Doodle-Image Matching For Bipartite Graph Analysis with Google's Quick Draw!

Heejung Chung, Alex HyunJi Nam

MOTIVATION (TWO PARTS)

1) Guiding Q: *Humans use doodles to simplify real life images. Can a model identify similarities between image and doodle of the same object without being explicitly trained on their category?*
2) Guiding Q: *How to leverage bipartite structure to group doodles and images more accurately?* Exploring bipartite graph clustering to mitigate the effect of classifier error/noise in edge weights

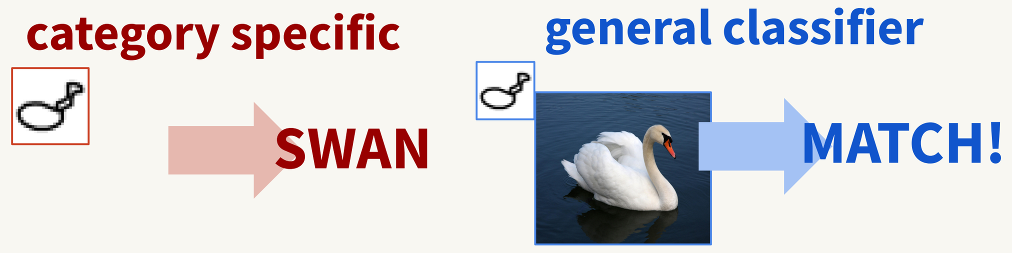
MATCHING

PROBLEM & CHALLENGES

Problem

- Unlike previous work with this dataset which classified individual doodles, we want to determine whether a given doodle and image pair are in the same category (e.g. swan), ie if they "match"
- Goal: To build an easily generalizable matcher that can transfer learning across diff. categories of images & doodles

previous work: category specific | **our work: general classifier**



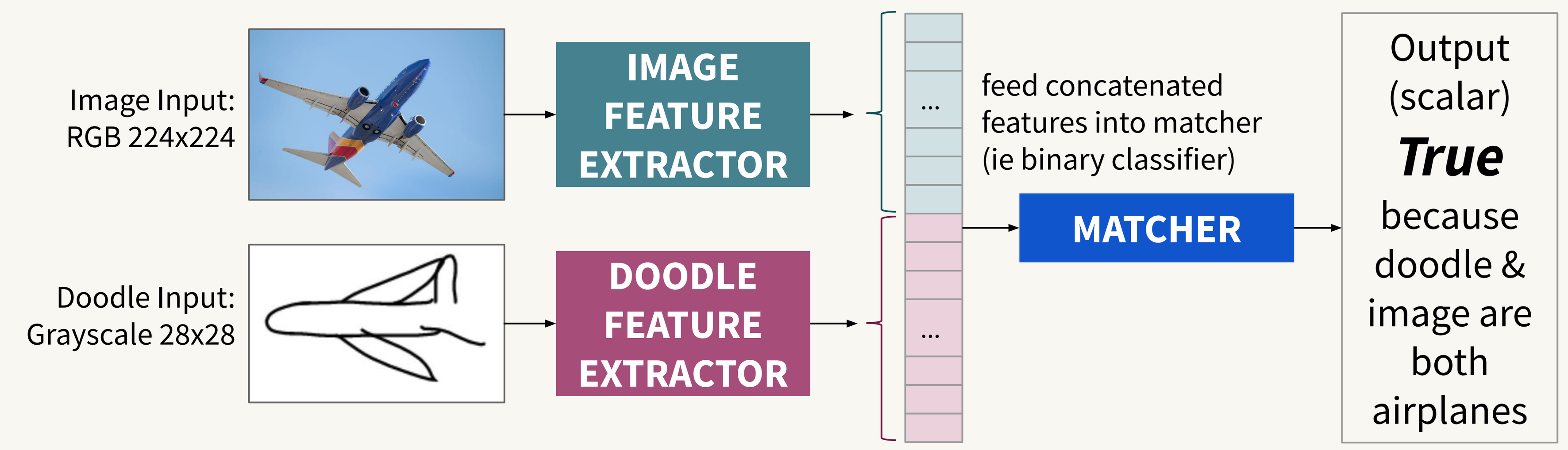
Data infrastructure

- *Doodle*: 28x28 grayscale collected from kaggle
- *Image*: scraped from google with keywords
- *Baseline*: 3 object categories with 1200 balanced examples for training, 300 for val.
- *Better Models:* 236 object categories with 19,000 balanced +/- examples for training, 4000 for val.

Challenges

- doodle and img. dimensionality differences
- noise in doodles (poor quality drawings do not resemble images/real objects) → may be unsolvable even with complex model

APPROACHES



tried different combinations of components...

IMAGE EXTRACTOR

raw pixels scaled to 14x14

raw pixels scaled to 28x28

scaled to 3x224x224 → VGG16 intermediate layer → 512 features

DOODLE EXTRACTOR

raw pixels scaled to 14x14

raw pixels scaled to 28x28

padded & copied over 3 channels to 3x128x128 → VGG16 intermediate layer → 512 features

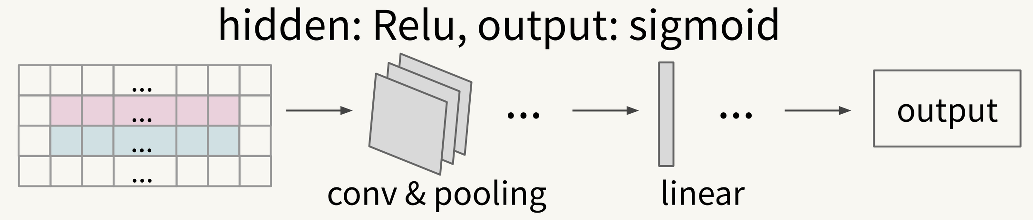
MATCHER

logistic, L1 regularization

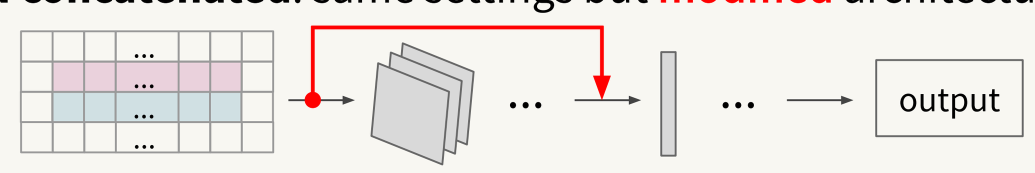
SVM, gaussian kernel

linear NN: VGG16 input, max epoch 20, batch size 10, dropout .1, MSE loss

CNN: same settings ⤴ + Adam optimizer, kernel size 2, hidden: Relu, output: sigmoid



CNN concatenated: same settings but modified architecture



RESULTS

Input Features: IMAGE	Input Features: DOODLE	MATCHER	Val Accuracy, varying C[1]: 0.1	1	10	100
Raw 14x14	Raw 14x14	Logistic	52.6	48.0	48.3	48.3
Raw 28x28	Raw 28x28	Logistic	51.3	51.3	51.0	51.0
Raw 28x28	Raw 28x28	SVM	50.7	**82.0**	**83.0**	**83.0**
Raw 28x28	Raw 28x28	SVM	52.9	58.2	60.6	N/A[2]
VGG16 → 512	VGG16 → 512	SVM	59.7	73.4	**77.5**	**77.1**

1) Logistic: higher C ⇒ more regularization, SVM: higher C ⇒ less regularization
2) Did not converge

NN Architecture: depth[2]	activation[3]	optimizer	Train Loss	Val. Loss	Val. Accuracy[4]	Epoch[5]
3	Relu	Adam	0.07	0.190	0.761	16
3	Relu	SGD	0.054	0.189	0.752	19
3	Hardtanh	SGD	0.085	0.195	0.737	20
3	Sigmoid	SGD	**0.25**	**0.250**	**0.505**	**1**
4	Relu	Adam	0.056	0.196	0.765	16
4	Relu	Adam	**0.055**	**0.192**	**0.767**	**18**
4	Relu	SGD	0.081	0.188	0.754	18
5	Relu	Adam	0.061	0.208	0.767	17

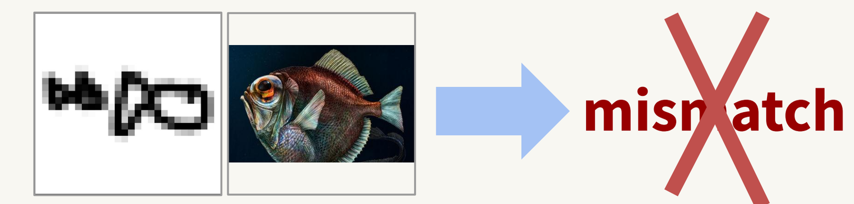
1) Includes input & output layers. Hidden layer dims may differ
2) Used sigmoid for all final activation
3) Best prediction accuracy from chosen epoch
4) Chose epoch with best prediction accuracy

CNN Architecture: conv depth[1]	linear depth	concat.	Train Loss	Val. Loss	Val. Accuracy[2]	Epoch
1	4	N	**0.251**	**0.250**	**0.505**	**1[3]**
1	3	N	0.038	0.233	0.739	19
2	2	N	**0.502**	**0.504**	**0.496**	**1[3]**
2	2	N	0.224	0.249	0.556	6
1	3	Y	**0.036**	**0.204**	**0.771**	**18**

1) Hidden layer sizes may differ
2) Best prediction accuracy chosen from 20 training epochs
3) Accuracy and losses did not improve over epochs

ANALYSIS

Qualitative Analysis (example)



doodle of two fish v.s. image of one fish (numbers may be misleading)

Overfitting with NN & CNN

- Train loss much smaller than val. loss, but val. accuracy decreases with more epochs

Poor generalization w/ ⇧ classes

- Prediction accuracy much higher when the model is trained on fewer classes of easily distinguishable objects

Ambiguity w/ generalization eval.

- Given a positively labeled doodle-image pair of an airplane, is the model detecting that they are both airplanes? Or is it detecting the more general similarity between the doodle and the image regardless of the category?

Confusion matrix from SVM

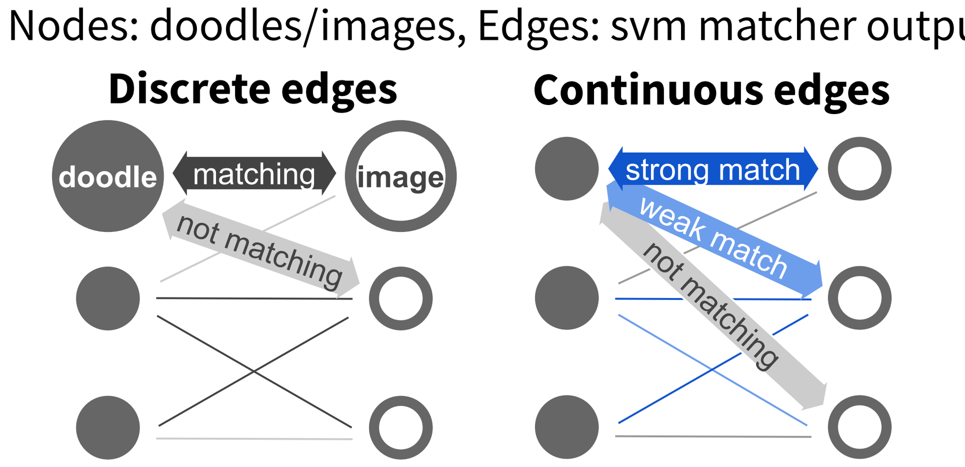
(F1 score = 0.778)

	Pred. mismatch	Pred. match
Real mismatch	**3194**	1098
Real match	848	**3395**

GRAPH ANALYSIS

Problem

Nodes: doodles/images, Edges: svm matcher output



** Will omit non-matching edges in subsequent diagrams*

Goal: find clusters, given different graph scenarios

- **1-to-1**, 1 doodle & 1 image from each category
- **singleton**, 1 node doesn't match any others
- **large clusters**, *n* doodles & *n* images from each category

Challenges

- Noisy edges (matcher accuracy < 80%)
- Bipartiteness ⇒ need special algorithms
- Tradeoff: efficiency (binary edges) vs. accuracy (continuous edges)

Pivot Bi Cluster (PBC) *introd. by Ailon and Avigdor-Elgrabli 2010*

1. Randomly choose a doodle as the ***pivot*** & adds adjacent images† to the cluster
2. Check if other doodles also connected to clustered images to determine probability* of (a) adding to cluster, (b) creating singleton & remove from graph, (c) leaving for future iterations
   - **Pros:** Versatile (no fixed cluster size; able to generate singletons)
   - **Cons:** Requires discrete edges, high variance due to randomly selected pivots

Original Version (Discrete: D-PBC)

† Deterministically add adj. images

* *Prob*(including in cluster) is based on # of (un)shared neighbors

Our Improved Version (Continuous: C-PBC)

† *Prob*(add adj. image) = edge weight*scaling factor (ie probabilistically filters images)

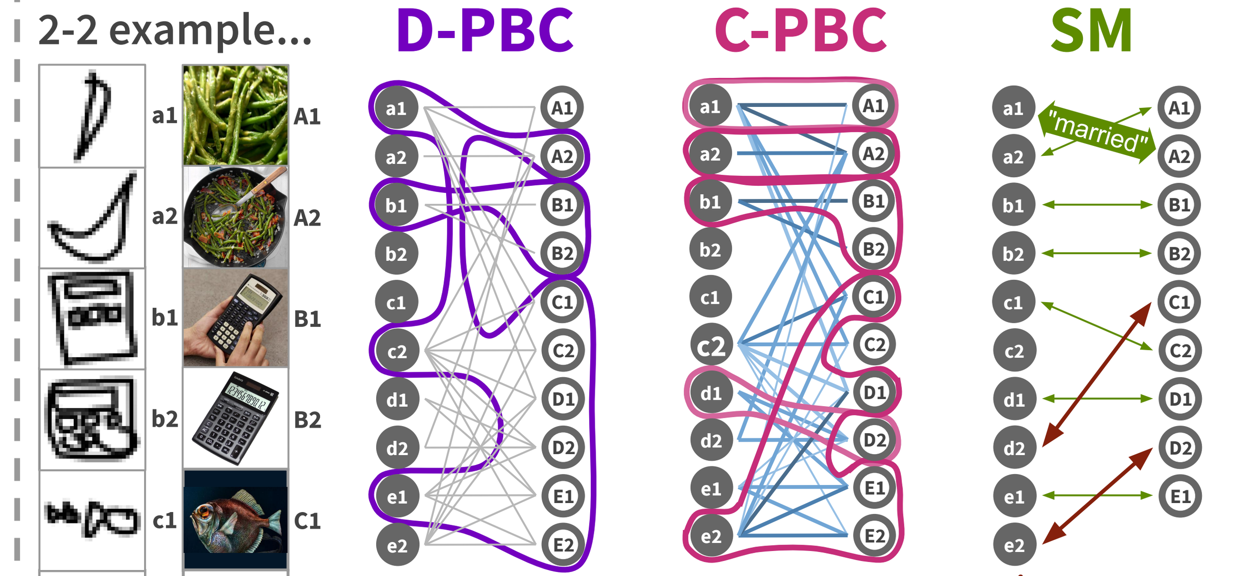
* *Prob*(including in cluster) is based on sum of corresponding edge weights

Stable Marriage Problem (SM) *Gale-Shapley 1962*

Everyone is initially unmatched → while some man *m* is unmatched, match *m* with *m*'s most preferred woman *w* if *w* is also unmatched or if *w* prefers *m* to its current match; otherwise, consider *m*'s next preferred woman → continue breaking and matching pairs until exhausted

- **Pros:** May be able to combat classifier noise
- **Cons:** Assumes 1-1 matching

2-2 example...



* non-clustered are singletons

* two ***incorrect*** marriages

Model	Other scenarios: singleton	1-1	many n-n
D-PBC	Sometimes misses singleton	1 misclassification	Large clusters w/ misclassification, many singletons
C-PBC	Consistently finds singleton	x misclassification some singletons	More accurate assignment than above but more singletons
SM	N/A	0~1 mismatched pair	0~1 mismatched pair

Improvements: D-PBC to C-PBC

- scaling factor: singletons vs large clusters
- weak connections may be ignored (denoising effect on classifier error)

Stable Marriage as a denoiser

- Solving marriage problem actually **corrects** lots of errors (often, matcher-based edge weights are not accurate)

General Limits with PBC

- high variance, too many singletons

NEXT STEPS

- How to quantitatively evaluate successful multiclass n-n clustering (ie how to score)
- How to combat variance of random alg. -- run many times and choose clustering with highest score?