Deep Visual-Semantic Embeddings

Explanation, and implementation

Julian Villella
Innovation Tech. Lead

Outline

Research

1. What is a visual-semantic embedding?
2. The vision model
3. The language model
4. Projecting into word embedding space
5. Results

Implementation

6. Our (updated) vision pipeline
7. Deploying our new model
8. Large scale ANN search
9. Practical applications
10. Final comments + Q&A

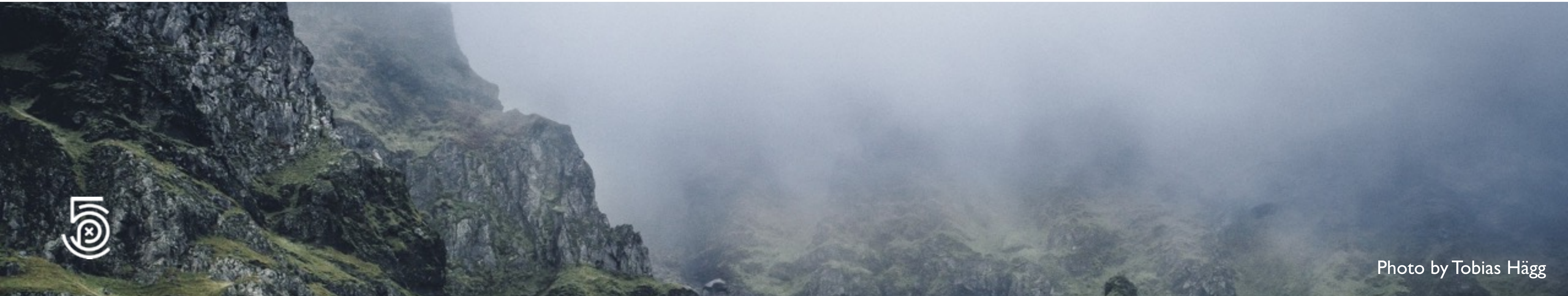


Photo by Tobias Hägg

What is a visual-semantic embedding?

Problem

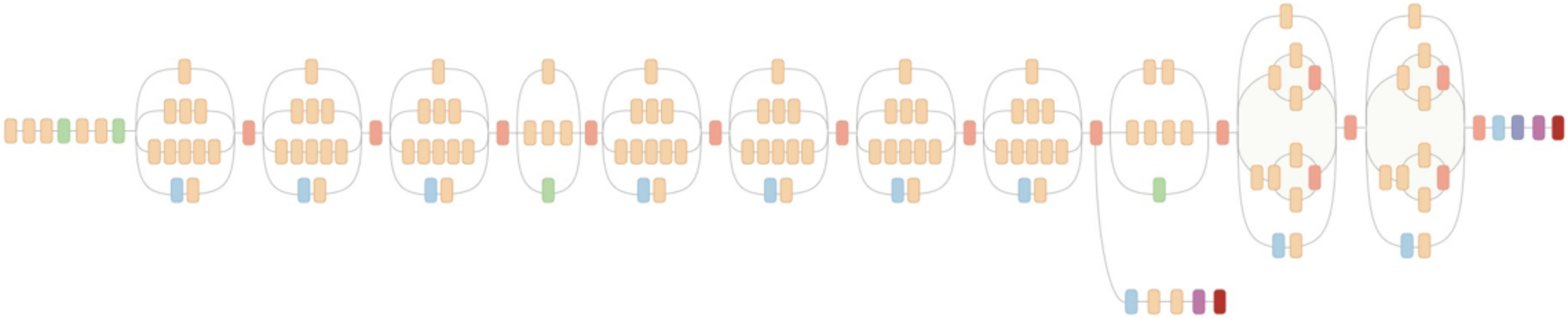
- Majority of image recognition systems are unable to scale to large number of labels
- Challenge to obtain training data as number of object classes grows
- Classifiers often treat classes as disconnect/ unrelated and unable to transfer learned labels to those unseen

Proposed solution (based on DeViSE paper)

- Train on both labeled data, and independent dataset of semantic information (btw. text)
- Proposed model learns how to project images into this semantic space
- Maintains state-of-the-art performance on N object categories
- Capable of “zero-shot” predictions

1/3 Vision Model

- Pre-trained vision model (i.e. ConvNet)
- We used Inception-v3
- Trained on ImageNet (1000 object classes)
- 5.6% top-5 error rate (comparable to human ability)



2/3 Language Model

- We used word2vec (skip-gram with negative sampling)
- Trained on unannotated text in unsupervised fashion
- Training,
  1. Learn weights of a single layer neural net
  2. Loop through each word in corpus (target), looking at N (window size) surrounding words (context)
  3. Maximize probability of returning context word over other random word in corpus (*negative sampling*)

the **quick** brown fox jumped over the lazy dog

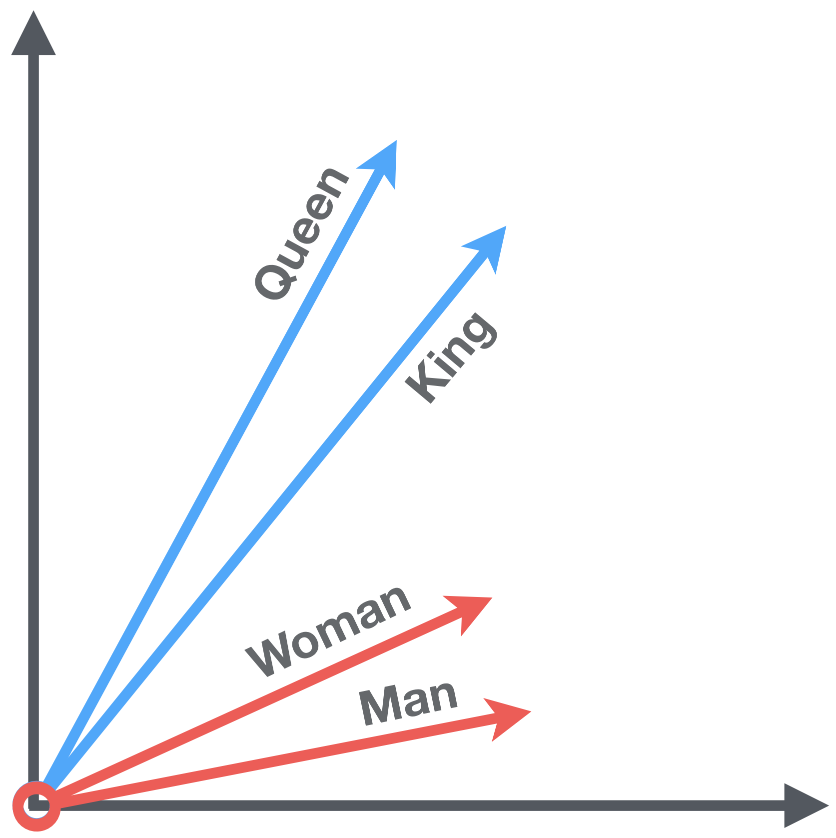
—

Given (context, **target**) and *window size of 1*,

we have ([the, brown], **quick**)

2/3 Language Model (cont.)

- Weights of hidden layer form our “word embeddings”
- Model learns semantics between words (demo)
    - I.e. words with similar meaning have similar embeddings (cosines)
- We can do arithmetic on these word embeddings
- We trained our own word2vec model on popular tags
    - Used in language-ai for translation disambiguation (for better localized search)
    - 65k words, 200-D



King + Man - Woman = Queen

3/3 Project image into WE space

- Take image representation from last classification layer of vision model
  - “image embeddings” (2048-D)
- Learn a projection from the image embedding into the word embedding space
- Projection layer is just a linear transform
- Use a loss function that trains the model to return higher cosine similarity between image embedding and word embeddings of its label (photo tag)
- At classification time,
  - Compute this projected embedding (“joint-embedding”)
  - Find nearest labels in its embedding space

Results

Benefits

- Scales to size of word embedding space (65k in our case)
- Zero-shot learning
- Learns semantic information about the image

Results

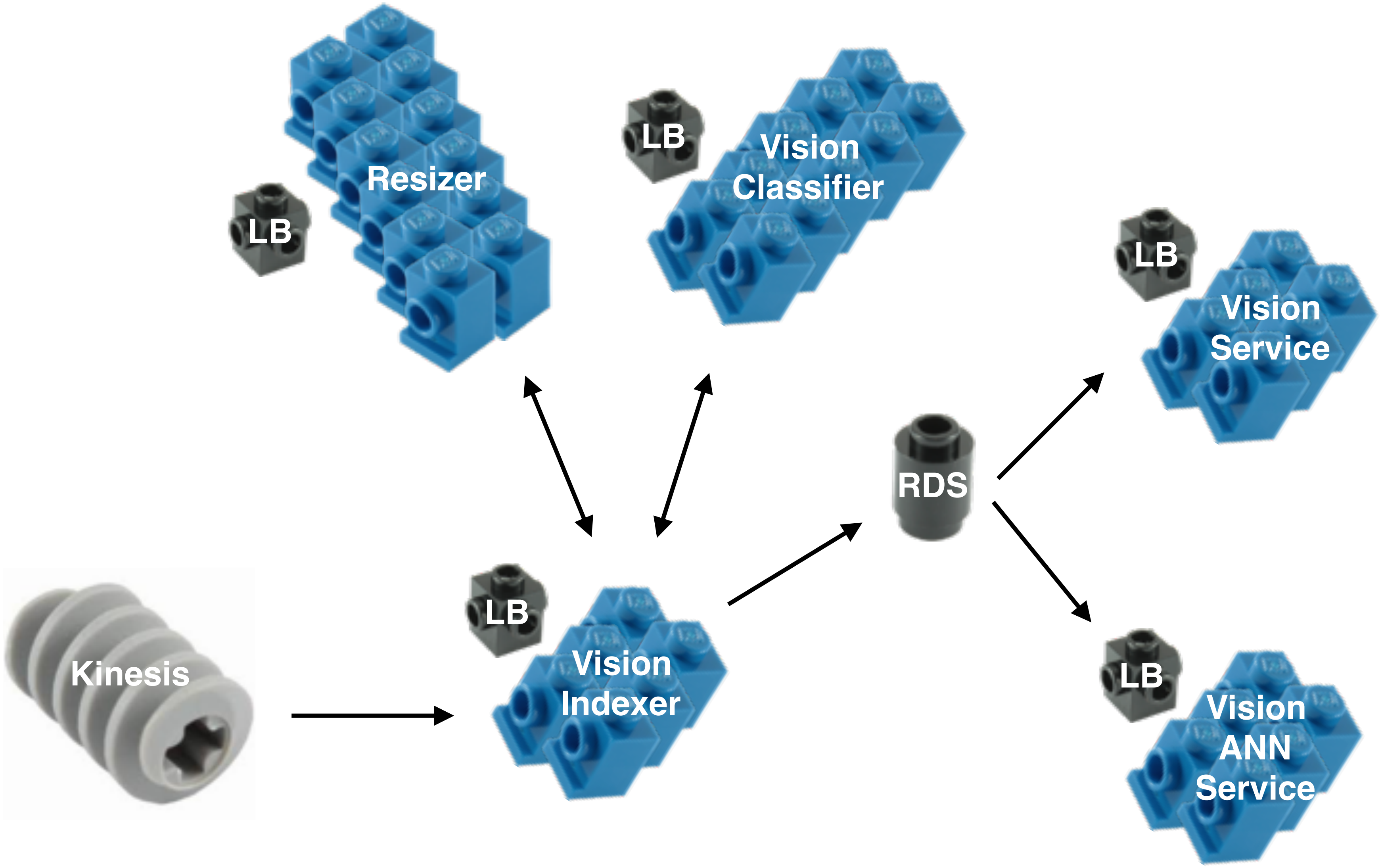
- TensorBoard visualization
- Zero-shot classification
- Gallery completion
- Reverse image search

Implementation

Putting our new model in production

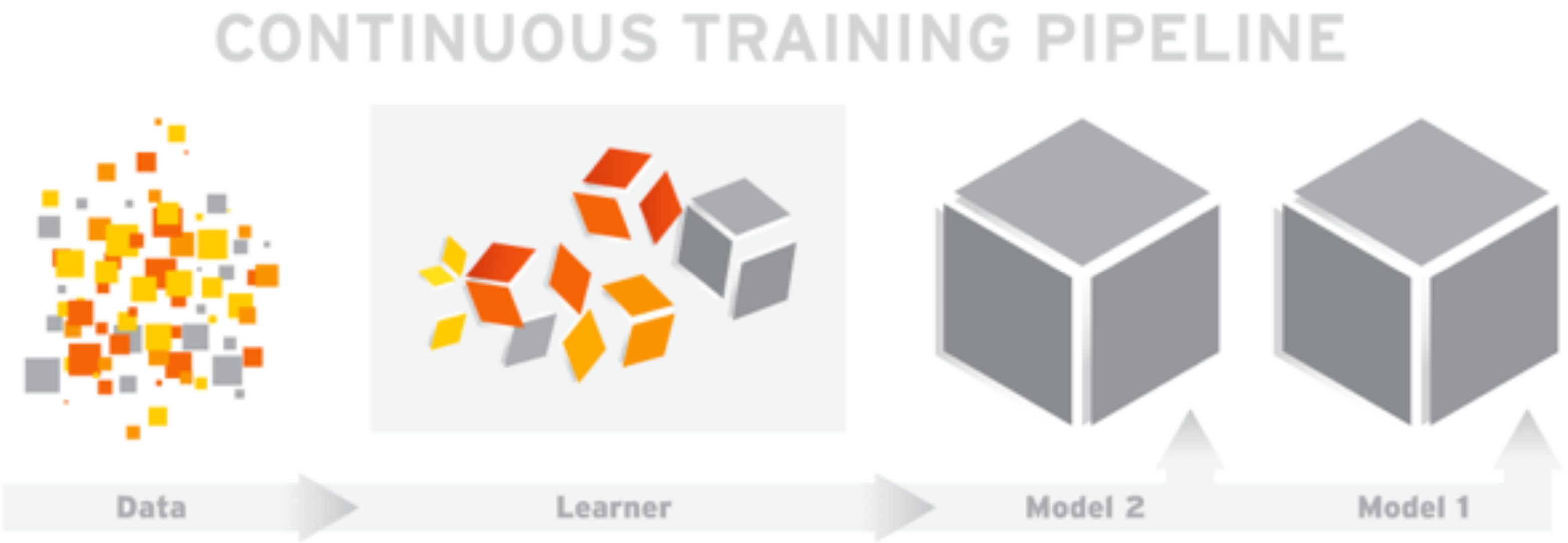
Our vision pipeline

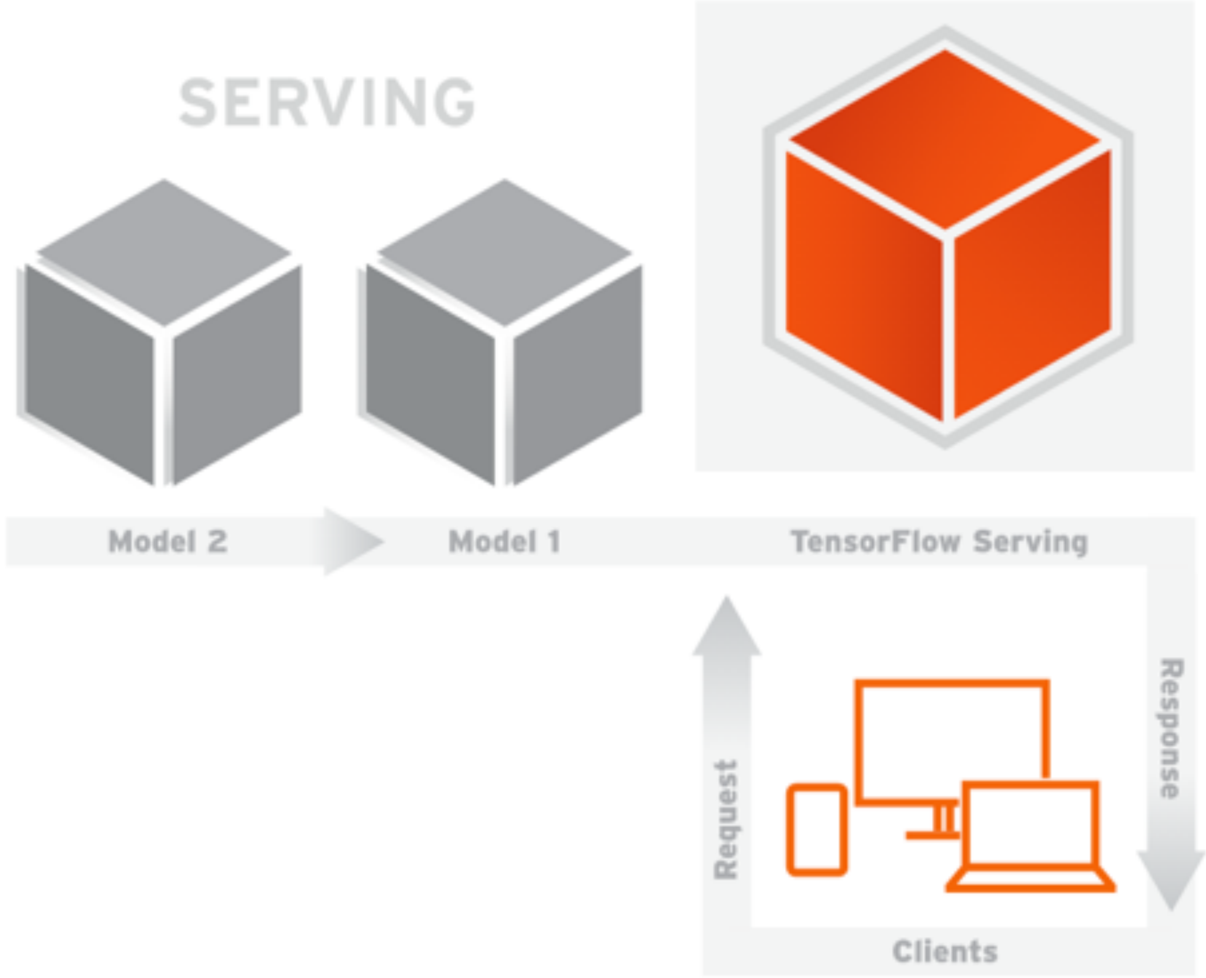
- Kinesis activity stream
- Vision indexer consumes activities
- Vision service, ann service to query
- Vision classifier running
  - Joint-embeddings model
  - Google LeNet model
- RDS stores image embeddings
- Datadog graphs



Deploying our new model

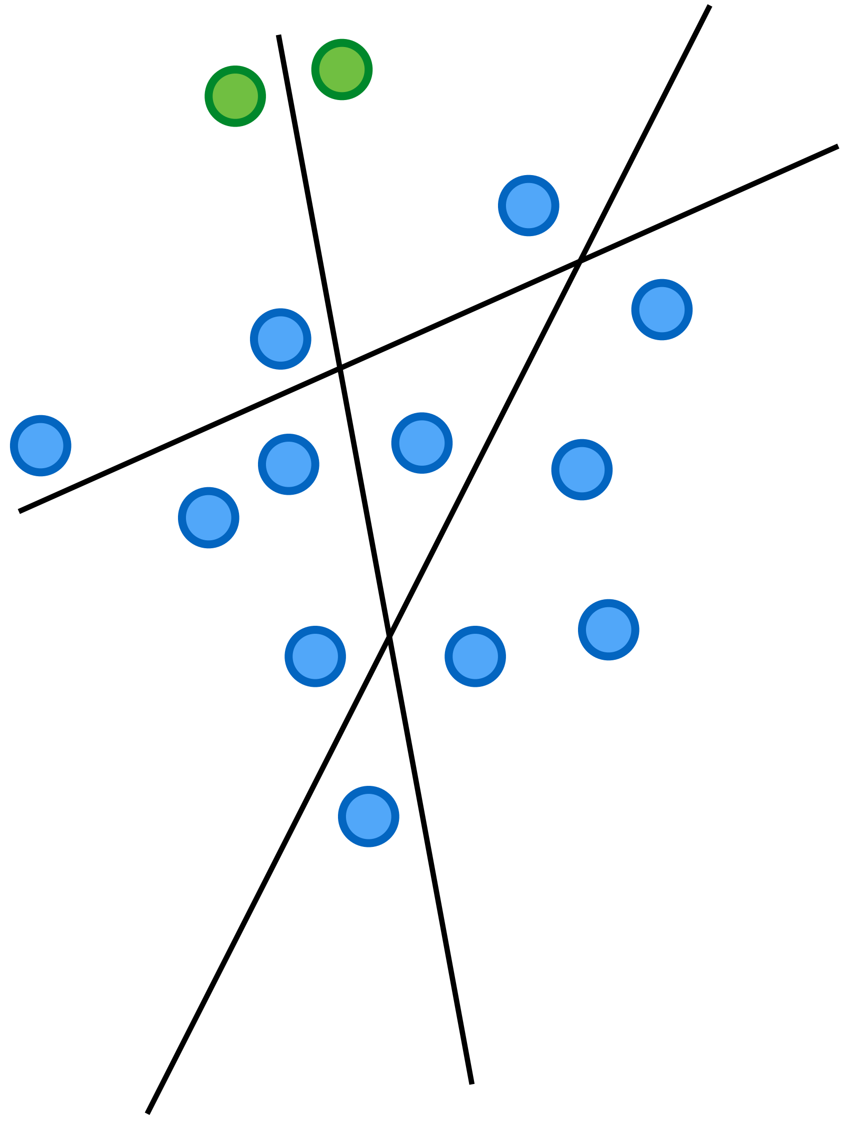
- Built in TensorFlow
- Uses TensorFlow Serving model server in production
- Running on a p2.xlarge EC2 instance (5000 CUDA cores)
    - Fits both Caffe and TensorFlow model
    - TensorFlow Serving loves memory :)
- Datadog graphs

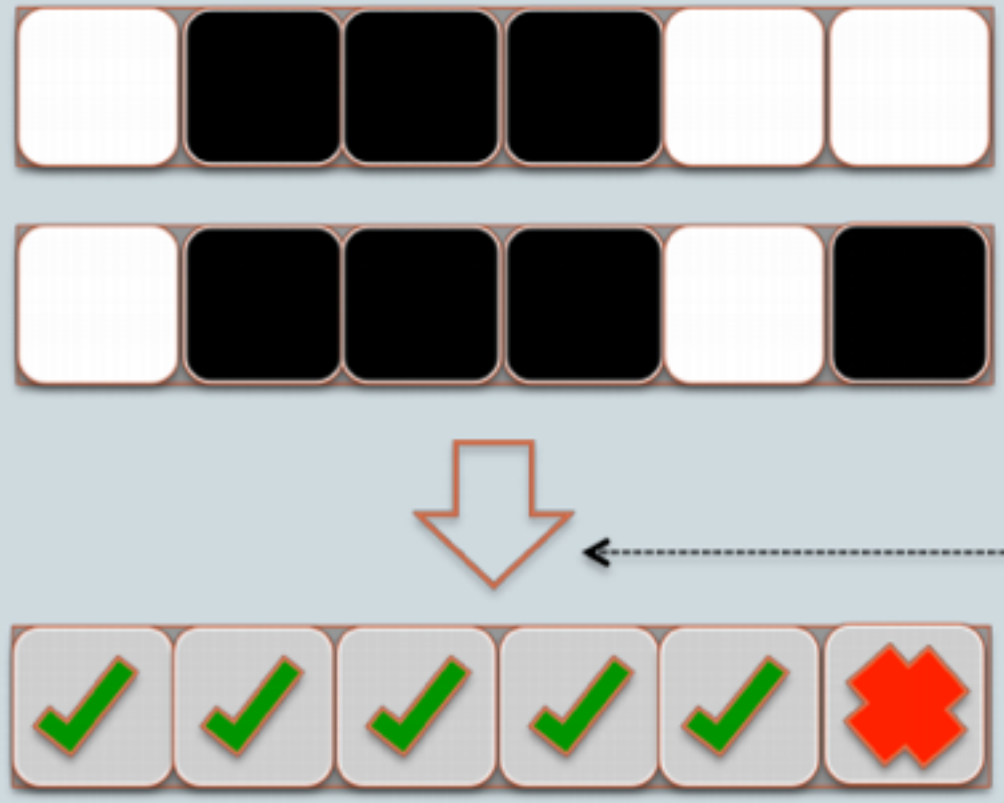




Large-scale approx. nearest neighbour search

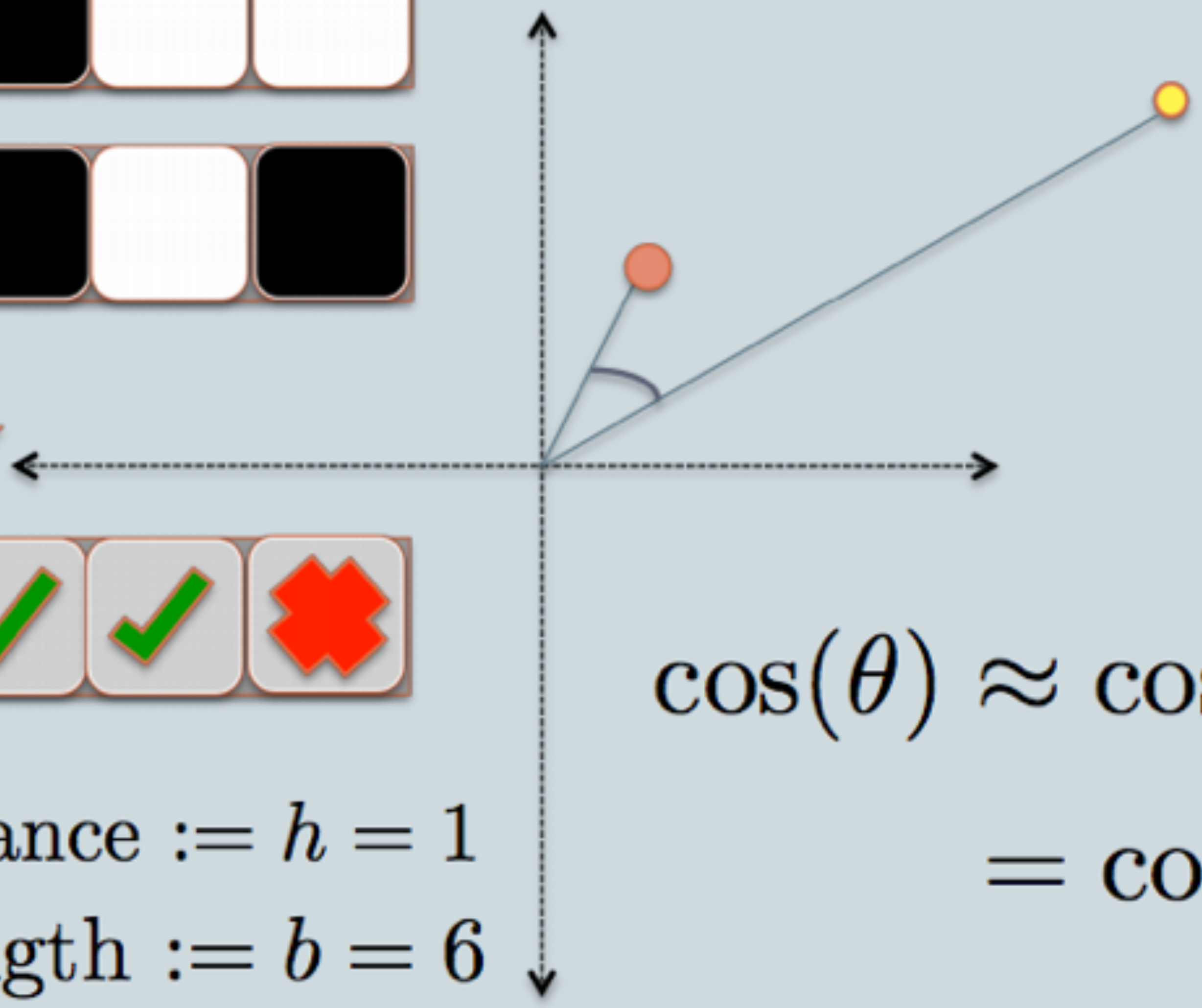
- *How do we find nearest neighbours? (sort by highest cosine similarity)*
  - We have 100M photos - this will not work
- Vision service uses locality sensitive hashing to compute a 24 bit string indicating the “bucket” each image resides
  - 2^{24} buckets (16M buckets)
  - Each bit says what side of a hyperplane we are on
- At query time, we only look sort the photos in each bucket
- Problems
  - Our dataset is non-uniform (i.e. lots of landscape photos, few still life)
  - What if we want more photos than there are in a given bucket?
  - What if the true NN is in an adjacent bucket?



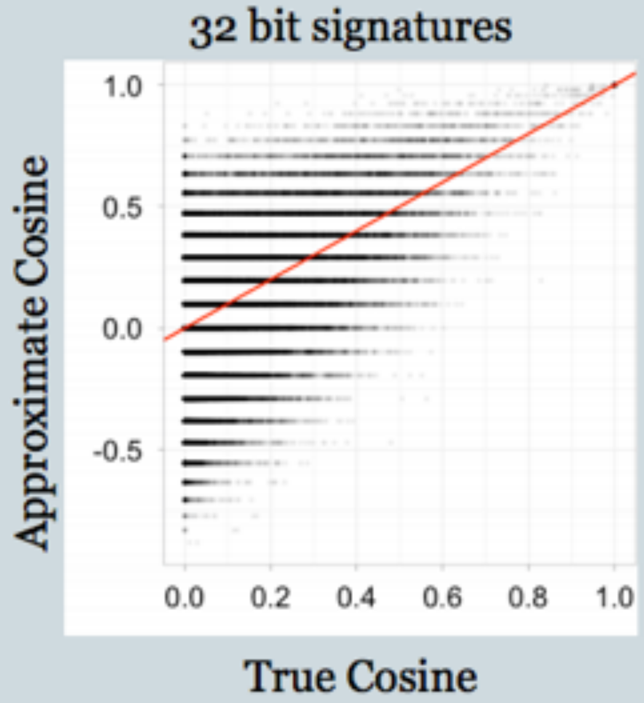


Hamming Distance $:= h = 1$

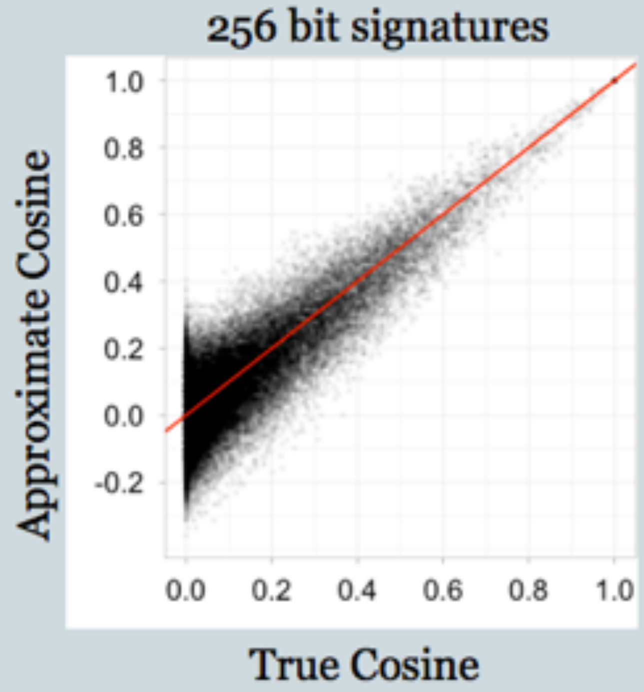
Signature Length $:= b = 6$



$$\cos(\theta) \approx \cos\left(\frac{h}{b}\pi\right) = \cos\left(\frac{1}{6}\pi\right)$$



Cheap

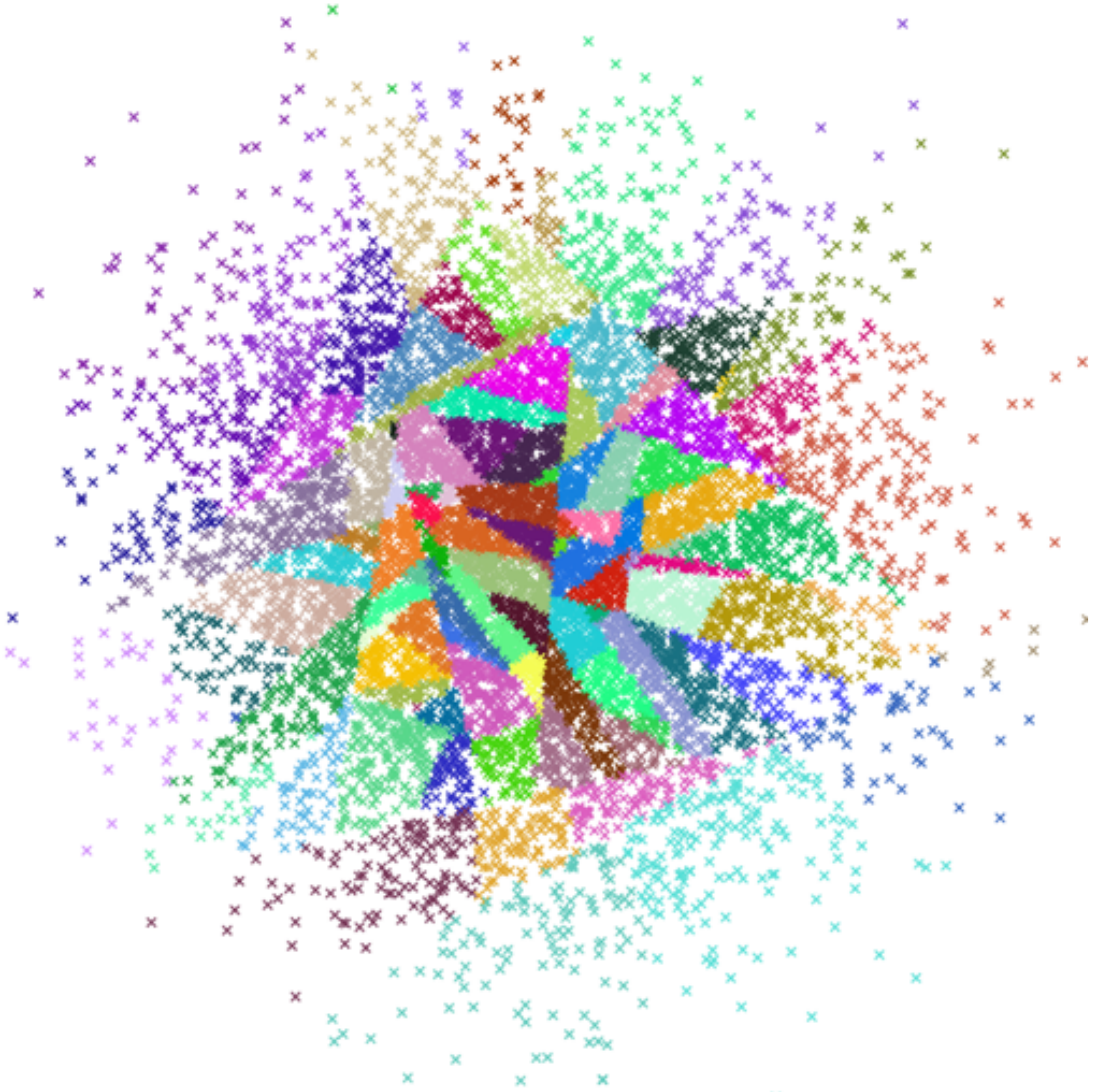


Accurate

Illustration from Benjamin Van Durme & Ashwin Lall

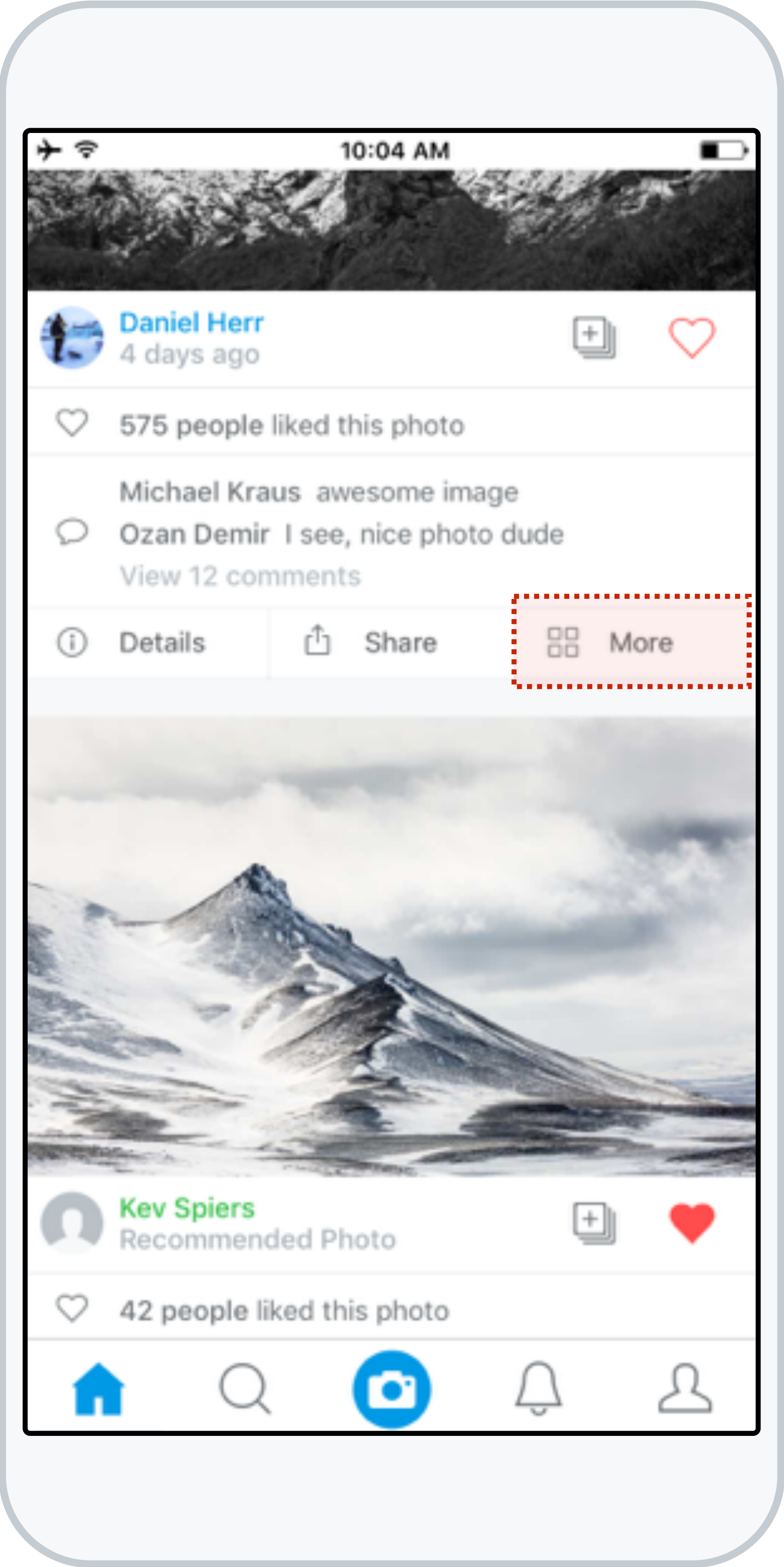
Large-scale ANN search (cont.)

- What if the true NN is in an adjacent bucket?
  - Forest of locality sensitive hashers
  - Each photo gets multiple hashes, and we join all photos in each bucket, NN search in there
- Vision ANN service is built on top of Spotify Annoy
  - Forest of binary search trees
  - Builds priority queue
  - Very fast (few ms)



Practical applications

1. Classification with much larger vocabulary (65k in our case)
2. Reverse image search (user uploads image)
3. Related image search (see right screenshot)
4. Search (find image neighbours to word query)
5. Gallery suggestions (mean of embeddings)
6. Vetting and/or ordering quest submissions
7. What about finding photographers for assignments given the work the upload? (similar to gallery completion task)



Resources

- **Projects**
  - Inception-v3
  - language-ai (our language model)
    - notebooks
  - vision-ai (our joint-embeddings model)
    - notebooks
    - locality sensitive hashing notebooks
  - Vision ANN Service
  - Spotify Annoy
  - TensorFlow, TensorFlow Serving
- **Demos**
  - “Zero-shot” predictions
  - Gallery completion
  - Reverse image search
- **Articles**
  - Word2Vec Tutorial - The Skip-Gram Model
  - DeViSE paper
  - How Zendesk Serves TensorFlow Models in Production

Questions?

Photo by Dina Belenko