



Learning with Small Samples

Including zero-shot learning

Nour Karessli
DSR 2018



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



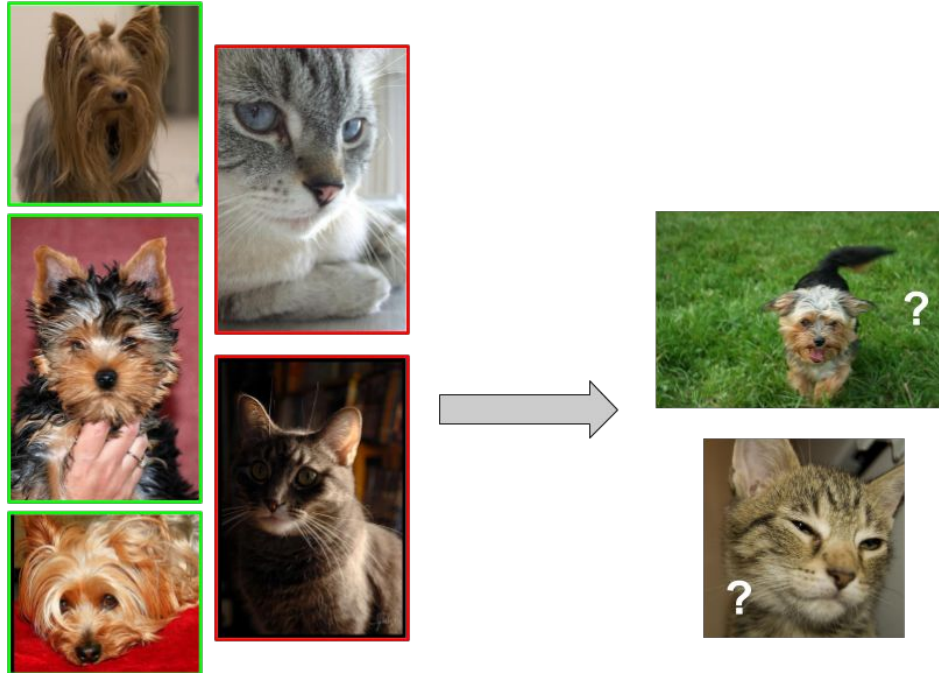
Tips & Tricks



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

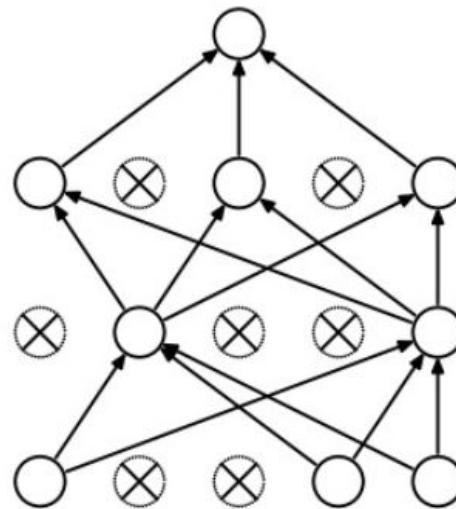
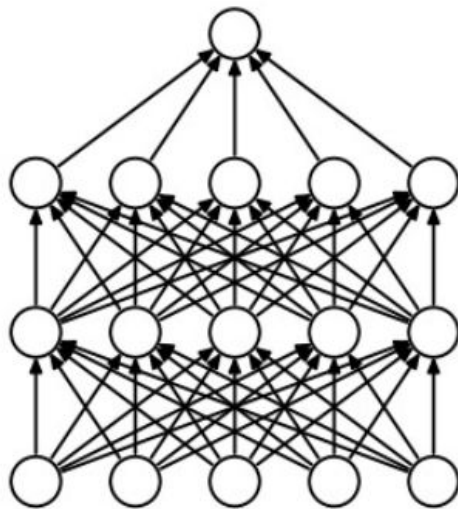
- Most common form of regularization
- Penalizing the squared magnitude of weights in the objective

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

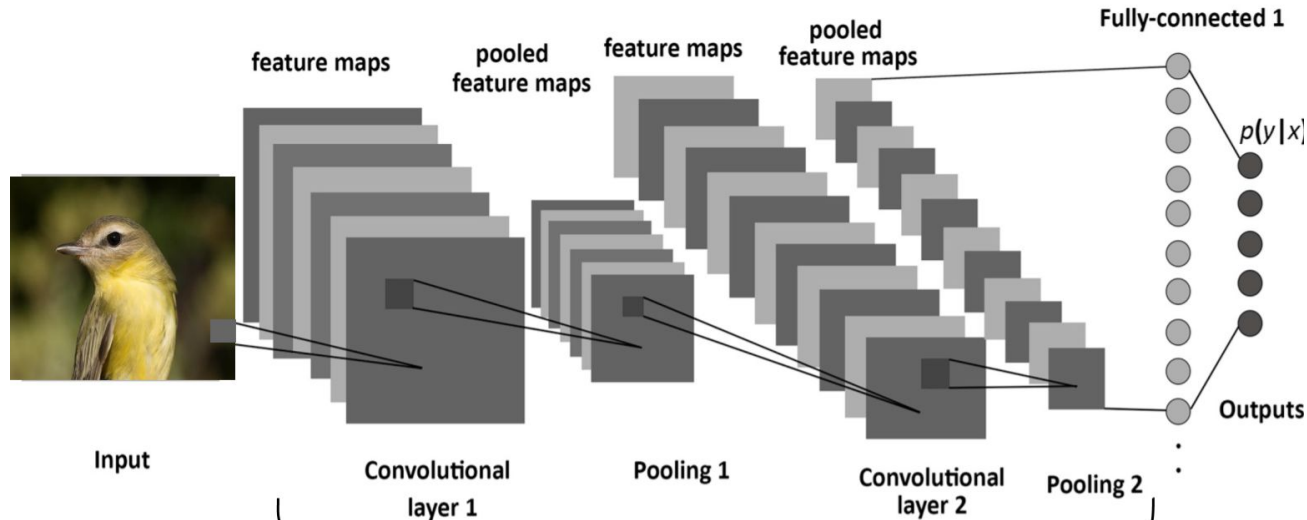
Problem Training a model from scratch only with small data is **challenging** and suffer from **overfitting**

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

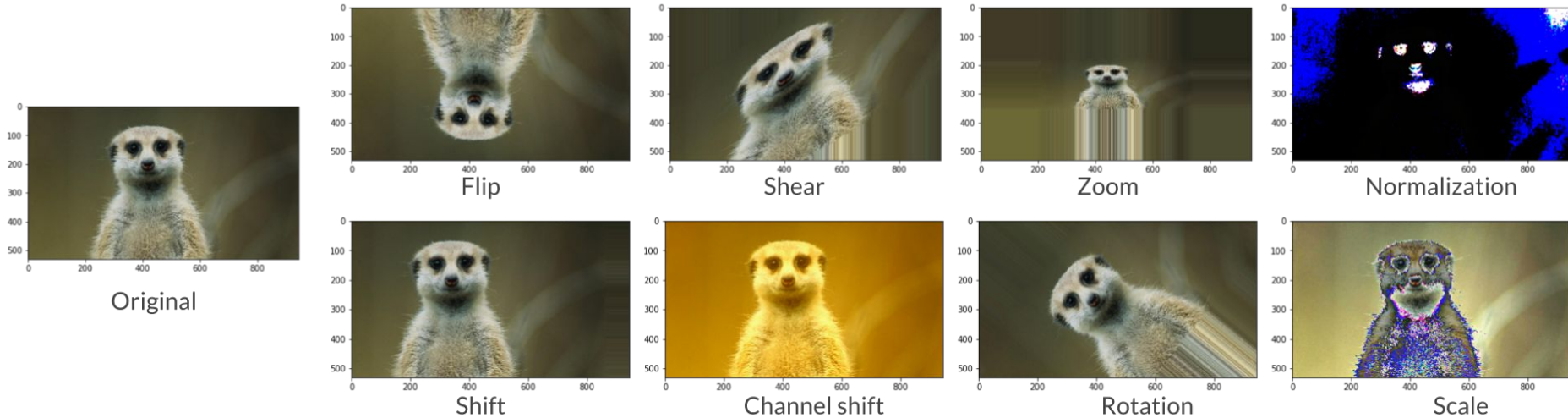


Data augmentation

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

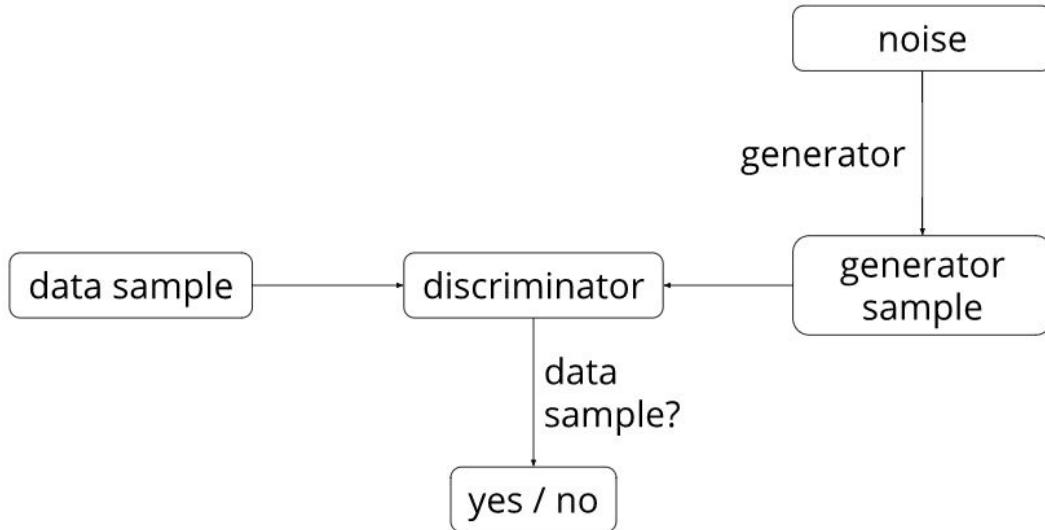
- Affine transformations
- Generative Adversarial Networks (**GANs**)

Affine transformations



Generative Adversarial Networks (GANs)

Learn data distribution



Generative Adversarial Networks (GANs)

Frontal face generator



GT profile

1

2

GT frontal

Generative Adversarial Networks (GANs)

Image to image translation



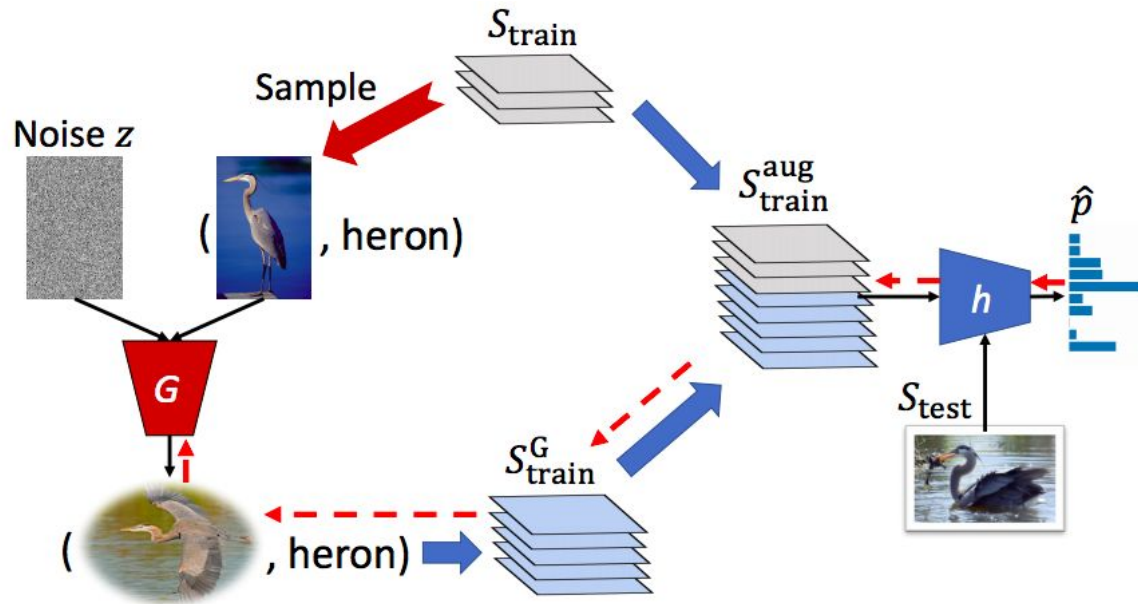
Low-shot learning + GANs

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

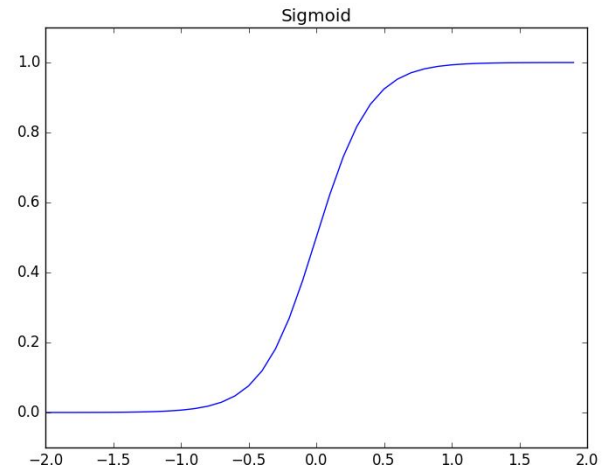
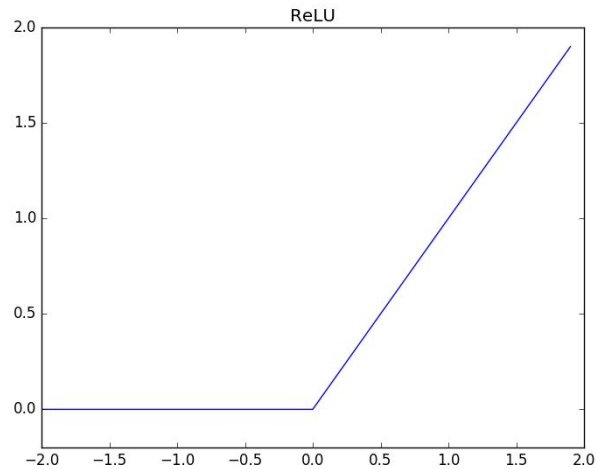


Image classifier with small set

Relu vs. Sigmoid

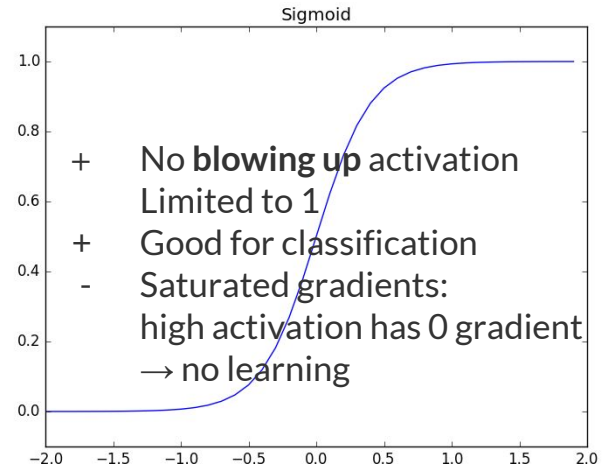
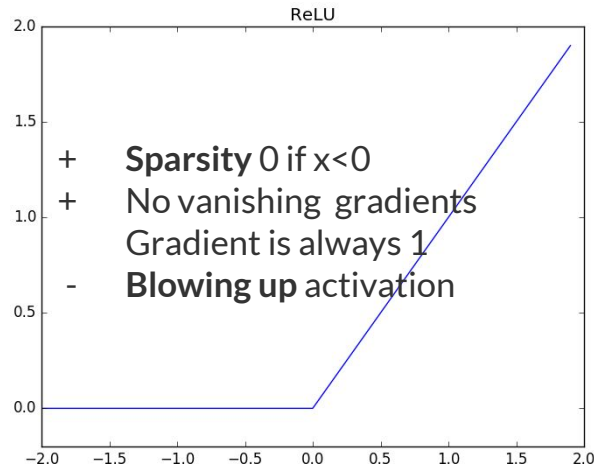




Image classifier with small set

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

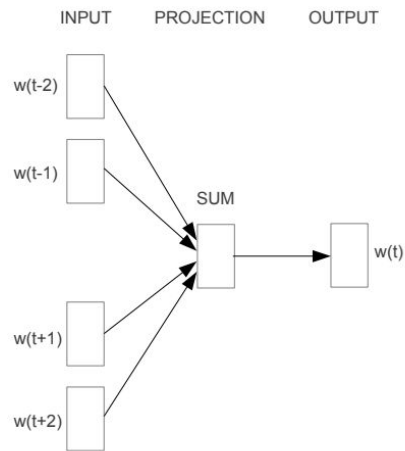
- Load pre-trained model vgg (weights same as before)
Note: specify input size according to our images
- Create a new model (vgg + previous mlp)
- Freeze first 15 layers of the new model
- Compile new model with binary_crossentropy loss and SGD with low learning rate (finetuning)
- Train with images



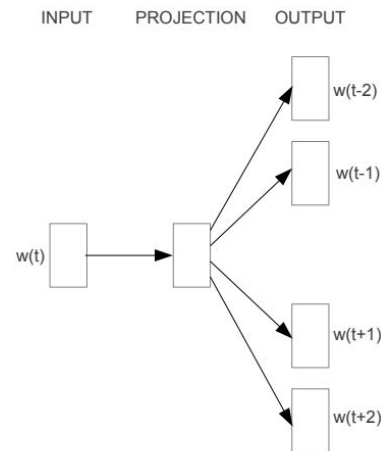
Word Embedding

Word embedding

Dense representations: Word2vec



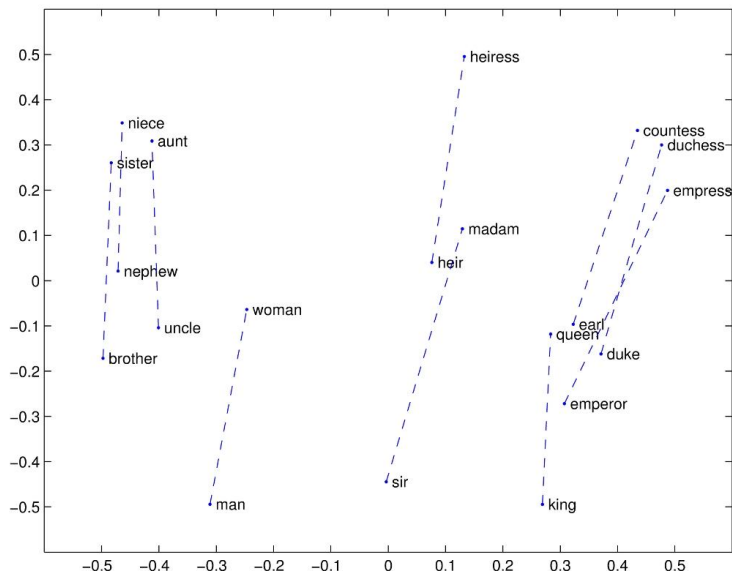
CBOW



Skip-gram

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 and [blog post](#)
- [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!