Quick, Draw! Doodle Recognition

Kristine Guo (kguo98@stanford.edu), James WoMa (jaywoma@stanford.edu), Eric Xu (ericxu0@stanford.edu)

Motivation

Quick, Draw!
- Players draw a picture of a given object
- Computer attempts to guess object category

Our Project:
- Classify 28x28 hand-drawn doodles into 345 categories
- Goal: Compare performance of KNN with CNN and discover underlying features of doodles

Data

Google publicly released a Quick, Draw! Dataset
- Over 50 million images across 345 categories
- Each drawing is a 28x28 grayscale matrix
- Provided ground truth labels

Cross Validation
- Randomly selected 1% of dataset
- Split that into train/val/test folds with 70/15/15 distribution
- Dataset sizes:
  - Training: 352,955 examples
  - Validation: 75,655 examples
  - Test: 75,832 examples

Evaluation

Mean Average Precision @ 3 (MAP@3)

- $$MAP@3 = \frac{1}{U}\sum_{u=1}^{U}\sum_{k=1}^{min(n,3)} P(k)$$
- U: # scored drawings in the test data
- P(k): the precision at cutoff k
- n: # predictions per drawing

Mean Average Precision @ 1 (MAP@1)
- Measures single-prediction accuracy

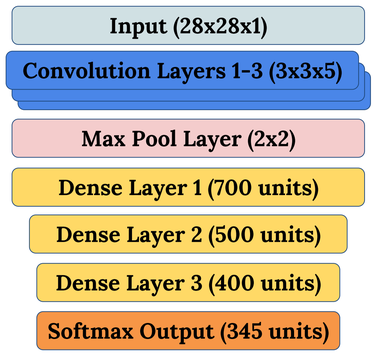
Models

Baseline: 1-Nearest Neighbor
- Calculate each category's centroid by averaging together training examples
- Classify test example with closest centroid

Extension 1: KNN with Multiple Clusters
- Goal: find distinct category representations
- Calculate 5 centroids per category using k-means (with k-means++ initialization)
- Take the top k closest centroids to use as votes for the example's classification

Extension 2: KNN with Weighted Votes
- Weight centroids that are further away from the examples less
- Distance weighting: $w_i = 1/dist[x_i, c]$
- Ranking weighting: $w_i = 1/sqrt(i)$

Convolutional Neural Network

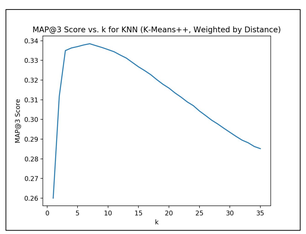


- Dense layers use ReLu activation function
- Dropout with rate 0.2 after each dense layer
- Train over 20 epochs with 1e-3 learning rate and batch size of 32

Results

	1-NN	KNN++, *k*=3	KNN++ (distance), *k*=7	KNN++ (rank), *k*=29	CNN
MAP@1 (Train)	18.4%	29.2%	28.3%	29.3%	52.9%
MAP@3 (Train)	24.2%	36.7%	36.3%	37.1%	61.8%
MAP@1 (Dev)	18.2%	17.3%	26.3%	26.8%	53.5%
MAP@3 (Dev)	23.9%	28.0%	33.8%	34.5%	62.2%
MAP@1 (Test)	17.9%	17.0%	26.2%	26.7%	53.4%
MAP@3 (Test)	23.6%	27.7%	33.7%	34.4%	62.1%

Table 1. MAP@1 and MAP@3 scores for all methods on all three datasets.



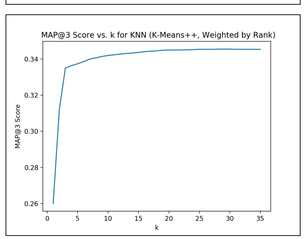
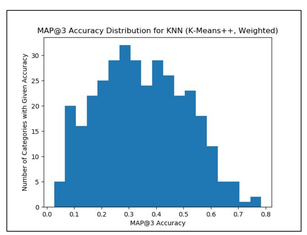


Figure 1. MAP@3 scores plotted against different values of k for KNN++ with weighted voting (rank, distance).



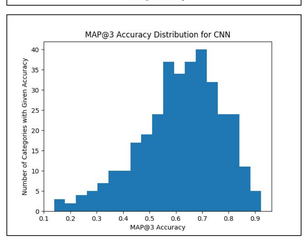
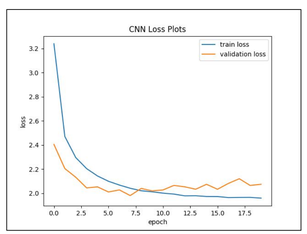


Figure 2. MAP@3 accuracy distributions on the test set for KNN++ (weighted by rank) and CNN.



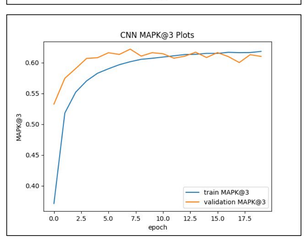


Figure 3. CNN loss (top) and MAP@3 scores (bottom) on training and validation set.

Discussion

- Models produced scores that were significantly higher than randomly guessing (~0.3% MAP@1 and ~0.5% MAP@3)
- Running k-means with k-means++ initialization successfully produced different representations of centroids for each category (Figure 4)
- Some categories produced nearly identical centroids (Figure 5), making it difficult to classify drawings by only comparing pixels with L2 distance in KNN

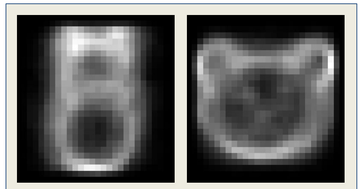


Figure 4. *Two bear centroids computed from k-means++*

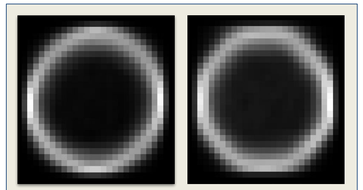


Figure 5. *Comparison of mean Circle and mean Octagon*

- KNN with weighted votes by rank produced the highest scores out of the KNN models and provided stable performance at high k values
- KNN was able to differentiate between general structures of doodles (i.e., it often guessed onion, apple, and blueberry together)
- KNN models were unable to learn local features such as the stem of onions or apples that distinguish them from blueberries (Figure 6)

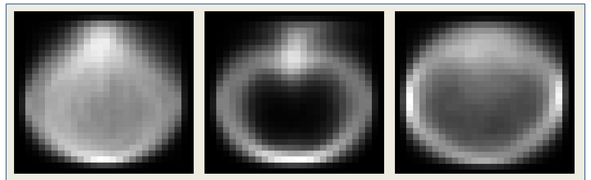


Figure 6. *Comparison of mean Onion, Apple, and Blueberry (in order)*

- CNN on the other hand utilizes the convolutional filters to learn these local features and outperformed baseline models by a large margin

Future Work

- Experiment with advanced CNN architectures (VGG-Net, ResNet)
- Train models on complete dataset along with stroke order information
  - e.g., velocity and acceleration
  - Stroke order allows for interesting RNN models
- Build ensembles to achieve even higher scores

References

[1] Ha, D., & Eck, D. (2017). A neural representation of sketch drawings. arXiv preprint arXiv:1704.03477.

[2] Lu, W., & Tran, E. (2017). Free-hand Sketch Recognition Classification.

[3] Kim, J., Kim, B. S., & Savarese, S. (2012). Comparing image classification methods: K-nearest-neighbor and support-vector-machines. Ann Arbor, 1001, 48109-2122.