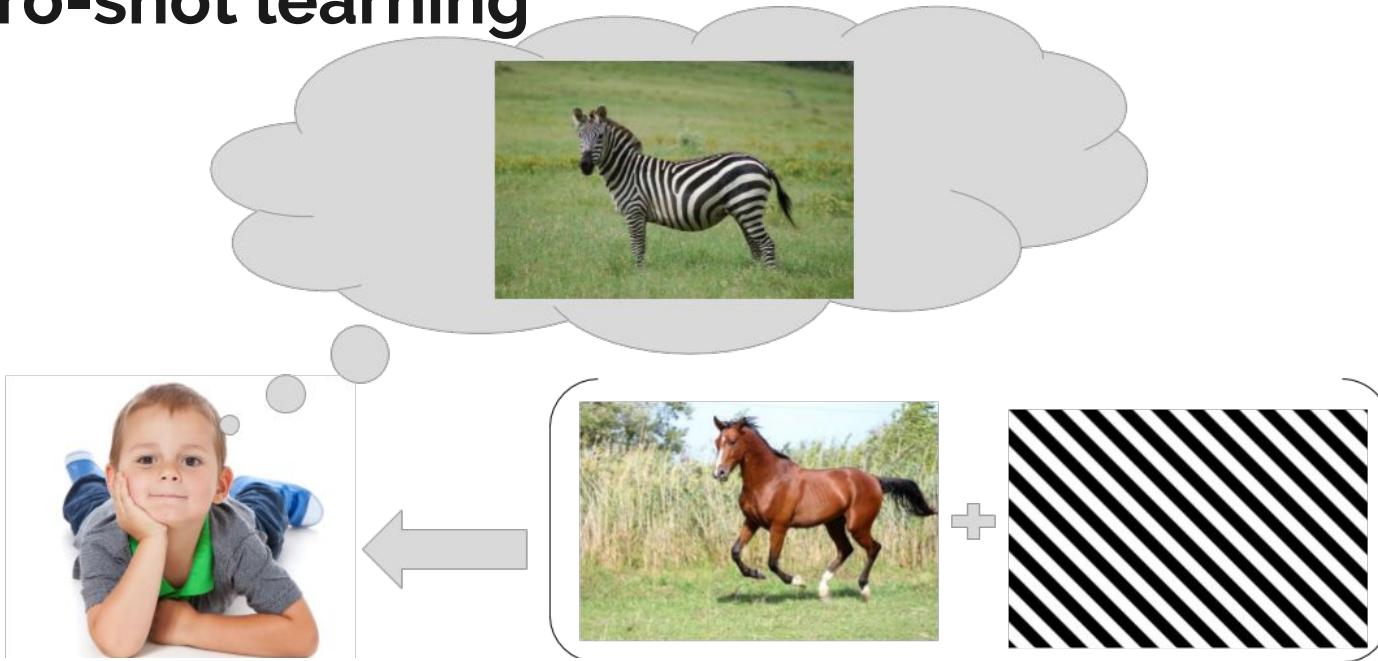
How?

Zero-shot learning



<http://www.youngparents.com.sg/development/10-steps-raising-self-confident-child/>
<http://www.thehorse.com/articles/33568/5-tips-for-packing-the-pounds-on-performance-horses>
<http://earlylearningtoys.org/stripes/>

Zero-shot learning



<http://www.youngparents.com.sg/development/10-steps-raising-self-confident-child/>

<http://www.thehorse.com/articles/33568/5-tips-for-packing-the-pounds-on-performance-horses>

<http://earlylearningtoys.org/stripes/> <https://pt.wikipedia.org/wiki/Zebra-de-grant>

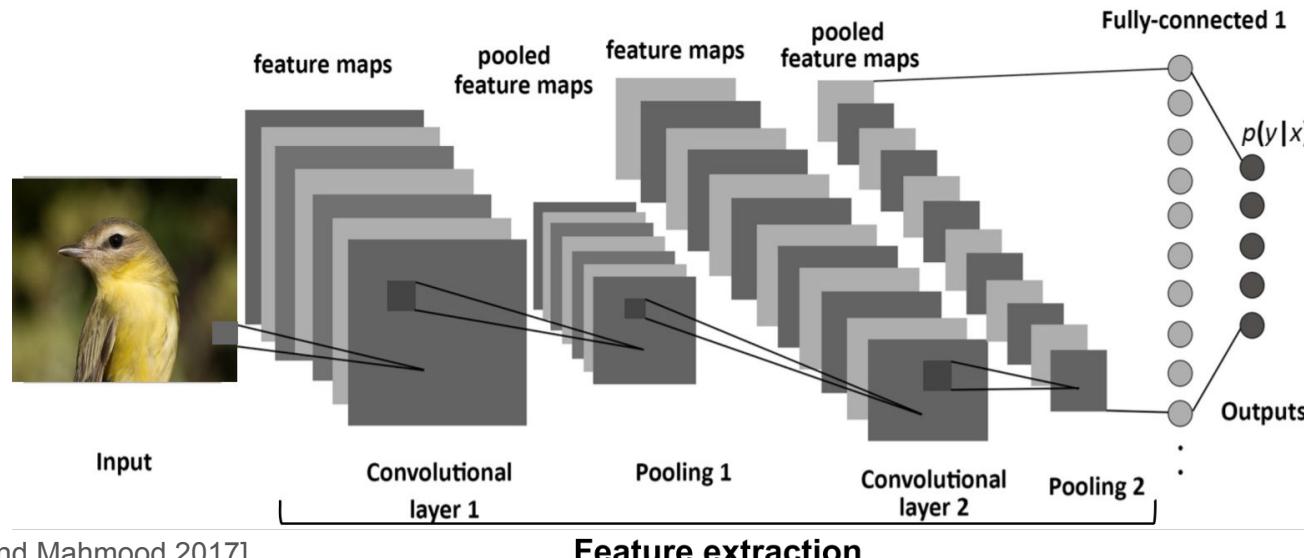


Zero-shot learning

Knowledge transfer & Side information

Knowledge transfer

Use bottleneck image features of pre-trained model

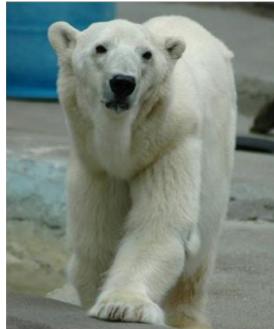




Structure

- Introduction & motivation
- Zero-shot learning
 - Definition
 - Side information
 - Zero-shot learning models
 - Exercise
- Low-shot learning
 - Definition
 - Low-shot learning models
- Tips & tricks
- Exercises

Side information - Attributes embedding



Polar bear

black: no
white: yes
brown: no
stripes: no
water: yes
eats fish: yes

[0 1 0 0 1 1]



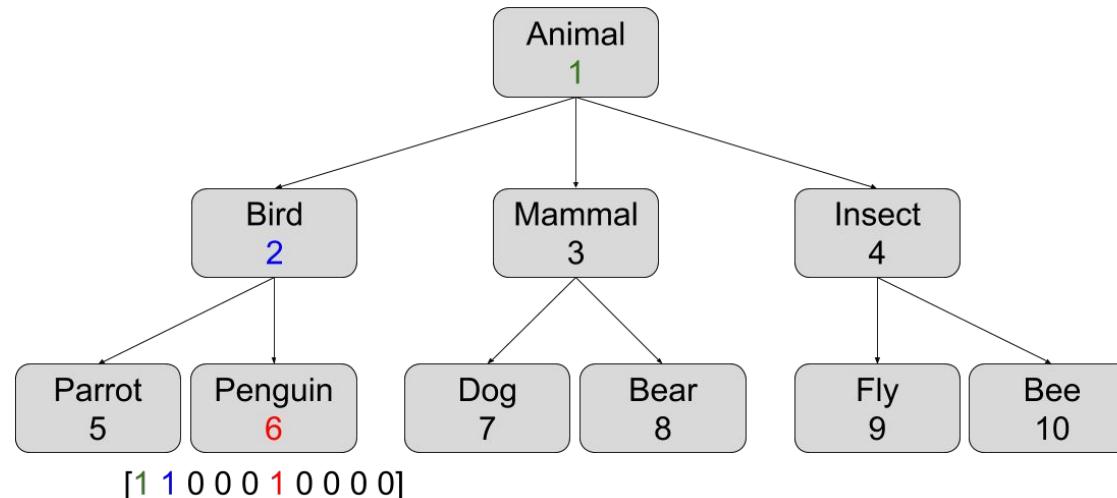
Otter

black: yes
white: no
brown: yes
stripes: no
water: yes
eats fish: yes

[1 0 1 0 1 1]

Side information - Hierarchical embedding

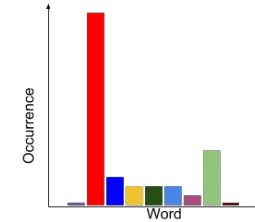
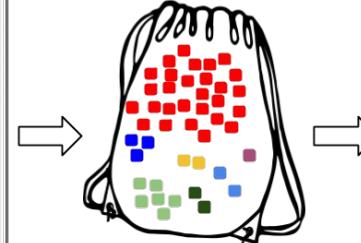
Hierarchical Label Embedding (HLE) extracted from Wordnet



Side information - Text embedding

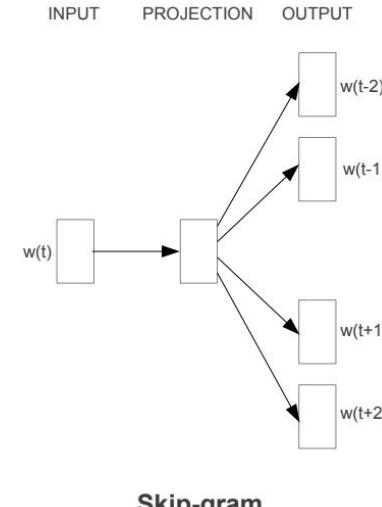
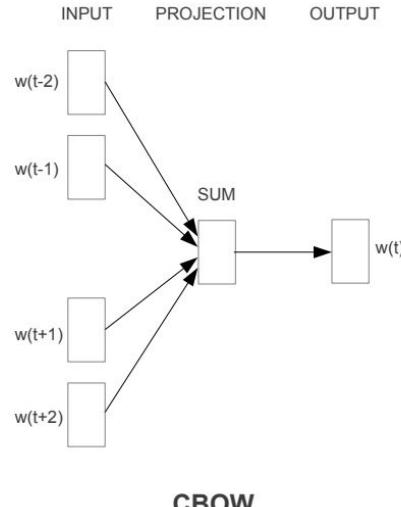
Sparse representations: Bag of Words from Wikipedia articles (BoW)

The screenshot shows the Wikipedia page for the Red-headed woodpecker (*Melanerpes erythrocephalus*). It includes sections such as Contents, Taxonomy, Description, Behavior, and Conservation, each containing text and images related to the bird's biology and habitat.



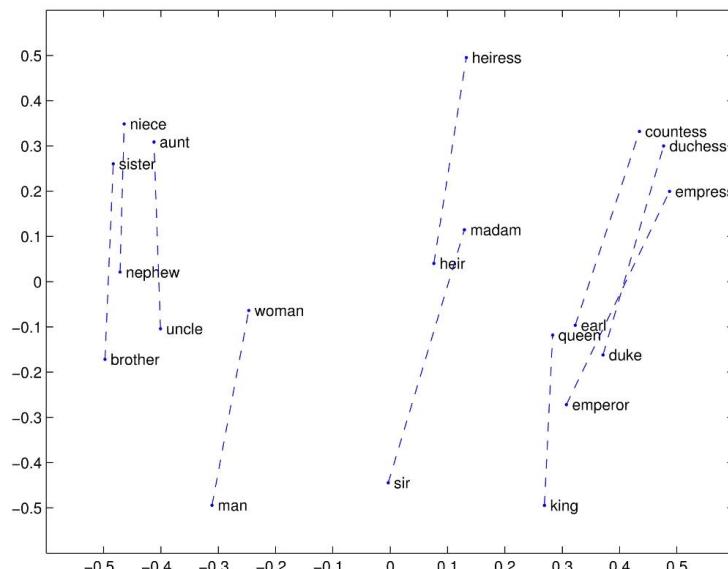
Side information - Text embedding

Dense representations: Word2vec



Side information - Text embedding

Dense representations: Global vectors for word representation (Glove)



[Pennington, et al. EMNLP2014]

Side information - Visual descriptions



The bird has a white underbelly, black feathers in the wings, a large wingspan, and a white beak.



This flower has a central white blossom surrounded by large pointed red petals which are veined and leaflike.



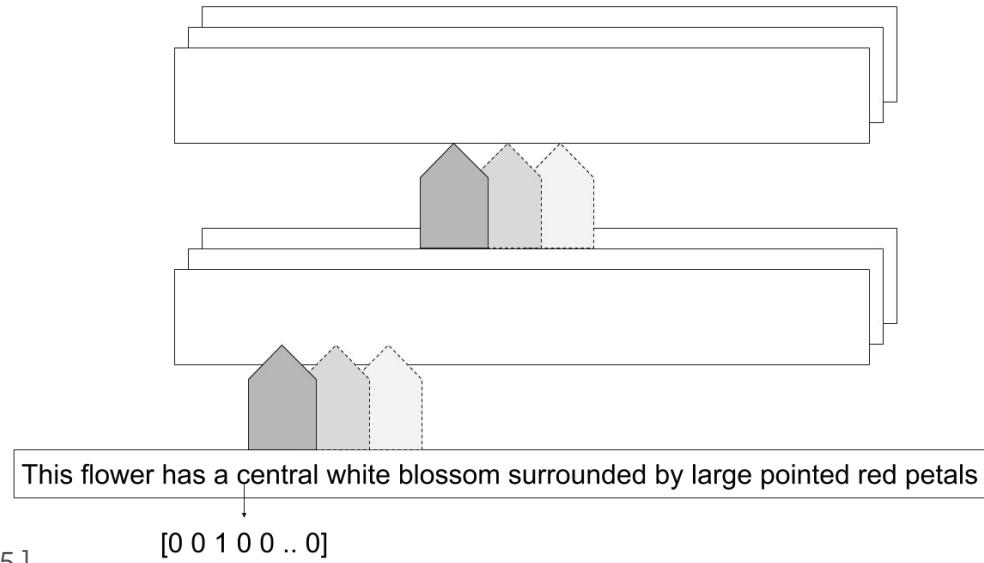
This bird has distinctive-looking brown and white stripes all over its body, and its brown tail sticks up.



Light purple petals with orange and black middle green leaves

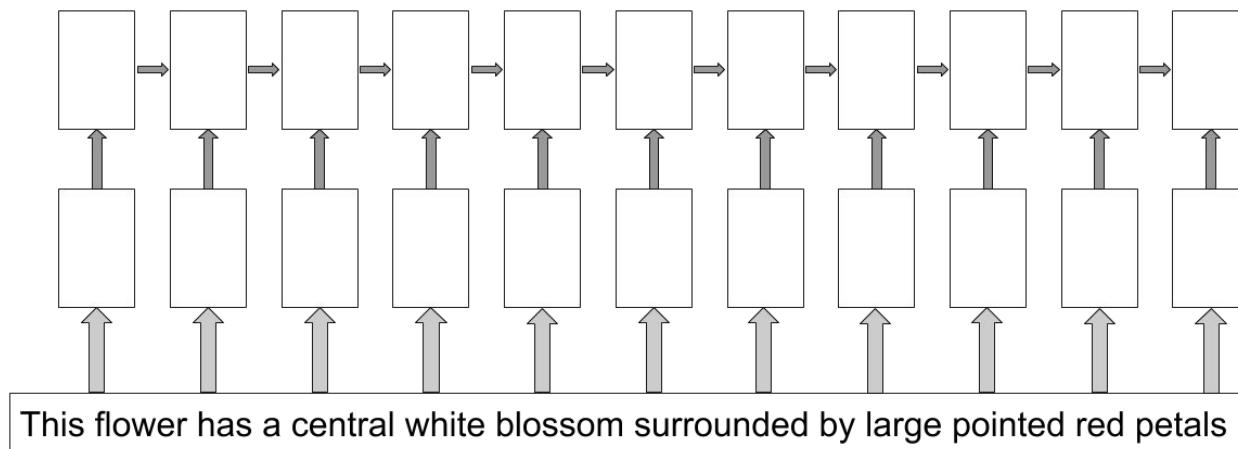
Side information - Visual descriptions

- Character level CNN



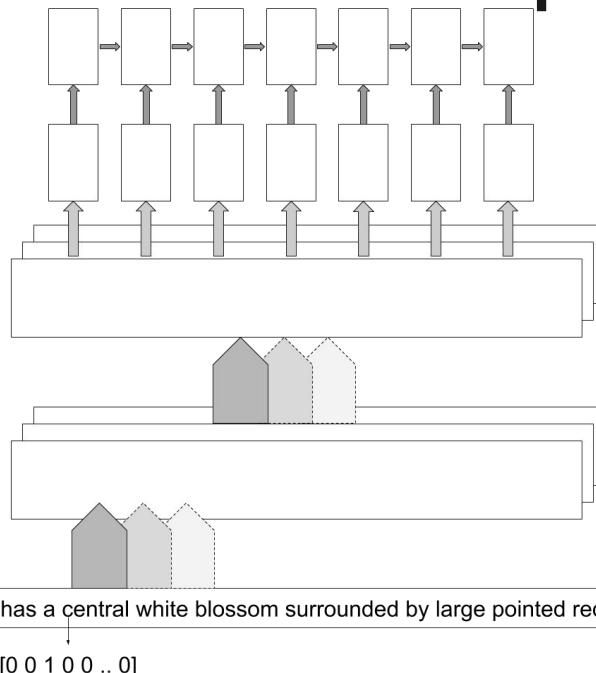
Side information - Visual descriptions

- LSTM



Side information - Visual descriptions

- CNN + RNN



Side information - Gaze embedding

- Discrimination of objects by novice
- Data collection is fast
- Implicit annotation,
you don't need to name the object



Side information - Gaze embedding

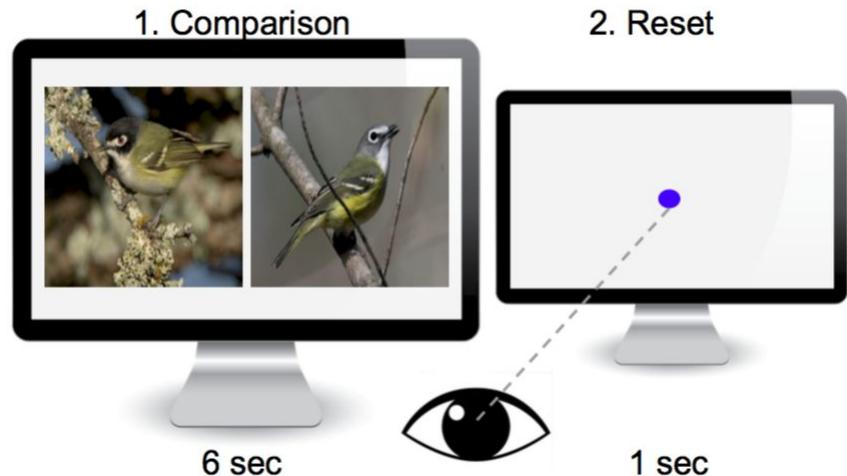
Experiment

1. Comparison



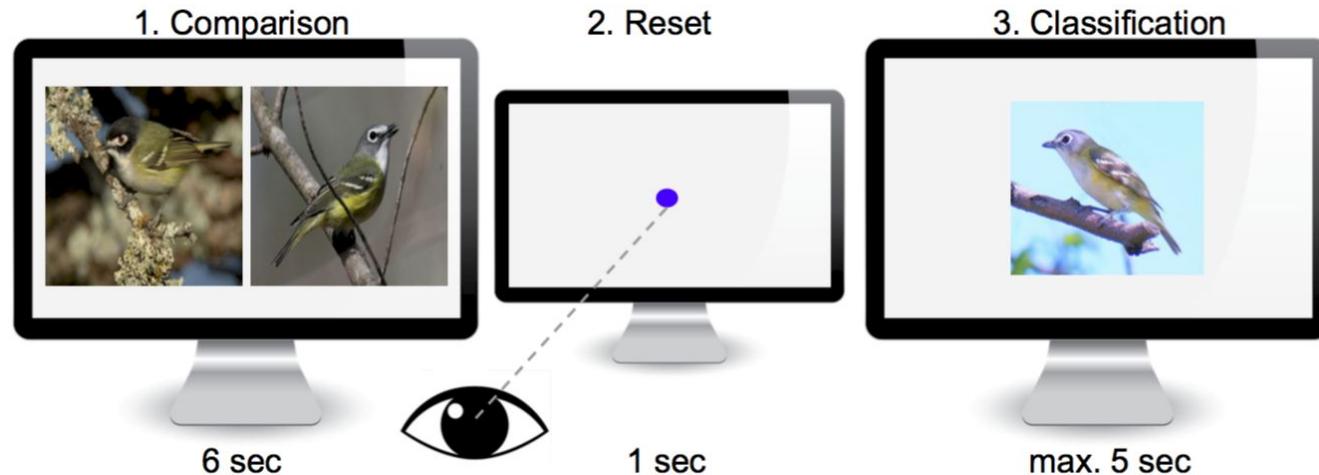
Side information - Gaze embedding

Experiment



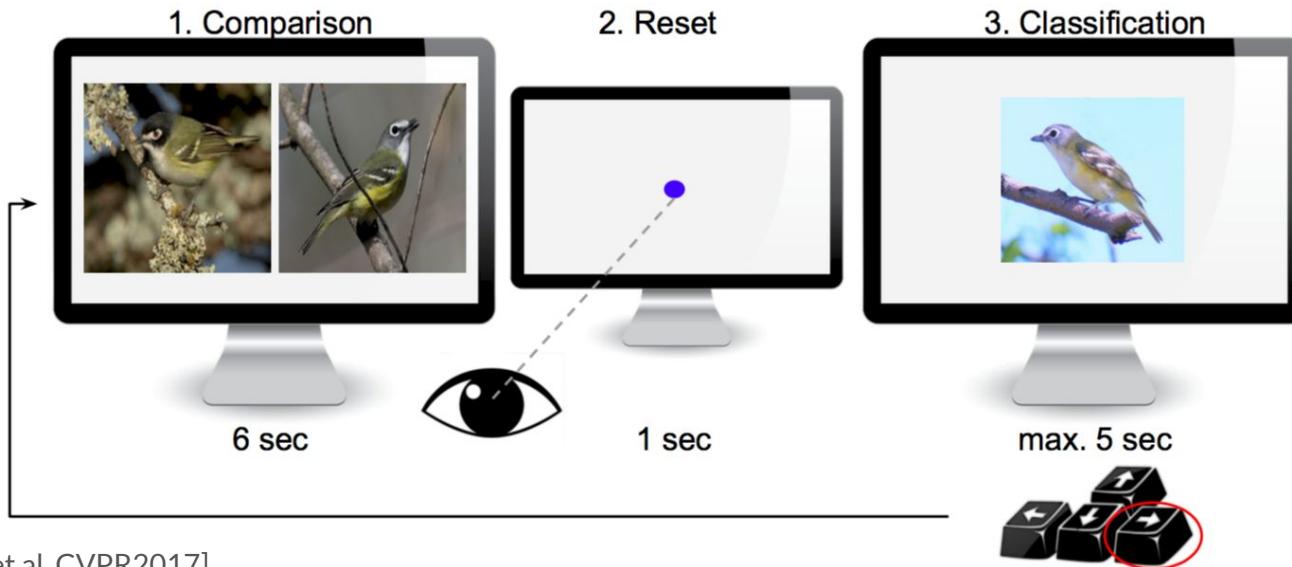
Side information - Gaze embedding

Experiment



Side information - Gaze embedding

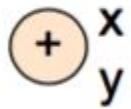
Experiment



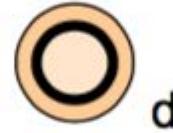
Side information - Gaze embedding

Features

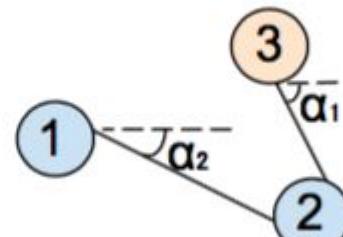
Location



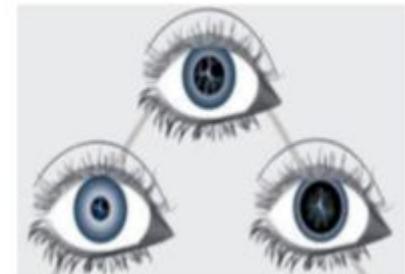
Duration



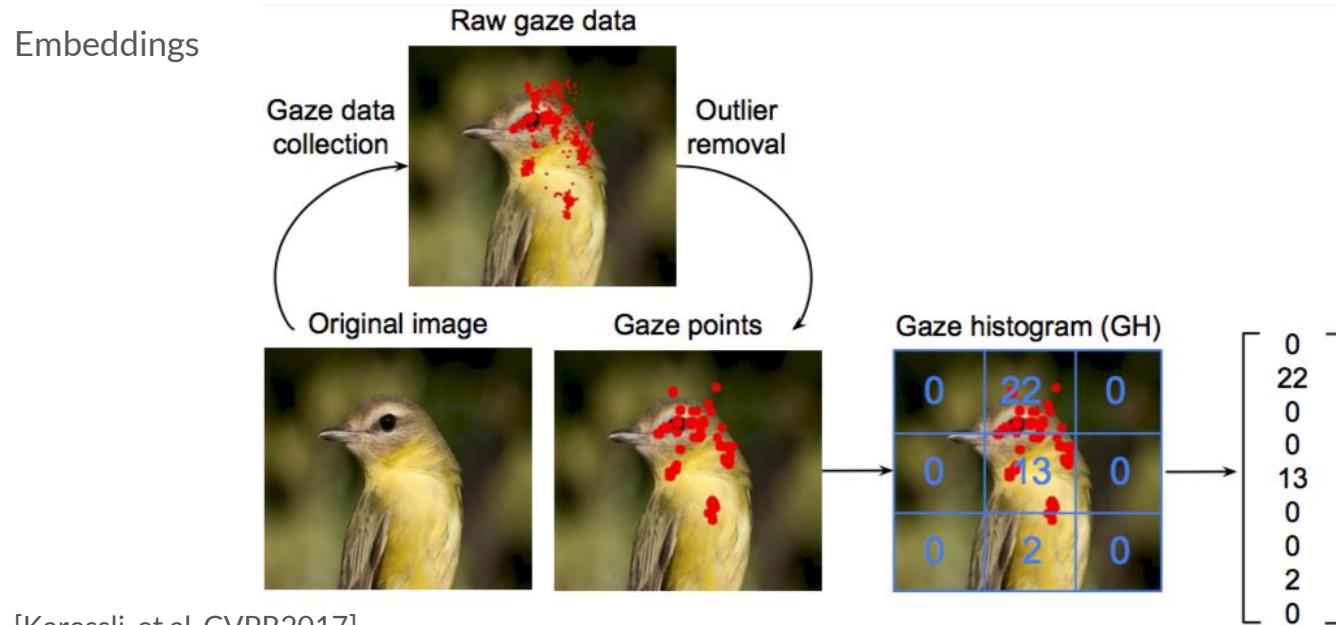
Sequence



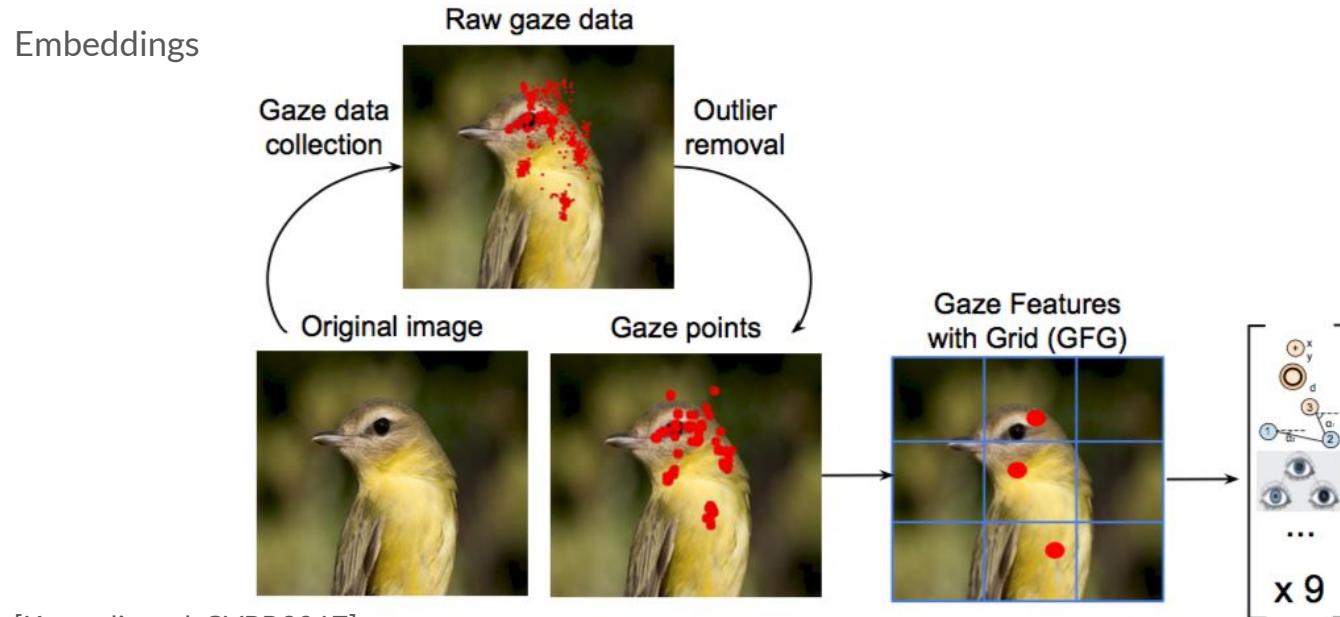
Pupil Diameter



Side information - Gaze embedding

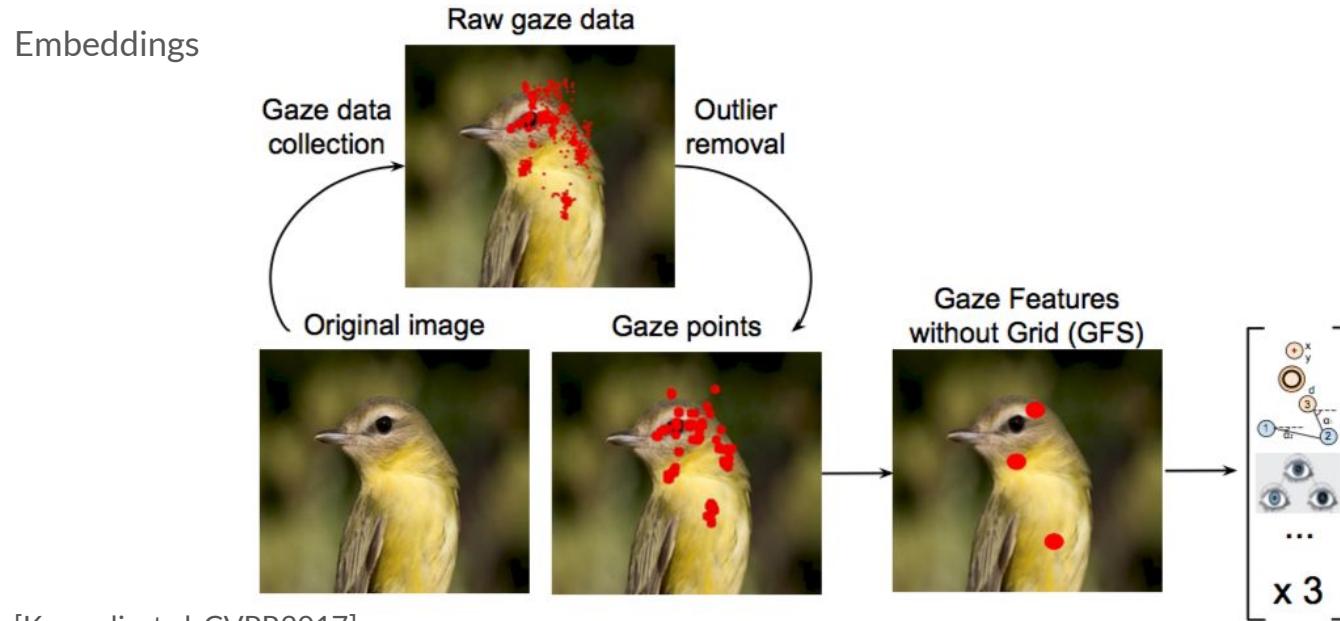


Side information - Gaze embedding

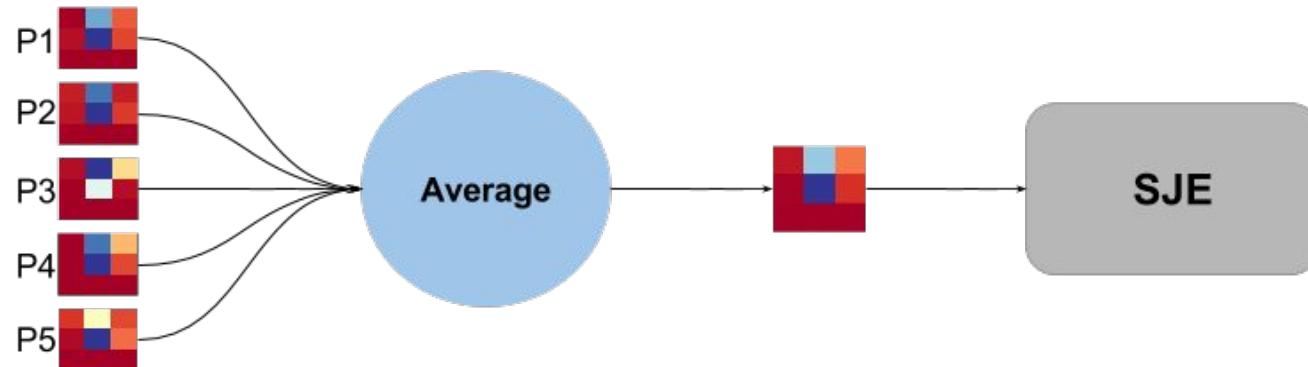


[Karelli, et al. CVPR2017]

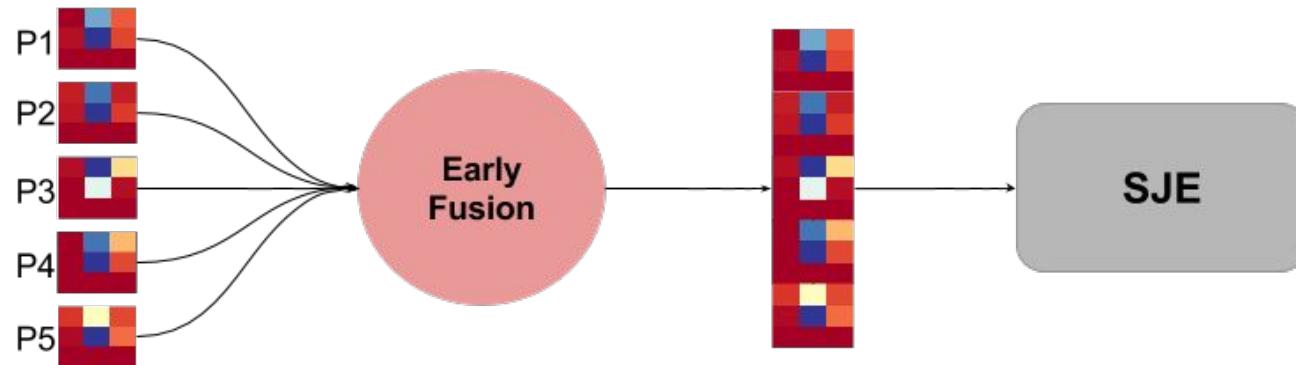
Side information - Gaze embedding



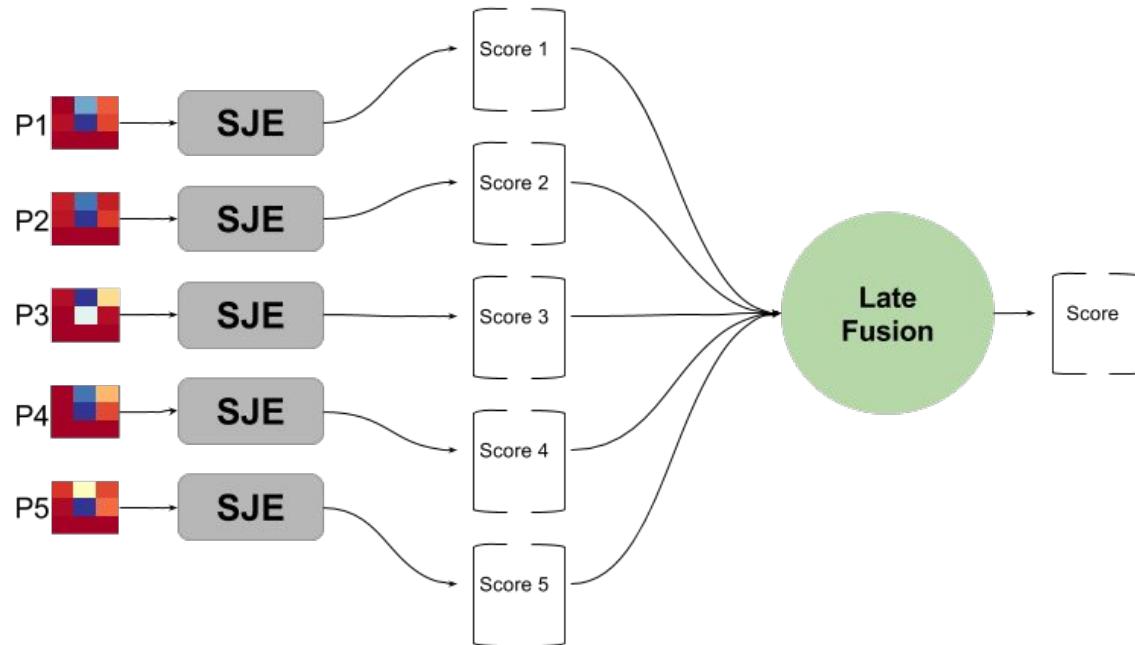
Side information - Gaze embedding



Side information - Gaze embedding



Side information - Gaze embedding



Side information summary

- Supervised
 - Expert
 - Class attributes
 - Novice
 - Detailed visual descriptions (deep representations of visual descriptions)
 - Human gaze
- Unsupervised
 - Hierarchical similarity
 - Text embeddings

CUB birds

- 11,788 images
- 200 bird species, 150 train+val set and 50 test classes



[Welinder, et al. 2010]

Results

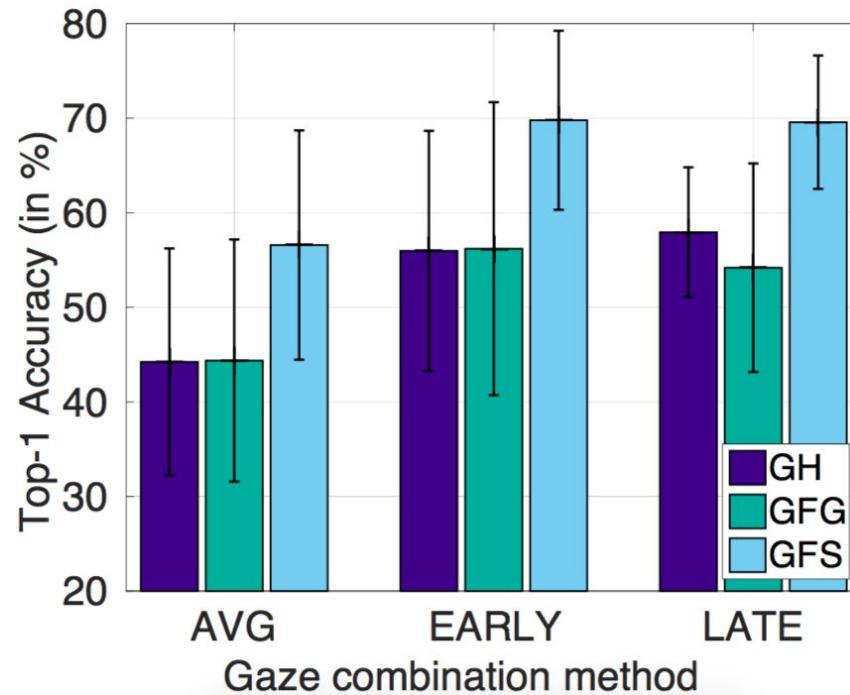
Source	Side information	Accuracy
Text	Hierarchical	20.6
	Bag of Words	22.1
	Word2vec	28.4
	Glove	24.2
Expert annotator	Attributes	50.1
Novice annotator	Detailed visual descriptions	56.8

Mini-CUB birds

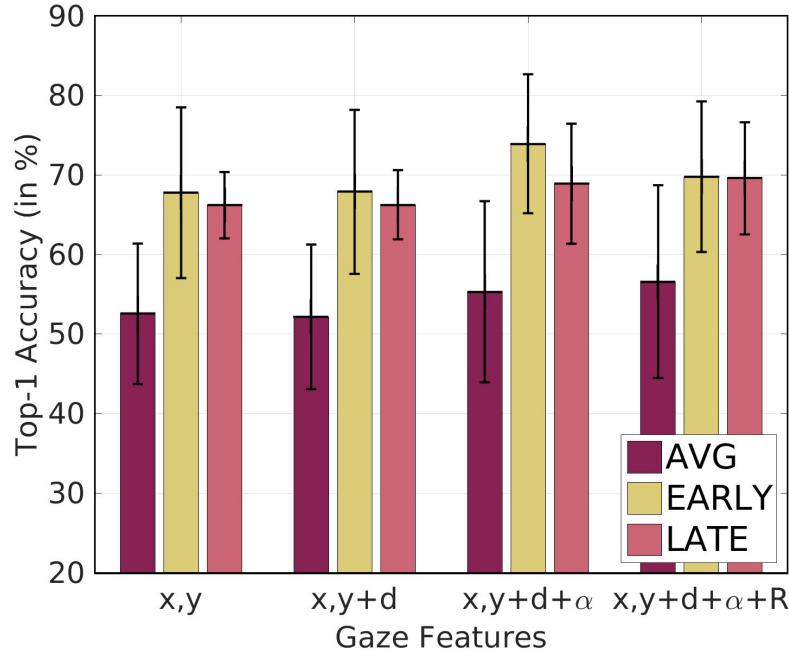
- 464 images
- 14 bird species, 11 train+val set and 3 test classes



Results



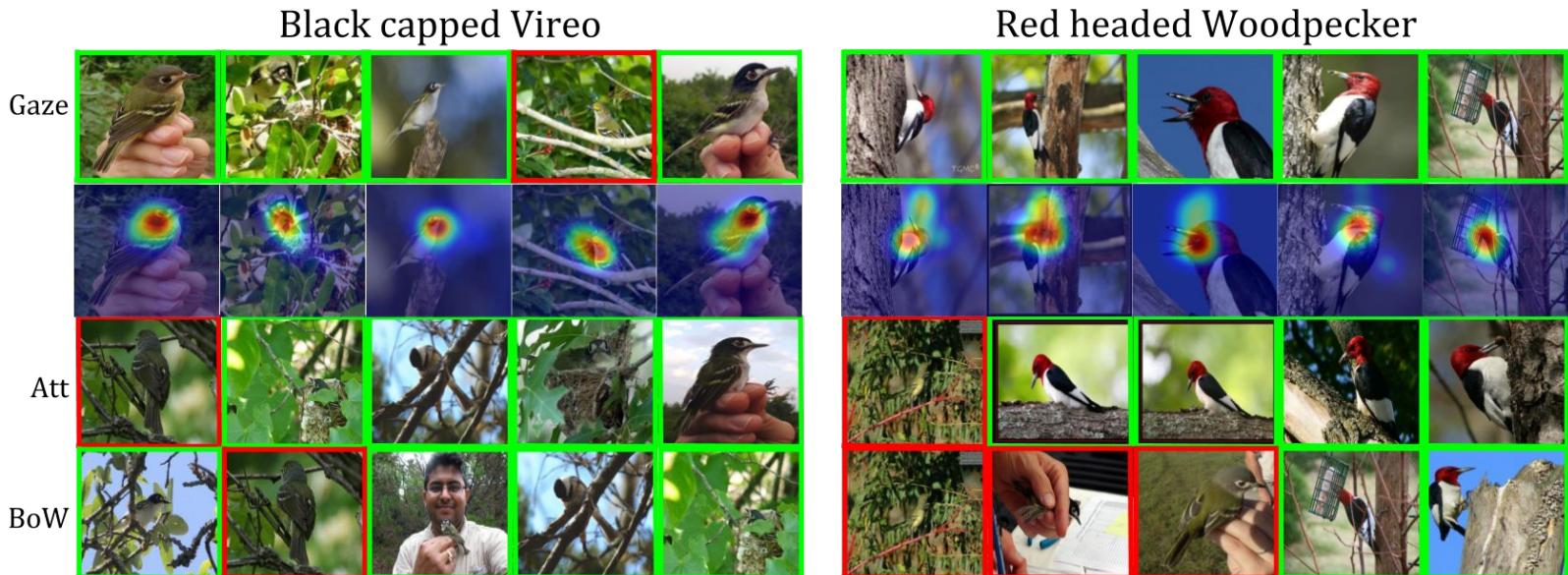
Results



Results

Method	Accuracy
Bag-of-Words from Wiki	55.2
Human annotated attributes	72.9
Gaze embeddings	73.9
Attributes + Gaze	78.2

Results

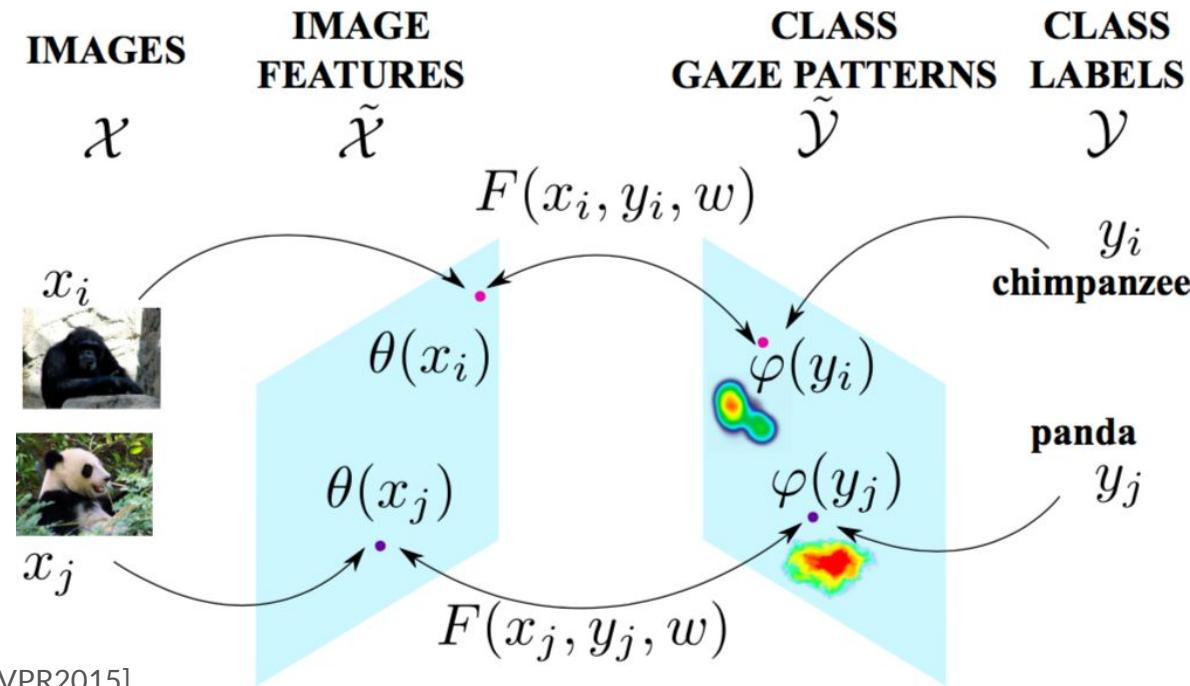




Structure

- Introduction & motivation
- Zero-shot learning
 - Definition
 - Side information
 - Zero-shot learning models
 - Exercise
- Low-shot learning
 - Definition
 - Low-shot learning models
- Tips & tricks
- Exercises

Zero-shot models



Task formulation

Training set $S = \{(x_n, y_n), n = 1..N\}, y_n \in Y^{train}$

We want to learn a function $f : X \rightarrow Y$ by minimize the empirical risk:

$$\frac{1}{N} \sum_{n=1}^N L(y_n, f(x_n; W)) + \Omega(W)$$

- Loss function
- Regularization term

Task formulation

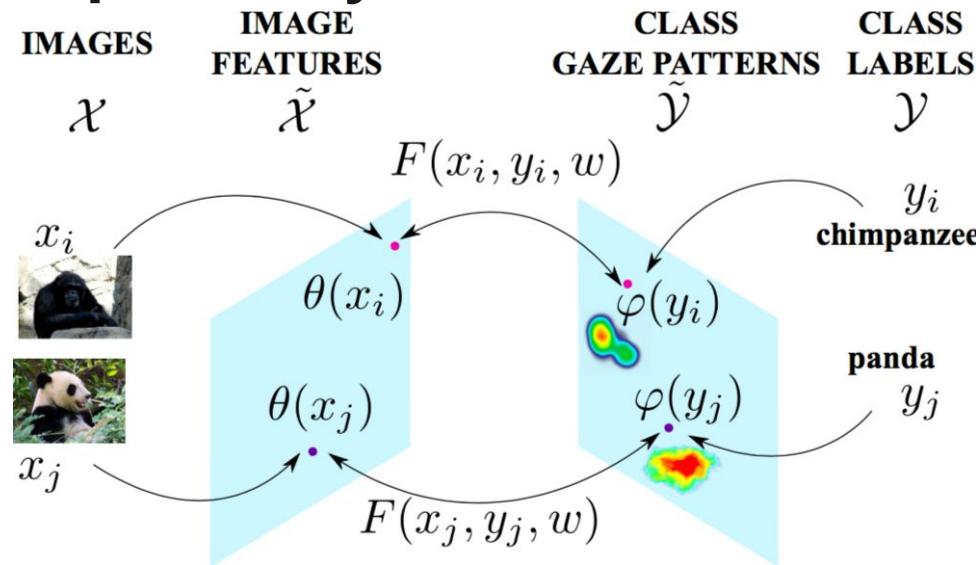
Training set $S = \{(x_n, y_n), n = 1..N\}, y_n \in Y^{train}$

Maximizing the compatibility:

$$f(x_n; W) = \operatorname{argmax}_{y \in Y} F(x, y; W)$$

At test time we predict $y \in Y^{test}$ that gives the highest compatibility $Y^{test} \subset Y$

Linear Compatibility



$$F(x, y; W) = \theta(x)^T W \phi(y)$$

Linear Compatibility - DEVISE

Deep visual semantic embedding, pairwise ranking

$$\sum_{y \in \mathcal{Y}^{tr}} [\Delta(y_n, y) + F(x_n, y; W) - F(x_n, y_n; W)]_+$$

$$\Delta(y_n, y) = \begin{cases} 0, & \text{if } y_n = y. \\ 1, & \text{otherwise.} \end{cases}$$

Linear Compatibility - ALE

Attribute label embedding, weighted pairwise ranking

$$\sum_{y \in y^{tr}} l_k [\Delta(y_n, y) + F(x_n, y; W) - F(x_n, y_n; W)]_+$$

$$l_k = \sum_{i=1}^k \alpha_i \quad \text{where} \quad \alpha_i = 1/i$$

Linear Compatibility - SJE

Structured joint embedding, multiclass objective

$$[\max_{y \in \mathcal{Y}^{tr}} (\Delta(y_n, y) + F(x_n, y; W)) - F(x_n, y_n; W)]_+$$

Linear Compatibility - ESZSL

Embarrassingly simple zero-shot learning, adds regularization terms

$$\gamma \| W\phi(y) \|^2 + \lambda \| \theta(x)^T W \|^2 + \beta \| W \|^2$$

γ, λ, β regularization parameters

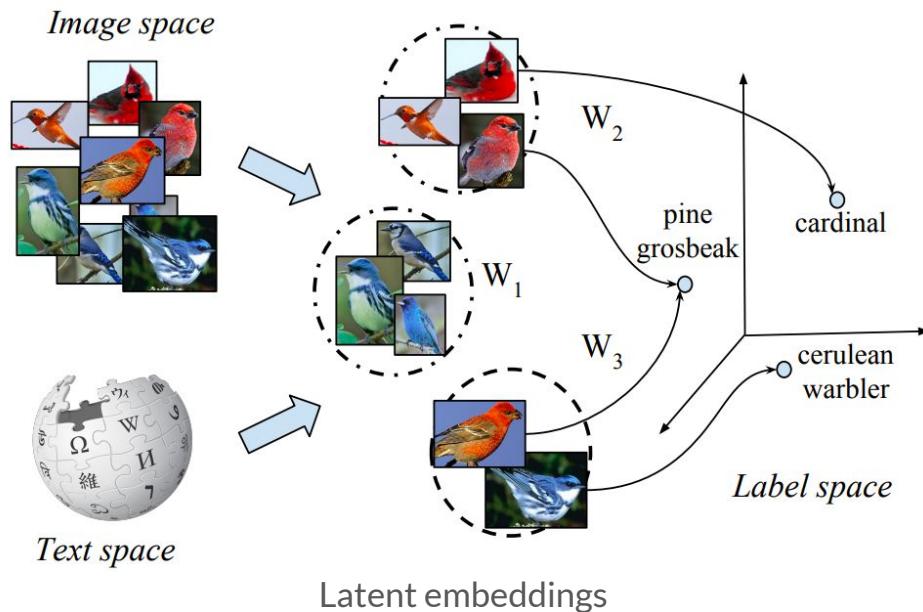
Linear Compatibility - SAE

Semantic autoencoder, linear objective

$$\min_W \| \theta(x) - W^T \phi(y) \|^2 + \lambda \| W\theta(x) - \phi(y) \|^2$$

- Autoencoder learns projection from image features to label embedding
- The autoencoder must reconstruct original image features

Nonlinear Compatibility - LATEM



Nonlinear Compatibility - LATEM

Latent embeddings, piecewise linear compatibility

$$F(x, y; W_i) = \max_{1 \leq i \leq K} \theta(x)^T W_i \phi(y)$$

- Support non-linearity
- Different W encodes different characteristics

Nonlinear Compatibility - CMT

Cross modal transfer, nonlinear compatibility using two layers NN

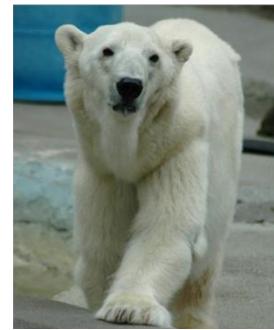
$$\sum_{y \in \mathcal{Y}^{tr}} \sum_{x \in \mathcal{X}_y} \|\phi(y) - W_1 \tanh(W_2 \cdot \theta(x))\|$$

Intermediate Classifier - DAP

Direct attribute prediction

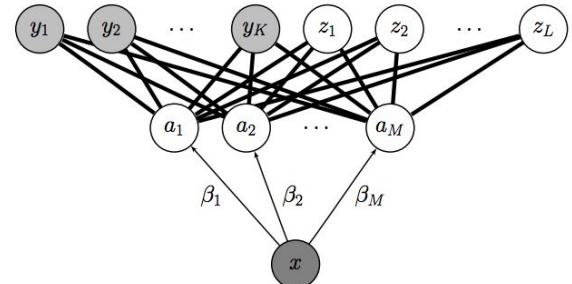
1. Learning probabilistic classifiers for each attribute
2. Combie scores

$$f(x) = \operatorname{argmax}_c \prod_{m=1}^M \frac{p(a_m^c | x)}{p(a_m^c)}$$



Polar bear

black: no
white: yes
brown: no
stripes: no
water: yes
eats fish: yes



Intermediate Classifier - CONSE

Convex combination of semantic embedding

$$f(x, t) = \operatorname{argmax}_{y \in \mathcal{Y}^{tr}} p_{tr}(y|x)$$

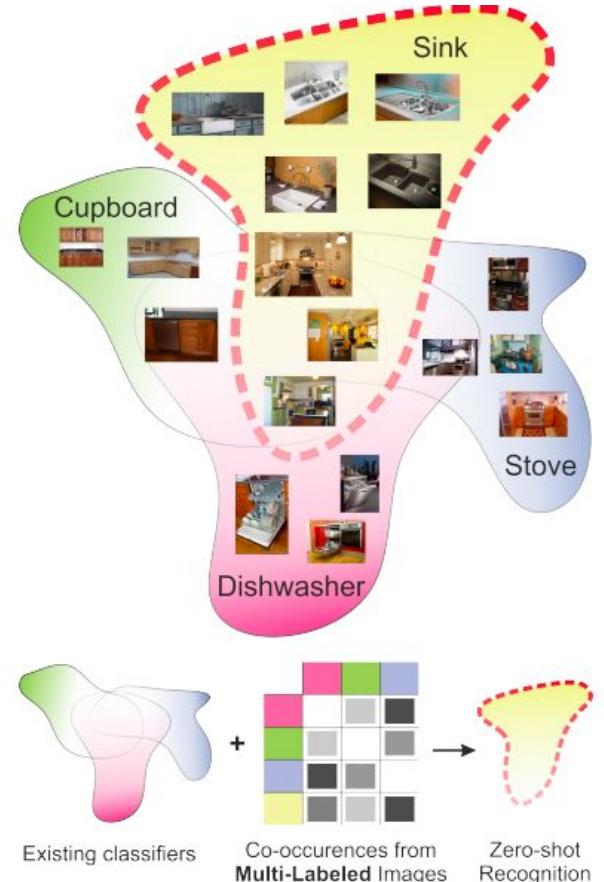
At test time, uses combination of semantic embeddings

$$\frac{1}{Z} \sum_{i=1}^T p_{tr}(f(x, t)|x).s(f(x, t))$$

Intermediate Classifier - COSTA

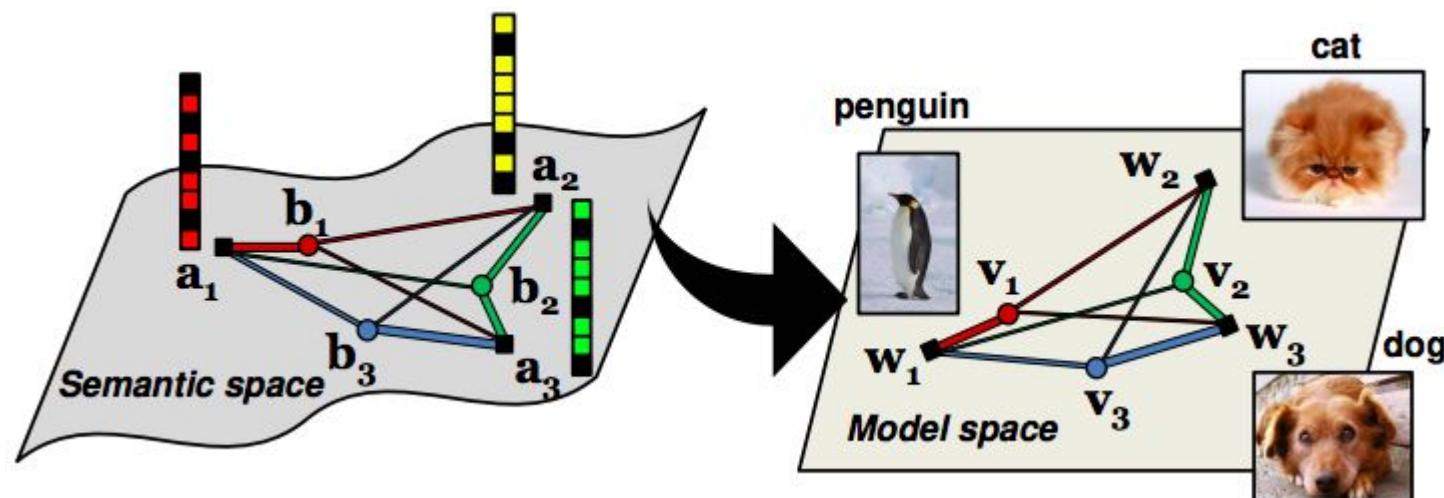
Co-occurrence statistics of visual concepts
between seen and unseen classes

$$\hat{\mathbf{w}}_l = \sum_k \mathbf{w}_k s_{lk}$$



Hybrid models - SYNC

Synthesized classifier, minimize distortion error



Hybrid models - SYNC

Synthesized classifier, minimize distortion error

$$\min_{w_c, v_r} \|w_c - \sum_{r=1}^R s_{cr} v_r\|_2^2$$



Models summary

1. **Linear compatibility**
DEVISE [Frome, et al. NIPS2013], ALE [Akata, et al. CVPR2015], SJE [Akata, et al. CVPR2015],
ESZSL [Romera-Paredes and Torr ICML2015], SAE [Kodirov, et al. CVPR2017]
2. **Nonlinear compatibility**
LATEM [Xian, et al. CVPR2016], CMT [Socher, et al. NIPS2013]
3. **Intermediate classifier**
DAP [Lampert, et al. CVPR2009], CONSE [Norouzi, et al. ICLR2014], COSTA [Mensink, et al.
CVPR2014]
4. **Hybrid models**
SYNC [Changpinyo, et al. CVPR2016]



AWA dataset

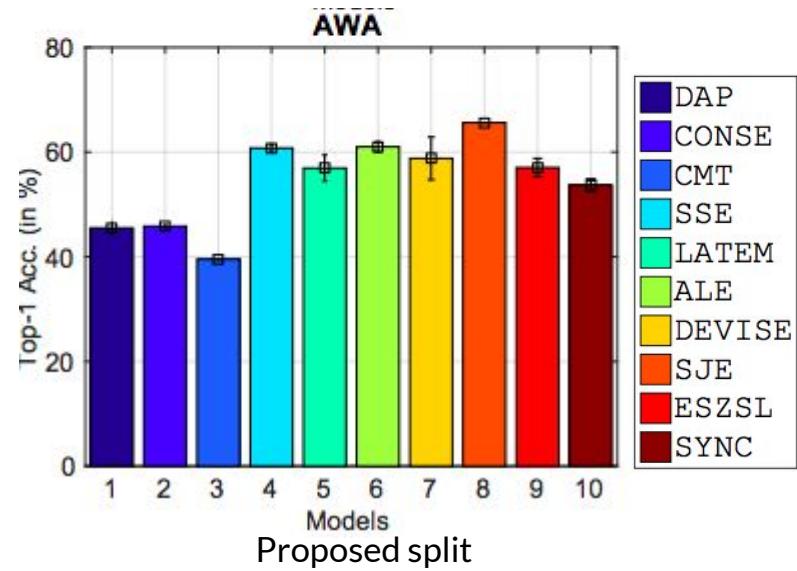
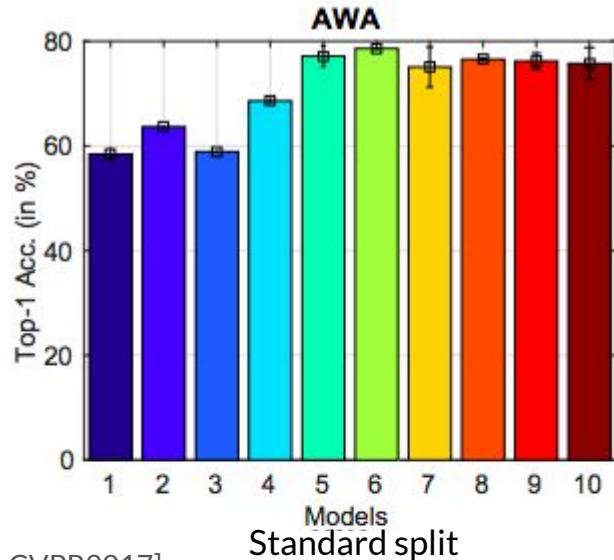
Animal with attributes dataset

- 30K images
- 50 classes
- 85 attribute
- Standard split 40 train+val 10 test
- Suggested split

Insures that none of the test classes is used to train the image features base model

Results

Evaluation on Animals with Attributes AWA





Q & A



Structure

- Introduction & motivation
- Zero-shot learning
 - Definition
 - Side information
 - Zero-shot learning models
 - Exercise
- Low-shot learning
 - Definition
 - Low-shot learning models
- Tips & tricks
- Exercises



Zero-shot learning exercise

SJE implementation

- Select dataset (AWA)
- Download data (image features & class embeddings)
- Implement zero-shot algorithm SJE
- Evaluate Top-1 accuracy on test set



Low-shot Learning



Structure

- Introduction & motivation
- Zero-shot learning
 - Definition
 - Side information
 - Zero-shot learning models
 - Exercise
- Low-shot learning
 - Definition
 - Low-shot learning models
- Tips & tricks
- Exercises

Go back to tail distribution..



<https://www.cars.com/>

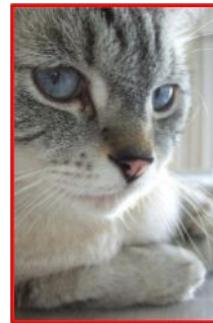
<http://www.foxnews.com/lifestyle/2017/11/09/how-to-keep-cat-from-scratching-your-sofa-to-shreds.html>

<https://www.livescience.com/55223-capybara-facts.html>

<https://www.indiamart.com/proddetail/hand-wrench-13045857897.html>

Low-shot learning

- Ability to generalize only with a few examples
- Exploits prior learning on other classes





Structure

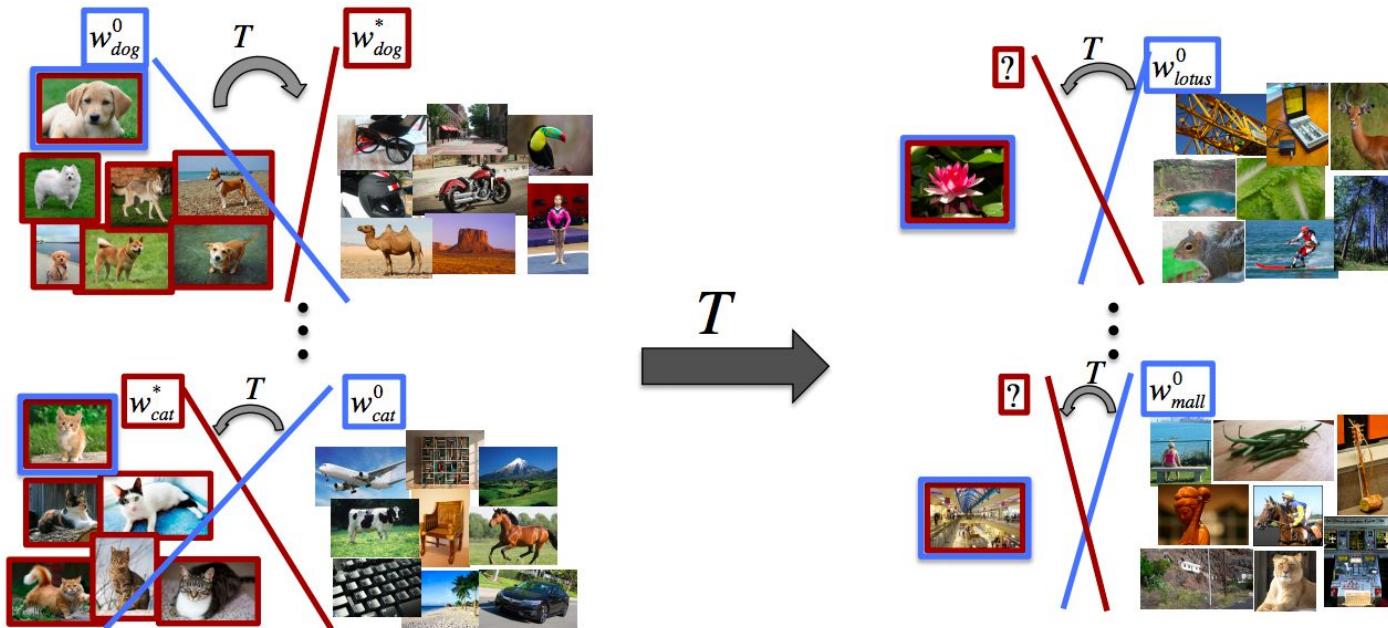
- Introduction & motivation
- Zero-shot learning
 - Definition
 - Side information
 - Zero-shot learning models
 - Exercise
- Low-shot learning
 - Definition
 - Low-shot learning models
- Tips & tricks
- Exercises

Low-shot learning approaches

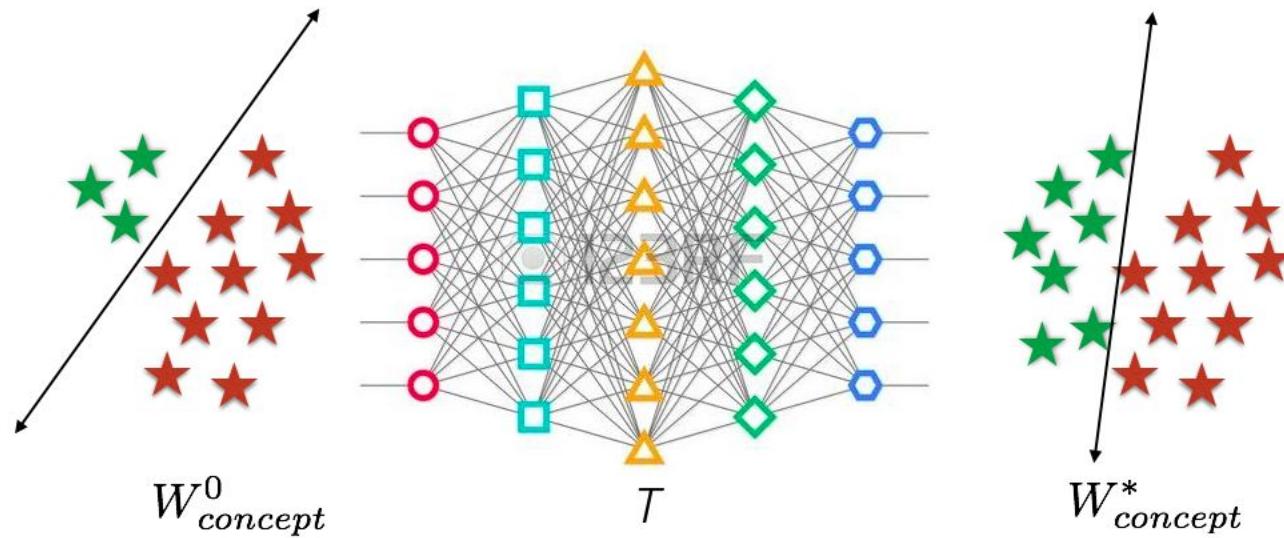
We will overview three recent works on the problem of low-shot image classification

- Learning to learn
- Matching nets
- Shrinking and Hallucinating Features

Learning to learn



Learning to learn



Learning to learn

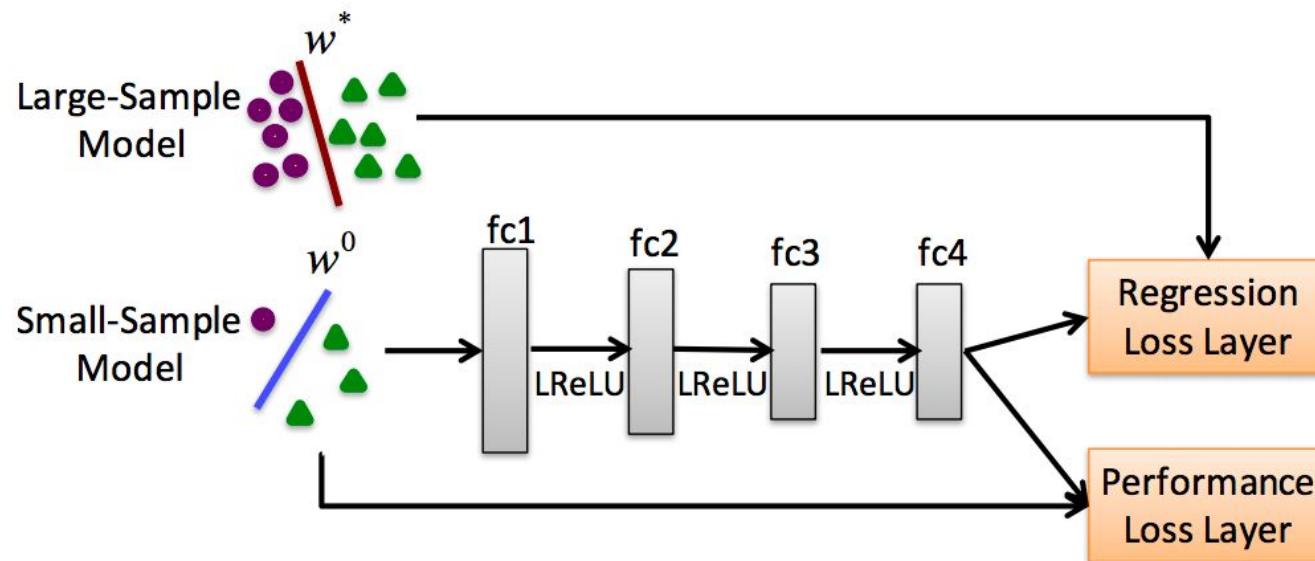
Loss function

$$L(\Theta) = \sum_{j=1}^J \left\{ \underbrace{\frac{1}{2} \| \mathbf{w}_j^* - T(\mathbf{w}_j^0, \Theta) \|_2^2}_{\text{Model regression term}} + \lambda \underbrace{\sum_{i=1}^{M+N} \left[1 - y_i^j \left(T(\mathbf{w}_j^0, \Theta)^T \mathbf{x}_i^j \right) \right]_+}_{\text{Data fitting term}} \right\}$$

Model regression term
Euclidean distance

Data fitting term
Hinge loss

Learning to learn





Learning to learn

Novel categories:

- **Initialization**
learn model from small set of K (image,label) pairs
- **Transformation**
perform the learned transformation T
- **Refinement**
retrain SVM using the transformed model as regularizer

$$R(\mathbf{w}) = \frac{1}{2} \|\mathbf{w} - T(\mathbf{w}^0, \Theta)\|_2^2 + \eta \sum_{i=1}^K [1 - y_i (\mathbf{w}^T \mathbf{x}_i)]_+$$

Matching Networks

Given a support set $S = \{(x_i, y_i)\}_{i=1}^k$ learns the mapping $S \rightarrow C_S(x)$

The classifier defines the probability distribution, P is parameterized by a neural network:

$$C(x^{test}) = P(y^{test}|x^{test}, S)$$

Prediction:

$$\text{argmax}_y P(y|x^{test}, S)$$



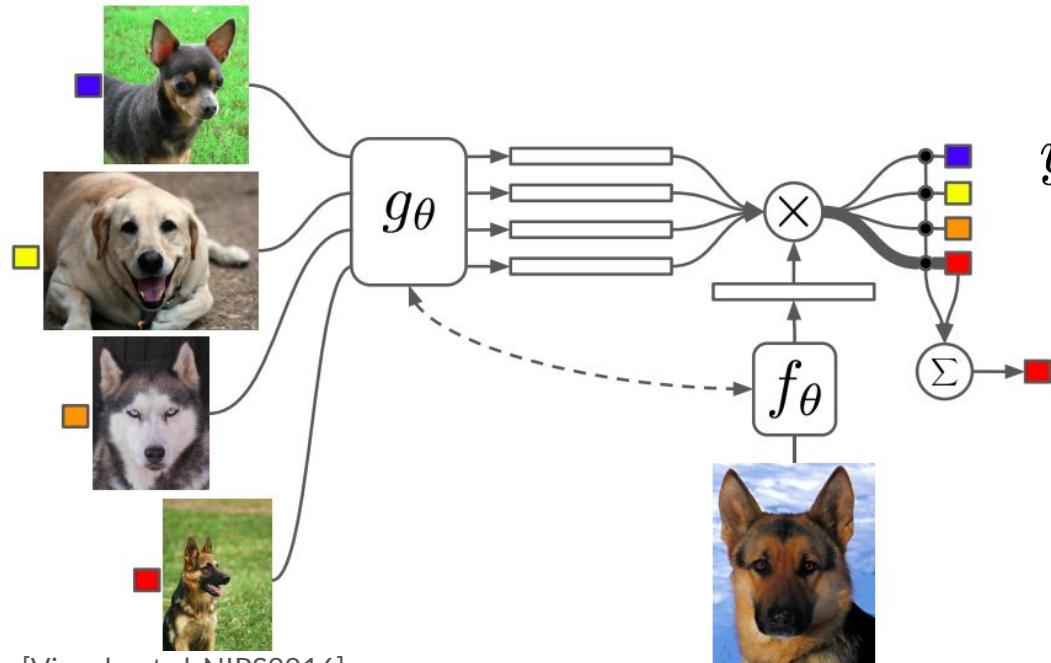
Matching Networks

Prediction

$$y^{test} = \sum_{i=1}^k a(x^{test}, x_i) y_i$$

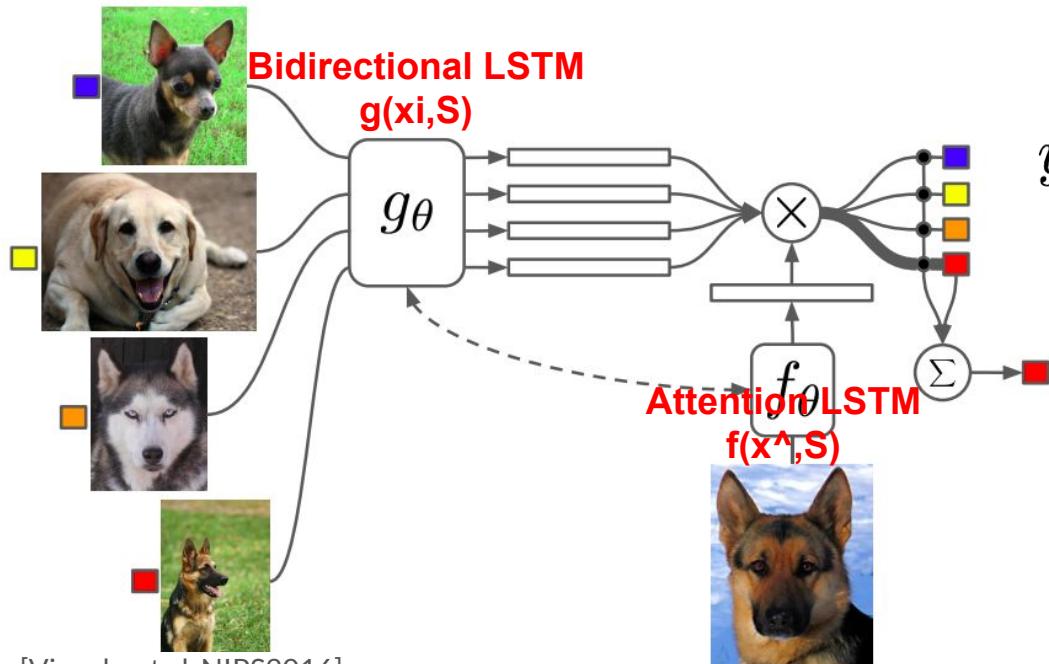
- Attention mechanism
- Linear combination of support set labels

Matching Networks



$$y^{test} = \sum_{i=1}^k a(x^{test}, x_i) y_i$$

Matching Networks



$$y^{test} = \sum_{i=1}^k a(x^{test}, x_i) y_i$$

0.1*Chihuahua
0.1*Labrador Retriever
0.5*German Shepherd
0.3*Siberian Husky

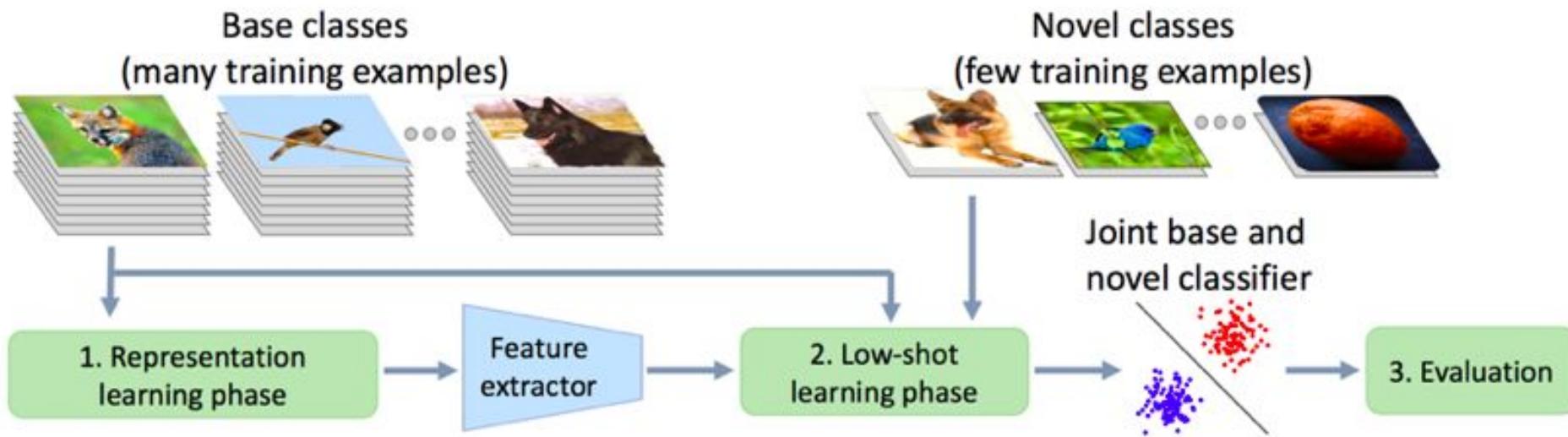
Matching Networks

Training strategy

1. Sample task T (5 labels, up to 5 examples per label)
2. Sample a label set L from T e.g. {cats, dogs}
3. Sample a support set S examples from L
4. Sample batch B examples from L
5. Evaluate loss on B using S

$$\theta = \underset{\theta}{\operatorname{argmax}} E_{L \sim T} [E_{S \sim L, B \sim L} [\sum_{(x,y) \in B} \log P_{\theta}(y|x, S)]]$$

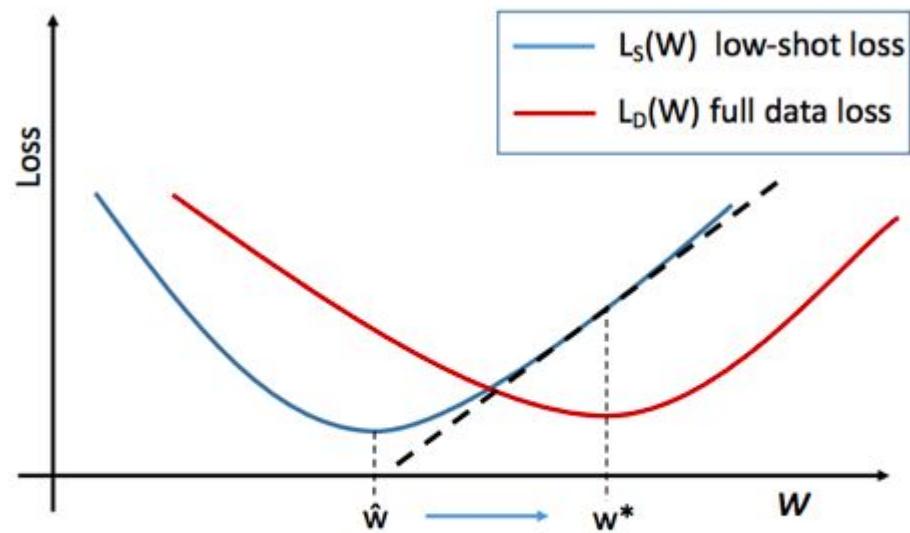
Shrinking and Hallucinating Features



Shrinking and Hallucinating Features

Introduces Squared Gradient Magnitude loss

- Minimise the loss of low-shot during representation learning
- better representation for low-shot learning



Shrinking and Hallucinating Features

Train feature extractor and classifier on D (all data) has the objective

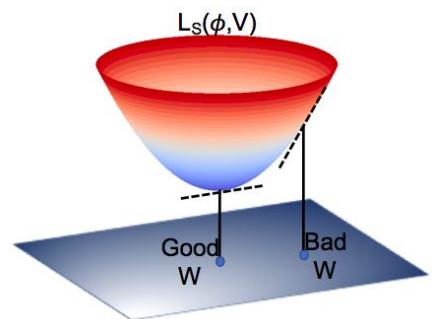
$$\min_{W, \phi} L_D(\phi, W) = \min_{W, \phi} \frac{1}{|D|} \sum_{(x,y) \in D} L_{cls}(W, \phi(x), y)$$

For small set S, the objective

$$\min_V L_S(\phi, V) = \min_V \frac{1}{|S|} \sum_{(x,y) \in S} L_{cls}(V, \phi(x), y)$$

Minimise

$$\tilde{L}_S(\phi, W) = \|\nabla_V L_S(\phi, V)|_{V=W}\|^2$$





Shrinking and Hallucinating Features

$$\begin{aligned}\tilde{L}_S(\phi, W) &= \sum_{k=1}^K (p_k(W, \phi(x)) - \delta_{yk})^2 \|\phi(x)\|^2 \\ &= \alpha(W, \phi(x), y) \|\phi(x)\|^2.\end{aligned}$$

$\alpha(W, \phi(x), y)$ Per example weight that is higher for misclassified data points

Final SGM loss

$$L_D^{SGM}(\phi, W) = \frac{1}{|D|} \sum_{(x,y) \in D} \alpha(W, \phi(x), y) \|\phi(x)\|^2$$



Shrinking and Hallucinating Features

Train feature representation by minimizing a linear combination of the SGM loss and the original classification objective

$$\min_{W, \phi} L_D(\phi, W) + \lambda L_D^{SGM}(\phi, W)$$

Shrinking and Hallucinating Features

Hallucinate samples



Assumption

perched bird with sky background

perched bird with green background

Any two examples z_1 and z_2 belonging to the same category represent a plausible transformation.

→ Given a novel category example x , apply to x the transformation that sent z_1 to z_2 .



Shrinking and Hallucinating Features

Fully supervised regression using MLP of 3 fully connected layers



Shrinking and Hallucinating Features

Fully supervised regression using MLP of 3 fully connected layers

1. Cluster feature vectors in each category into 100 clusters.



Shrinking and Hallucinating Features

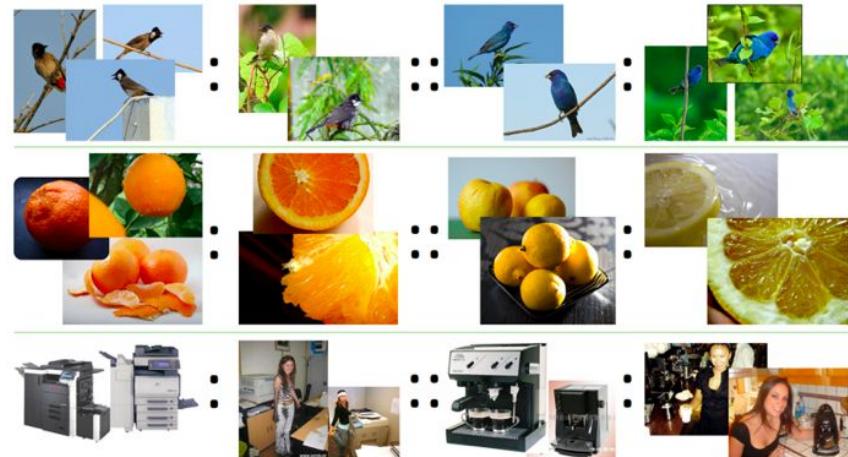
Fully supervised regression using MLP of 3 fully connected layers

1. Cluster feature vectors in each category into 100 clusters.
2. Form quadruple of centroids

Shrinking and Hallucinating Features

Fully supervised regression using MLP of 3 fully connected layers

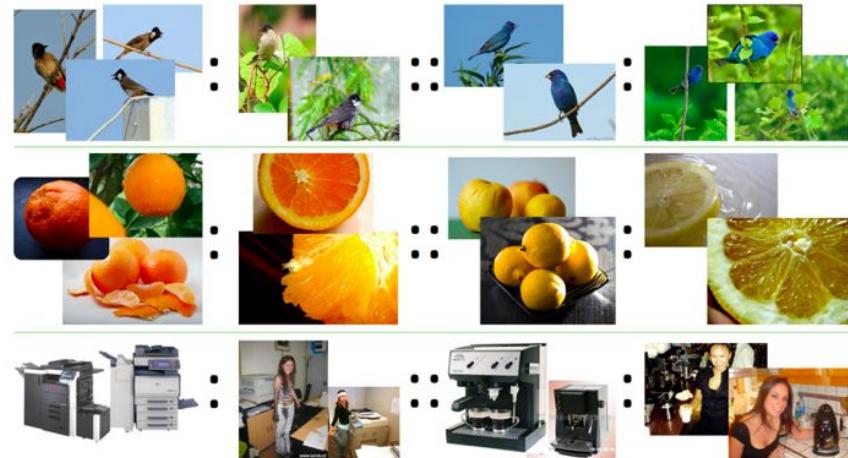
1. Cluster feature vectors in each category into 100 clusters.
2. Form quadruple of centroids
3. Feed 3 centroids and predict the forth



Shrinking and Hallucinating Features

Fully supervised regression using MLP of 3 fully connected layers

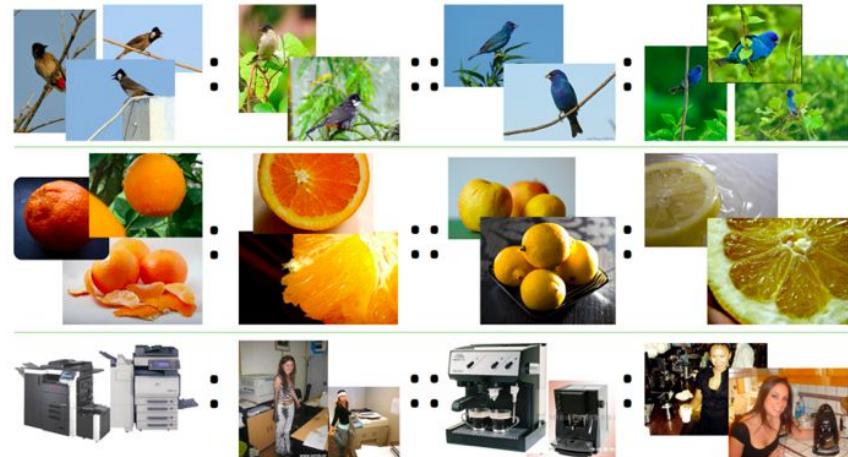
1. Cluster feature vectors in each category into 100 clusters.
2. Form quadruple of centroids
3. Feed 3 centroids and predict the forth
4. Minimize the weighted sum of two losses



Shrinking and Hallucinating Features

Fully supervised regression using MLP of 3 fully connected layers

1. Cluster feature vectors in each category into 100 clusters.
2. Form quadruple of centroids
3. Feed 3 centroids and predict the forth
4. Minimize the weighted sum of two losses
 - Classification loss
 - Mean squared error



Results

Representation	Lowshot phase	n=1	2	5	10	20
<i>ResNet-10</i>						
Baseline	Classifier	14.1	33.3	56.2	66.2	71.5
Baseline	Generation* + Classifier	29.7	42.2	56.1	64.5	70.0
SGM*	Classifier	23.1	42.4	61.7	69.6	73.8
SGM*	Generation* + Classifier	32.8	46.4	61.7	69.7	73.8
L2*	Classifier	29.1	47.4	62.3	68.0	70.6
Baseline	Model Regression [47]	20.7	39.4	59.6	68.5	73.5
Baseline	Matching Network [46]	41.3	51.3	62.1	67.8	71.8

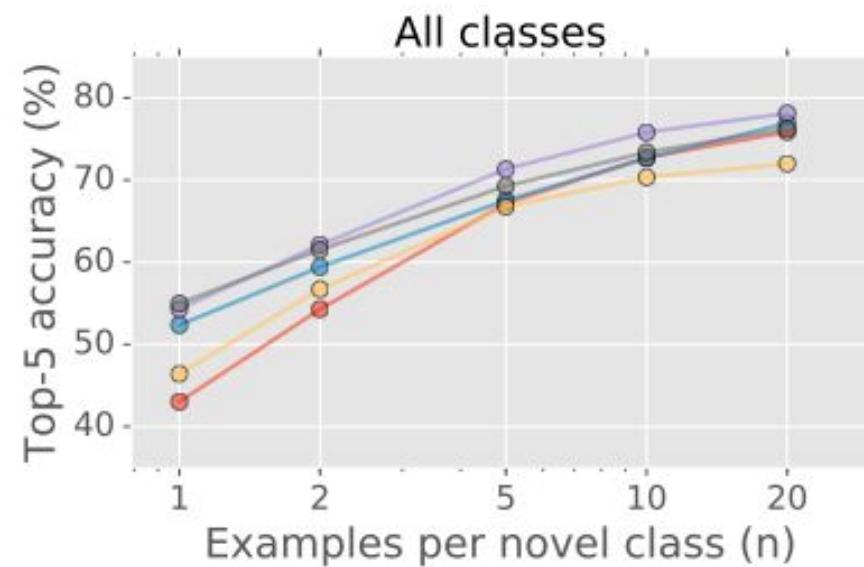
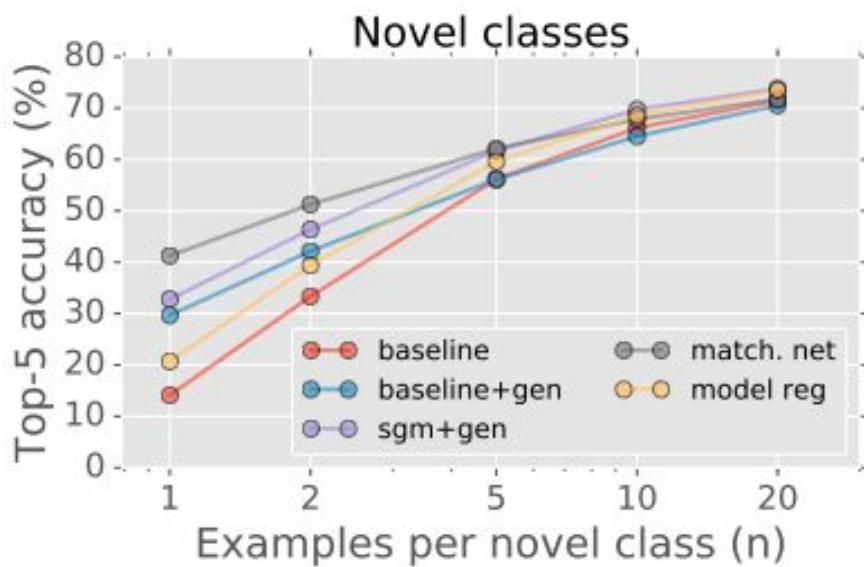
Top-5 accuracy on Imagenet1K for novel classes only

Results

Representation	Lowshot phase	n=1	2	5	10	20
<i>ResNet-10</i>						
Baseline	Classifier	43.0	54.3	67.2	72.8	75.9
Baseline	Generation* + Classifier	52.4	59.4	67.5	72.6	76.9
SGM*	Classifier	49.4	60.5	71.3	75.8	78.1
SGM*	Generation* + Classifier	54.3	62.1	71.3	75.8	78.1
L2*	Classifier	52.7	63.0	71.5	74.8	76.4
Baseline	Model Regression [47]	46.4	56.7	66.8	70.4	72.0
Baseline	Matching Network [46]	55.0	61.5	69.3	73.4	76.2

Top-5 accuracy on Imagenet1K for all classes

Results





Tips & Tricks

Structure

- Introduction & motivation
- Zero-shot learning
 - Definition
 - Side information
 - Zero-shot learning models
 - Exercise
- Low-shot learning
 - Definition
 - Low-shot learning models
- Tips & tricks
- Exercises

Generalization from small training set



Overfitting curse

Symptoms

- Very high training accuracy
- Very low testing accuracy

→ Model doesn't generalize to unseen data



Regularization

L_2 regularization

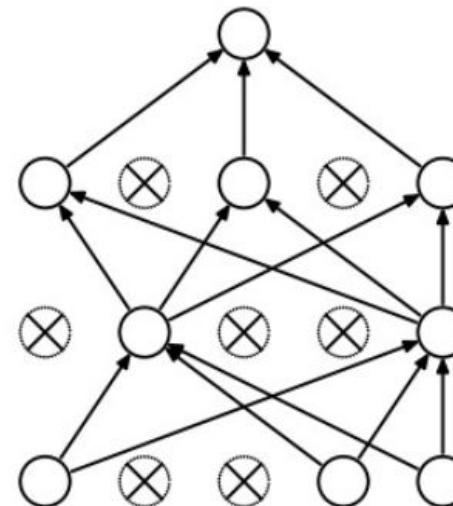
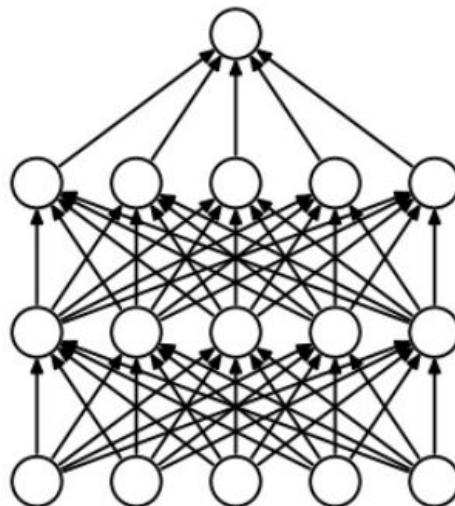
- Most common form of regularization
- Penalizing the squared magnitude of weights in the objective

L_1 regularization

- Relatively common form of regularization
- Penalizing L_1 of weights in the objective

Dropout

Removing a neuron from a designated layer during training or by dropping certain connection





Batch normalization

A common practice in NN, forces activations to have unit gaussian distribution

- Insert BN layer after FC and Conv, and before non-linearities

Robust networks to bad initialization

- interpreted as doing preprocessing at every layer of the network
- Normalization is differentiable



Transfer learning

Problem Training a model from scratch only with small data is **challenging** and suffer from **overfitting**



Transfer learning

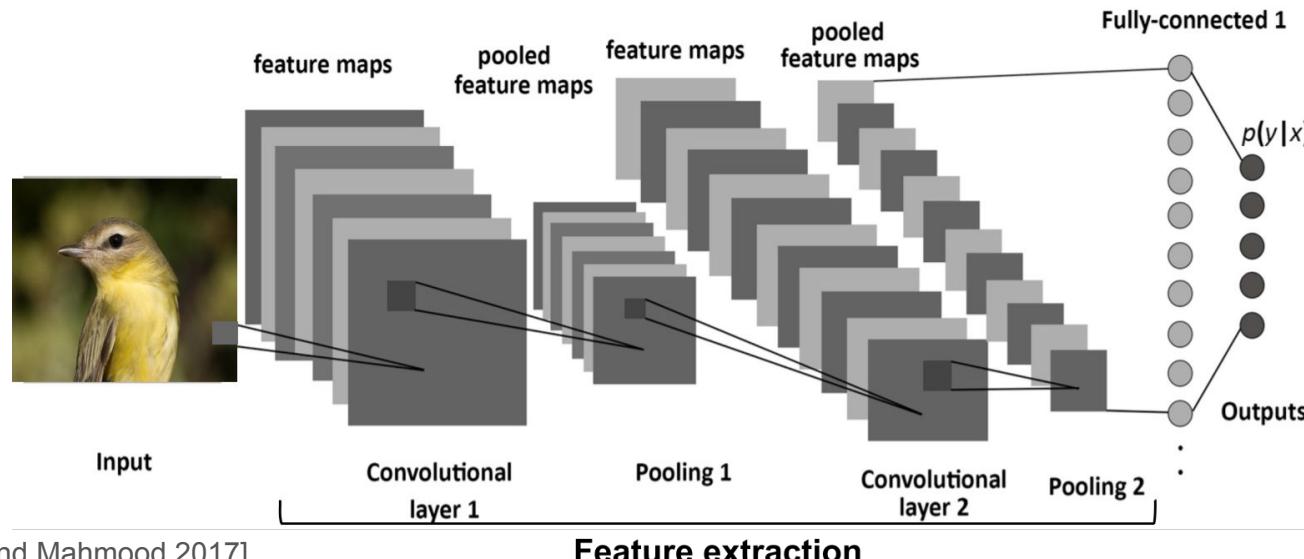
Problem Training a model from scratch only with small data is **challenging** and suffer from **overfitting**

Solution Use the knowledge of a pre-trained model on a border task to solve more specific one

- Bottle neck features
- Fine-tuning top layers

Transfer learning

Extract bottleneck features from a pre-trained network





Transfer learning

Finetune top layers of pre-trained network

- Remove top dense layers
- Add your own classification layers on top
- Freeze bottom layers
- Fine-tune top layers on small data

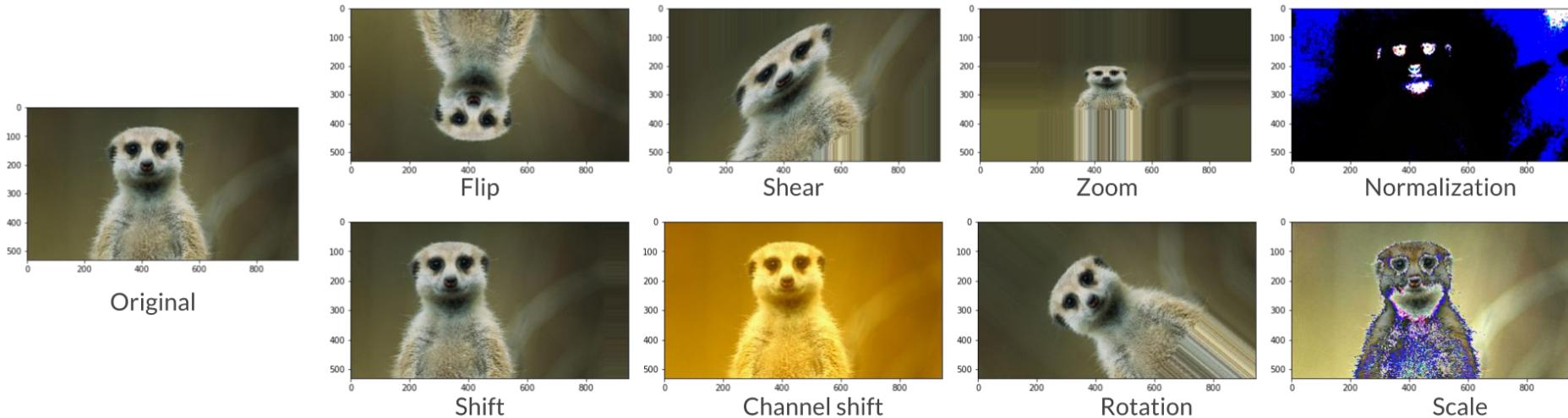


Data augmentation

Increase the amount of training data using information only in our training data

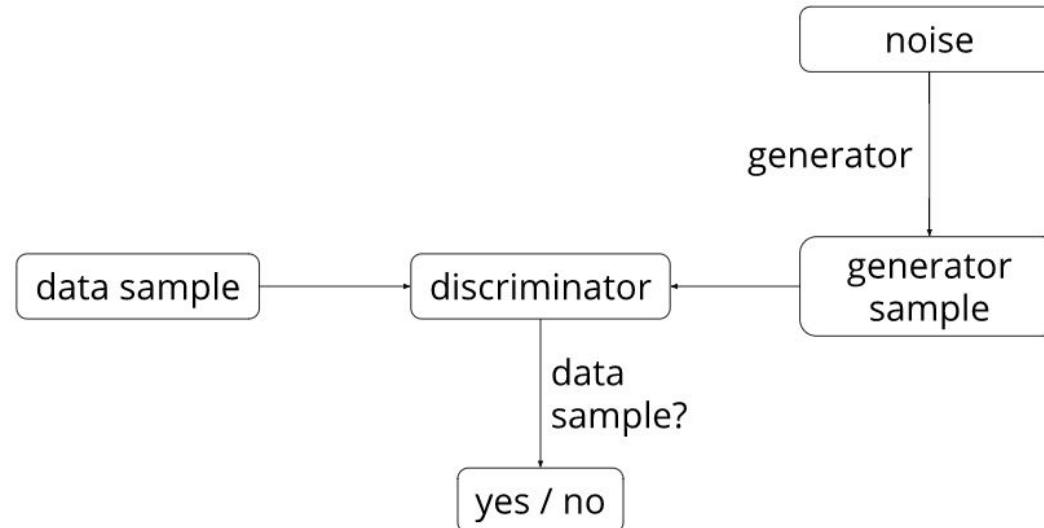
- Affine transformations
- Generative Adversarial Networks (**GANs**)

Affine transformations



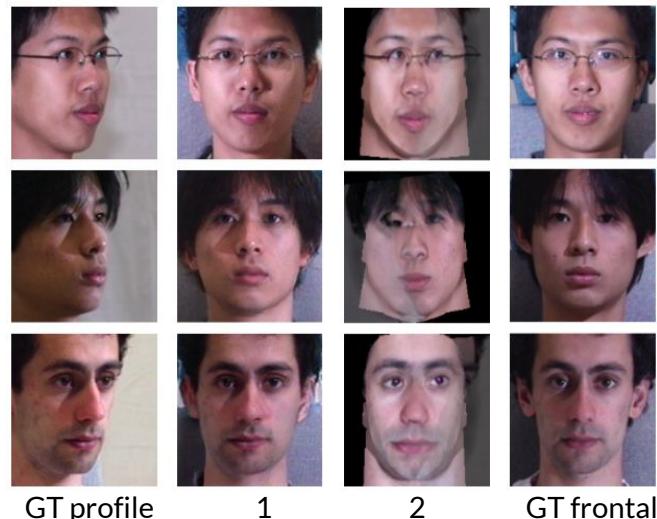
Generative Adversarial Networks (GANs)

Lean data distribution



Generative Adversarial Networks (GANs)

Frontal face generator



Generative Adversarial Networks (GANs)

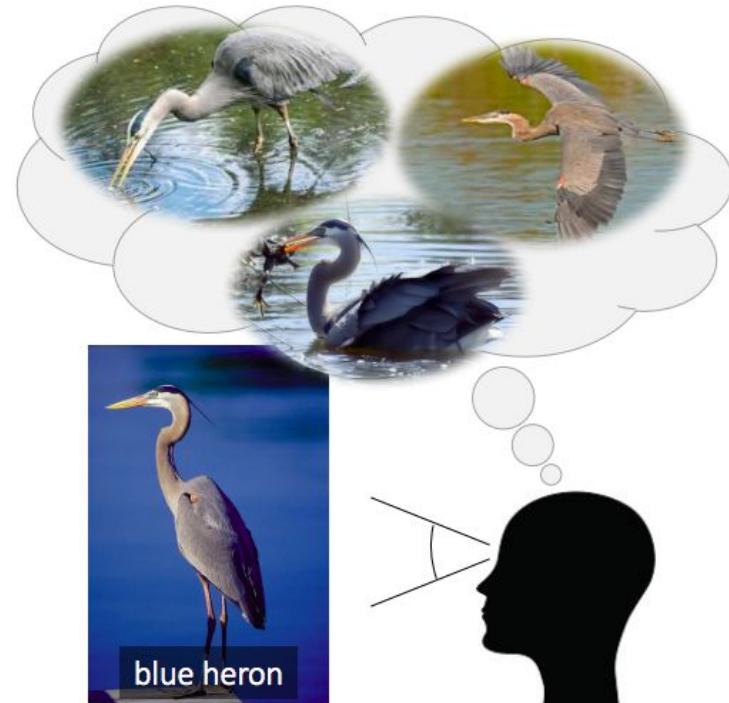
Image to image translation



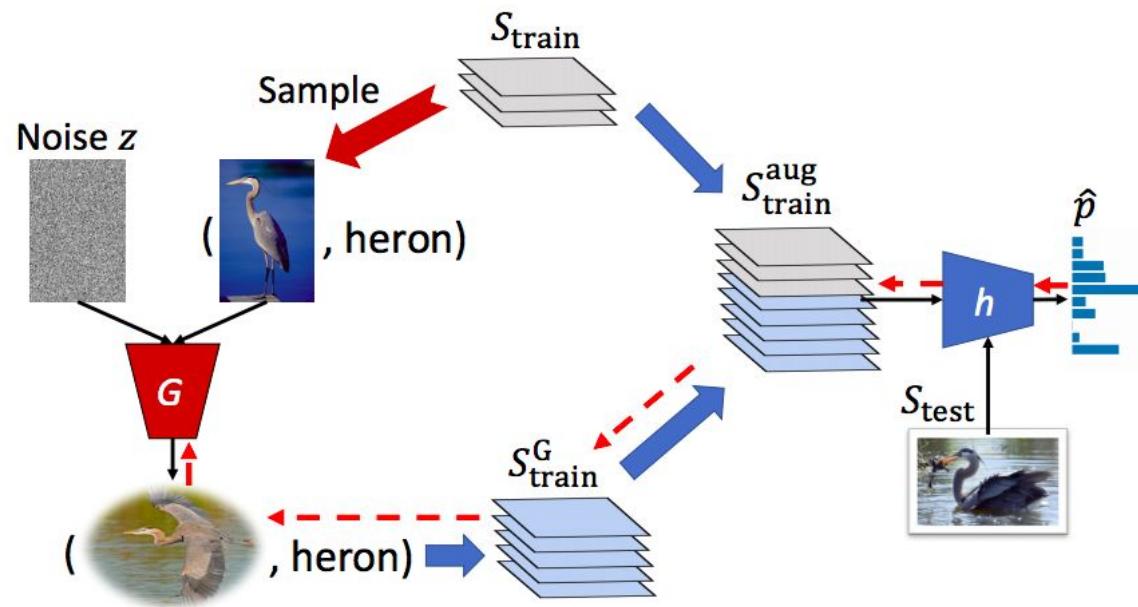
Low-shot learning + GANs

Low-Shot Learning from Imaginary Data

Meta-learning + Hallucination



Low-shot learning + GANs





Q & A

Structure

- Introduction & motivation
- Zero-shot learning
 - Definition
 - Side information
 - Zero-shot learning models
 - Exercise
- Low-shot learning
 - Definition
 - Low-shot learning models
- Tips & tricks
- Exercises



Image augmentation exercise



Image augmentation

- Clone git repo <https://github.com/Noura-kr/DSR.git>
- New notebook in DSR/notebooks/small_classifier/
- Read and plot meerkat.jpg
- Use keras keras.preprocessing.image.ImageDataGenerator to generate different images
- Plot results



Image classifier with small set exercise



Image classifier with small set

Train small network from scratch

- Clone git repo <https://github.com/Noura-kr/DSR.git>
- New notebook in DSR/notebooks/small_classifier/
- Define small conv net: 3 conv blocks (2Dconv,relu_activation,max_pooling) + 2 dense layers (don't forget flatten!)
- Compile network with binary_crossentropy loss and rmsprop
- Define image generator
- Train using generator
- Bonus: plot loss & accuracy

Image classifier with small set

Train small network from scratch

- Clone git repo <https://github.com/Noura-kr/DSR.git>
- New notebook in DSR/notebooks/small_classifier/
- Define small conv net: 3 conv blocks (2Dconv,relu_activation,max_pooling) + 2 dense layers (don't forget flatten!)
- Compile network with binary_crossentropy loss and rmsprop
- Define image generator
- Train using generator
- Bonus: plot loss & accuracy

Cheatsheet

Train image 2000

Validation images 800

Input size 150x150x3 (w,h,RGB)

Conv_1: filters 32, kernel size(3,3)

Conv_2: filters 32, kernel size(3,3)

Conv_3: filters 64, kernel size(3,3)

Dense_1: 64, relu

Dense_2: ?, sigmoid

Image classifier with small set

Extract features from pre-trained model

- from keras.applications import vgg16
- Define image generator OR loop through images in data directory
- Use model.predict_generator to get features
- Save features in .npy file

Image classifier with small set

Train small MLP on bottleneck features

- Define network with two dense layers (don't forget activations and dropout)
- Compile with binary_crossentropy loss and rmsprop
- Train with bottleneck features
- Bonus: plot loss & accuracy



Image classifier with small set

Relu vs. Sigmoid

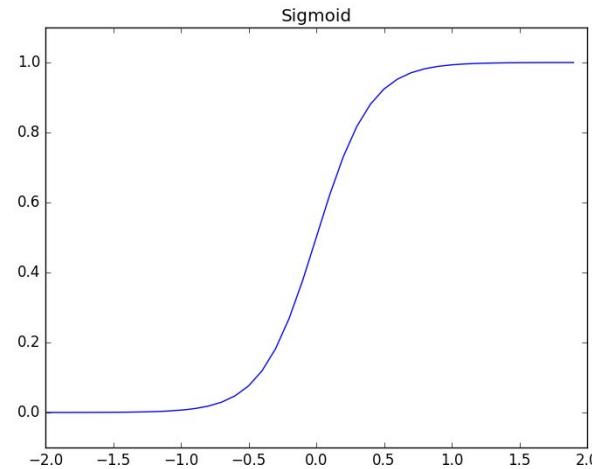
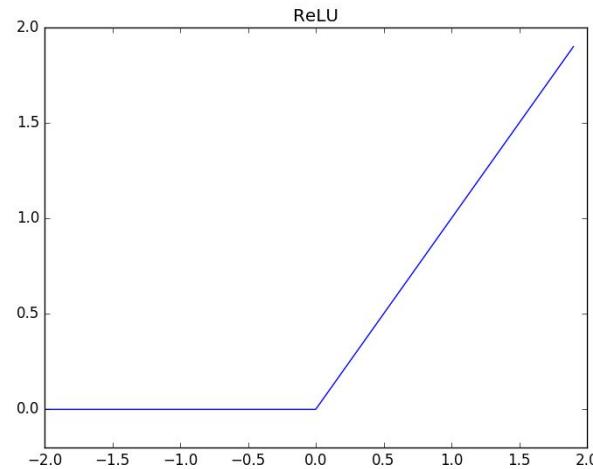


Image classifier with small set

Relu vs. Sigmoid

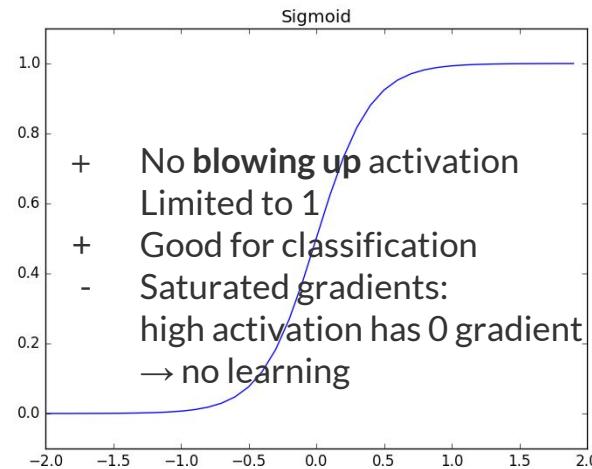
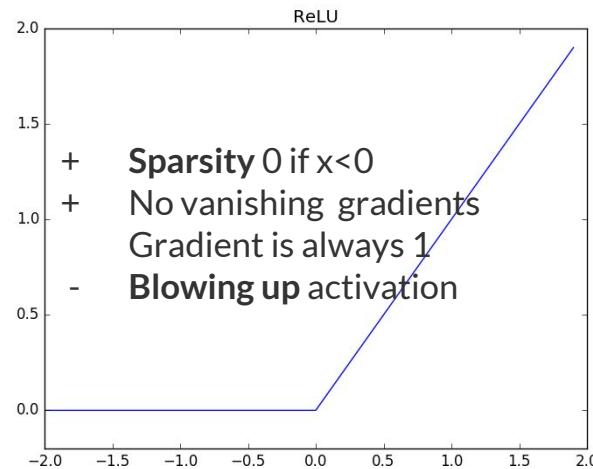




Image classifier with small set

Train small MLP on bottleneck features

- Define network with two dense layers (don't forget activations and dropout)
- Compile with binary_crossentropy loss and rmsprop
- Train with bottleneck features
- Bonus: plot loss & accuracy

Cheatsheet

Input size image features size

Flatten 3D feature maps

Dense_1: 256, relu

Dropout: rate 0.5

Dense_2: 1, sigmoid



Image classifier with small set

Fine-tune pre-trained network

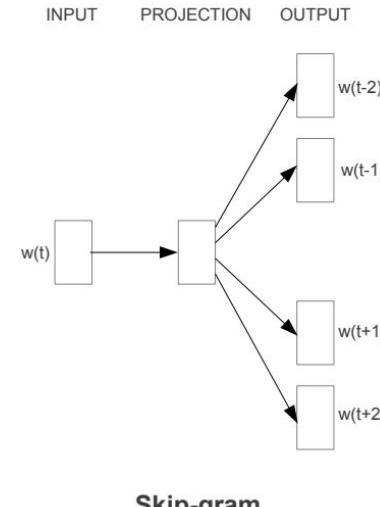
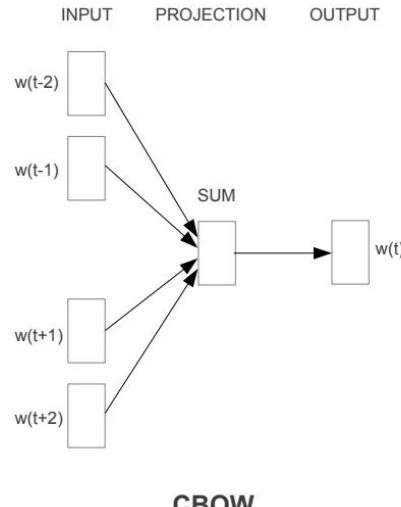
- Load pre-trained model vgg (weights same as before)
Note: specify input size according to our images
- Create a new model (vgg + previous mlp)
- Freeze first 15 layers of the new model
- Compile new model with binary_crossentropy loss and SGD with low learning rate (finetuning)
- Train with images



Word Embedding

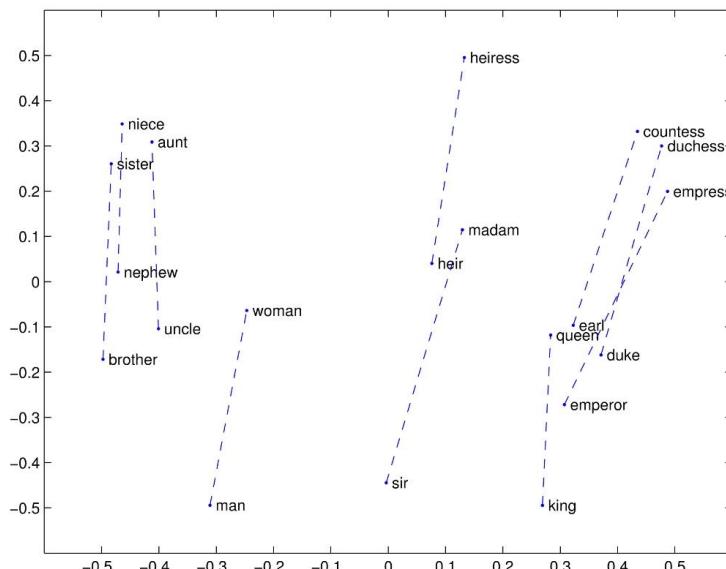
Word embedding

Dense representations: Word2vec



Word embedding

Dense representations: Global vectors for word representation (Glove)





Word embedding exercise

Word embedding exercise

Train your own word2vec

- pip install gensim, tsne, bokeh
- Use sample corpus cloned from repo OR download sample text corpus eng_news_2005_100K from <http://wortschatz.uni-leipzig.de/en/download/>
- Create new notebook train_word2vec
- Train word2vec model using gensim
- Sanity check
- Tsne & plot with bokeh

Word embedding exercise

Load pre-trained Glove

- Download pre-trained model from <https://nlp.stanford.edu/projects/glove/>
- Load the model using `gensim.models.KeyedVectors.load_word2vec_format`
Note: fix first line format
- **Trick:** for quicker loading
- Sanity check
- Tsne & plot with bokeh



Bonus



Extra resources

Some great resources if you want to dig deeper:

- [Blog post](#) about different optimization methods
- Matching networks for low-shot learning [code](#) implemented with tensorflow
- [Activation functions](#) comparison
- Stanford [course](#) on convolutional neural networks for visual recognition
- [Blog post](#) about using embedding layers in neural networks
- CVPR [tutorial](#) on zero-shot learning



Thanks!