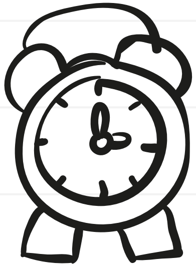
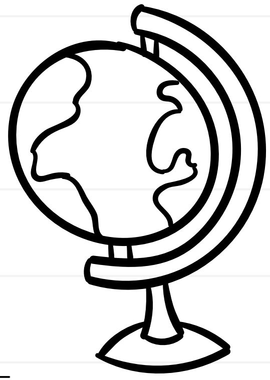


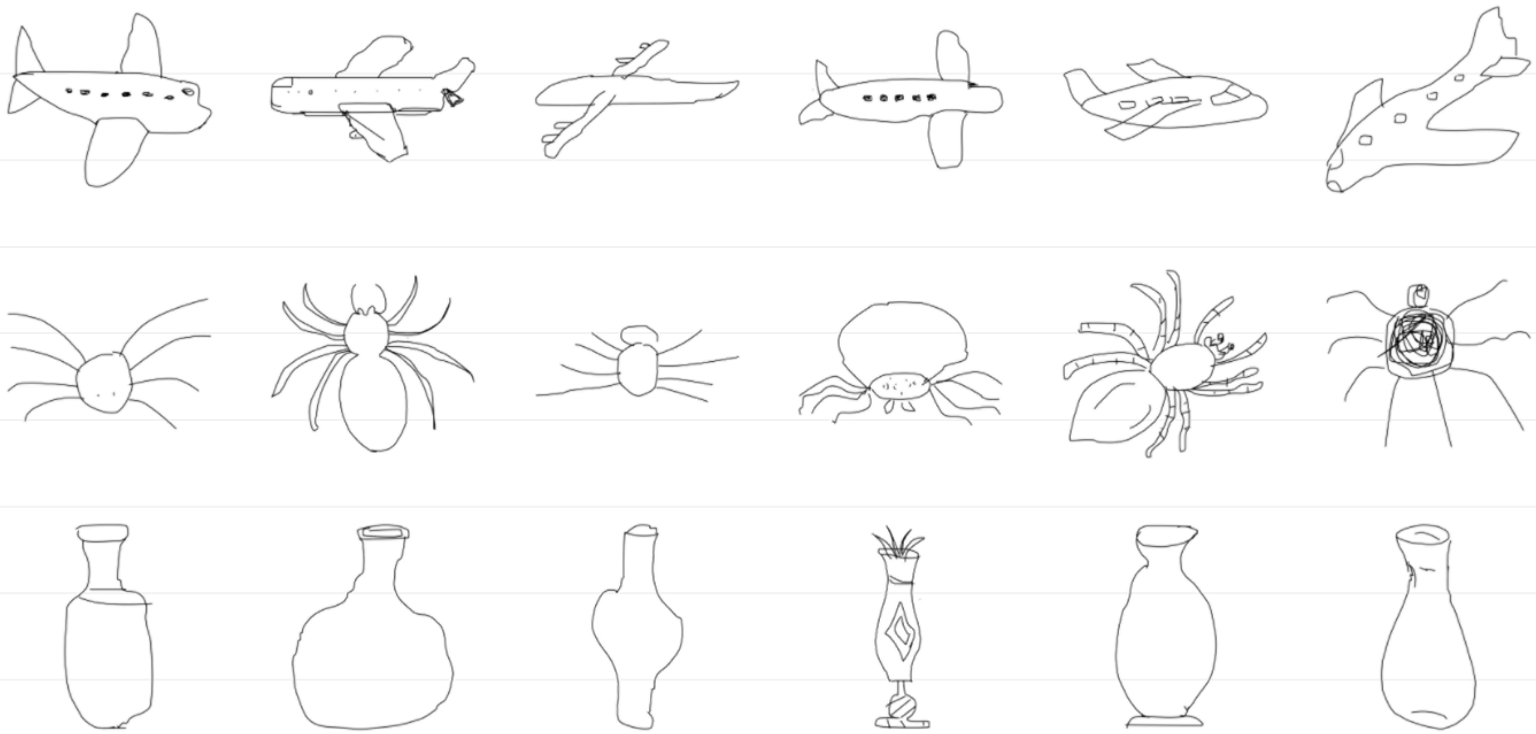
SKETCHES RECOGNITION



— Students: F. Minutella | A Schiavo | M. Daole | M. Gómez || Professors: C. Gennaro | F. Falchi | G. Amato | P. Bolettieri —

Project Description

1. Implement index LSH to allow fast similarity search on deep features and create an image search engine on top of it
2. Use the pre trained Deep Neural Network Inception to extract features from the dataset Sketches and the distractor MirFlickr
3. Index the extracted features using your search engine
4. Measure the retrieval performance of the image search engine
5. Fine tune the Inception for Sketches
6. Use the fine tuned Inception to extract features from Sketches and MirFlickr
7. Index the new extracted features using your search engine
8. Measure the retrieval performance of the image search engine using the new features
9. Compare the performance of the two features
10. Optional:
    - Build a web based user interface for your web search engine

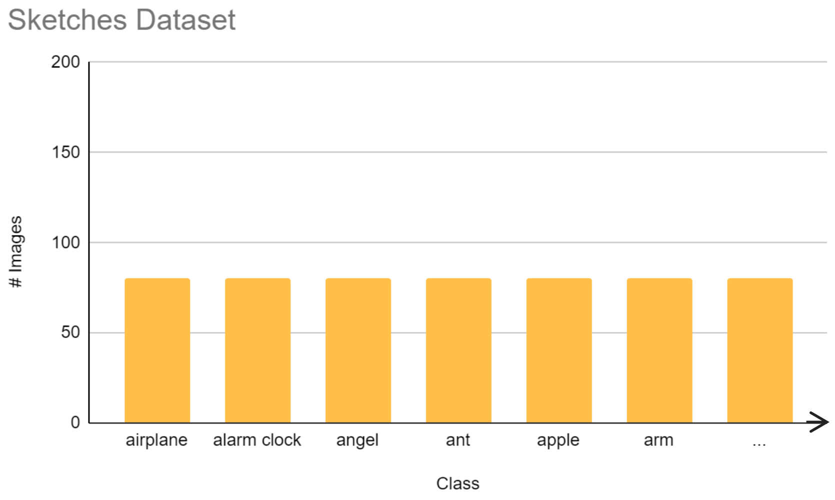


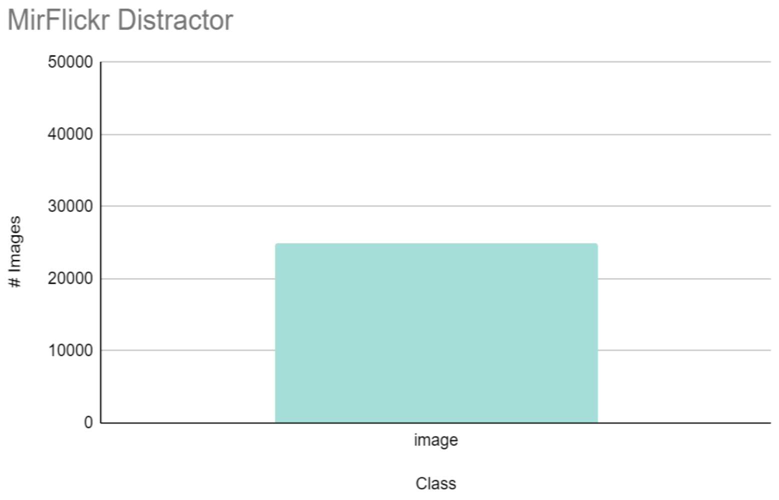
Our goal is to train machines to recognize and generalize abstract concepts in a manner similar to humans. As a first step towards this goal, we train our model on a dataset of hand-drawn sketches.

Distribution of our Datasets

250 Class and 20K Images

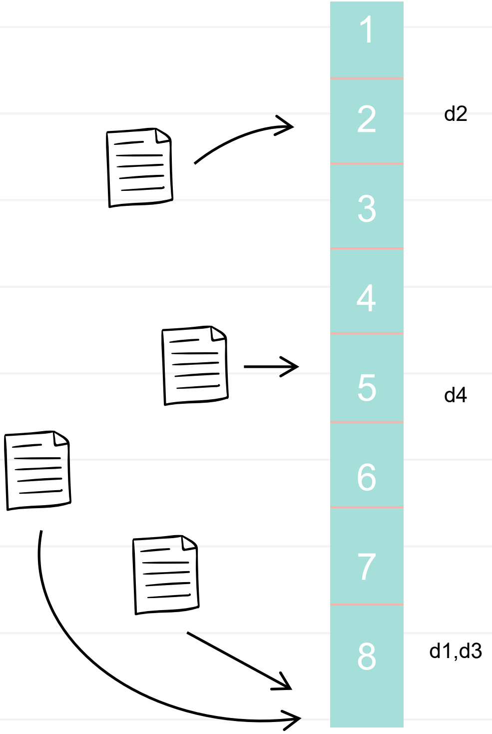
1 Class and 25K Images



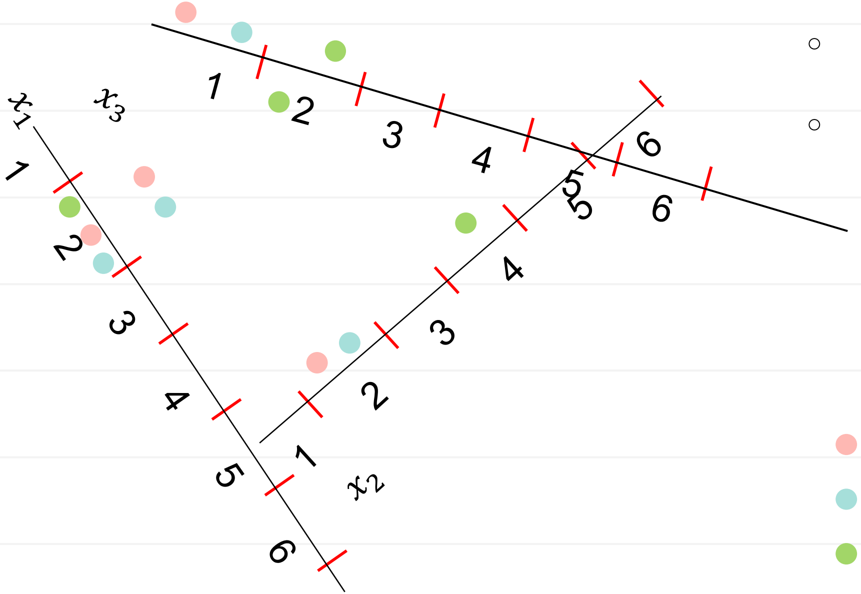


LSH – Locality Sensitive Hashing

- We don’t necessarily insist on the exact answer; instead, determining an approximate answer should suffice.
  - Similar Documents => Similar Hash-Code
- For each document d:
  - Generate K hash-code
  - Insert Document into hash-table



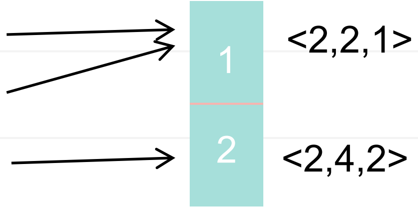
LSH - Locality Sensitive Hashing



- Hash functions `g()` such that given any two objects `p,q`
  - If $d(p,q) > c*r$ then $Pr[g(p)= g(q)]$ is small
  - If $d(p,q) \leq r$ then $Pr[g(p) = g(q)]$ is not so small

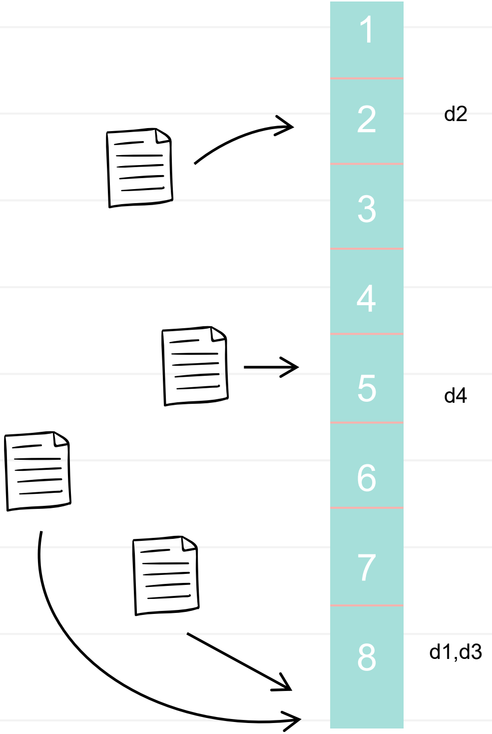
Definition: $g(p)=\langle h_1(p), h_2(p), ..., h_k(p), \rangle$
Where: $h_i(p) = \lfloor (p*X_i+b_i)/w \rfloor$

	x_1	x_2	x_3	
● (pink)	2	2	1	<2,2,1>
● (teal)	2	2	1	<2,2,1>
● (green)	2	4	2	<2,4,2>



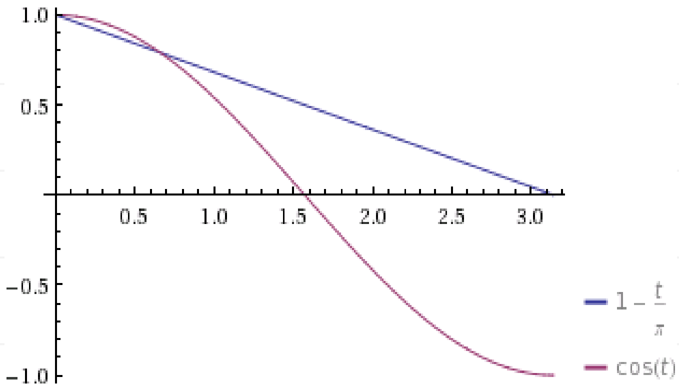
LSH - Implementation

- Preprocessing: Hash function selecting $<g_1, g_2, ..., g_L>$
- Insertion: Any point p Insert into L buckets $<g_1(p), g_2(p), ..., g_L(p)>$
- Query execution: With the query q retrieve all point from L buckets $g_1(q)$ and reorder according to the original distance function

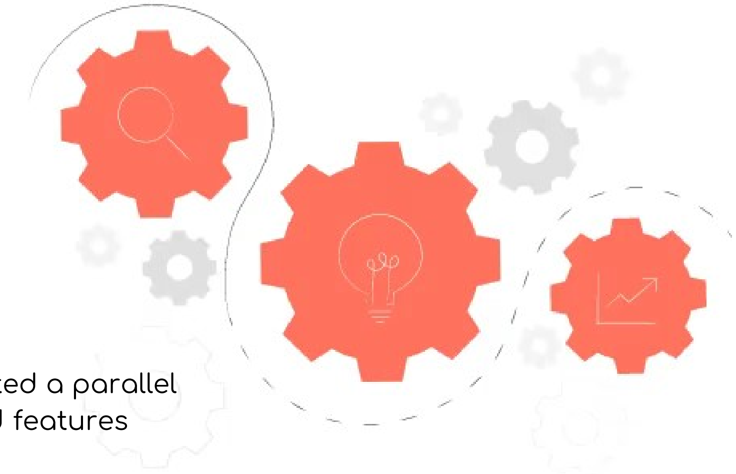


LSH – Bitwise

- The random projection method of LSH called SimHash is designed to approximate the cosine distance between vectors.
- The basic idea of this technique is to choose a random hyperplane at the outset and use the hyperplane to hash input vectors.
    - $h(v) = \pm 1$
- We called Bitwise because, instead ± 1 we convert this approach to 0 or 1 (bit)
- This approach is faster and require less storage



No Index Structure – Test



- Beside implementing the LSH index structure, we implemented a parallel structure **without any indexing,** just storing all the extracted features together.

- Why we did that ?

  - In order to test and tune CNNs without the bias introduced by the LSH index approximation.

  - In order to compare the retrieval results and performances without index against the ones obtained by using the LSH index.

Features Extraction

- Feature Extraction using the pretrained convolutional base

01

Create a image normalization preprocess with our images

02

Loading the pretrained DNN

03

Extract of the features

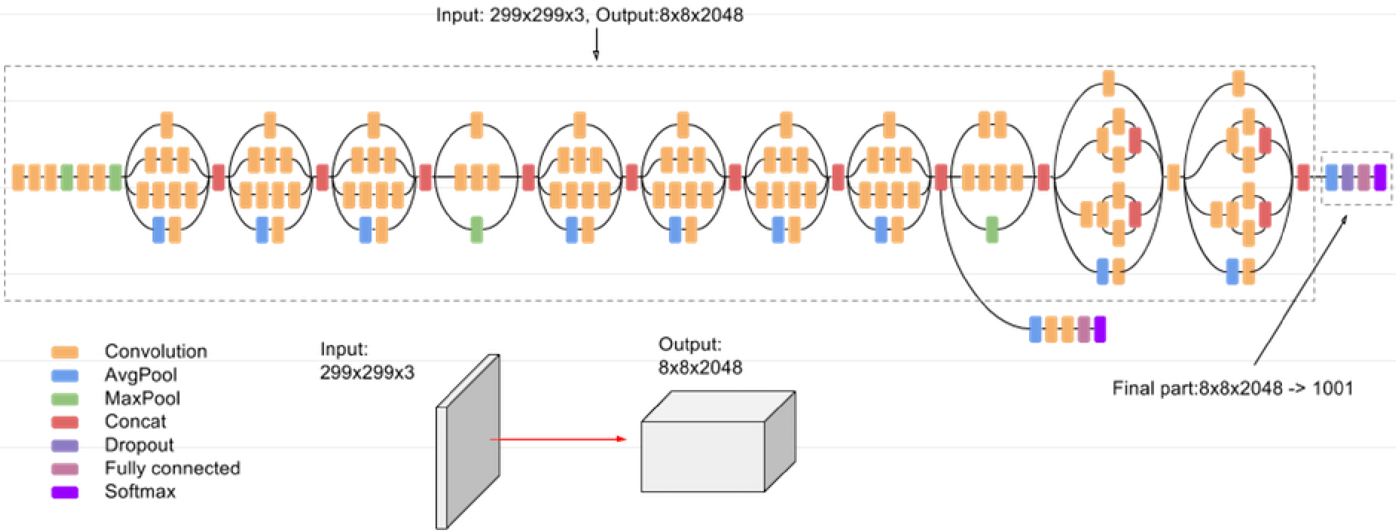
04

The result is our feature vector

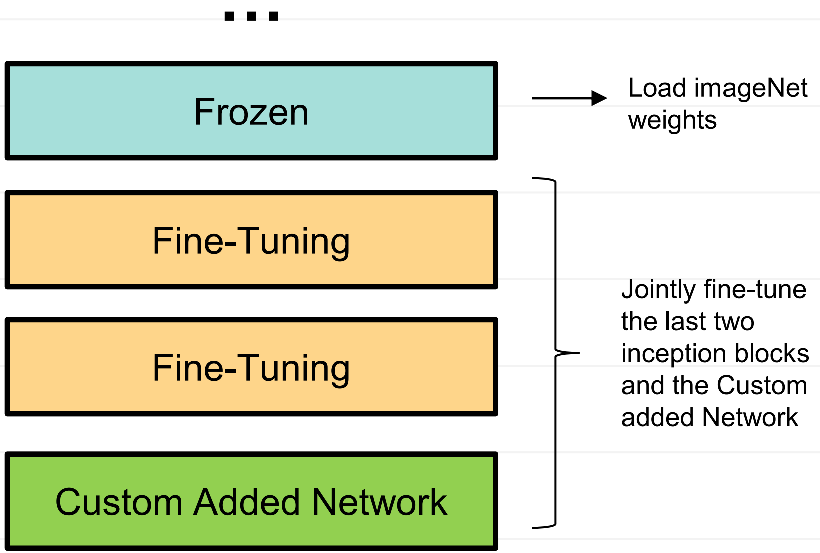
High Level Architecture

Inception V3

- Have a very important milestone in the development of CNN classifiers.
- The fundamental idea behind the Inception Neural Network is the inception block
- Intermediate Classifiers to solve Vanishing Gradient.



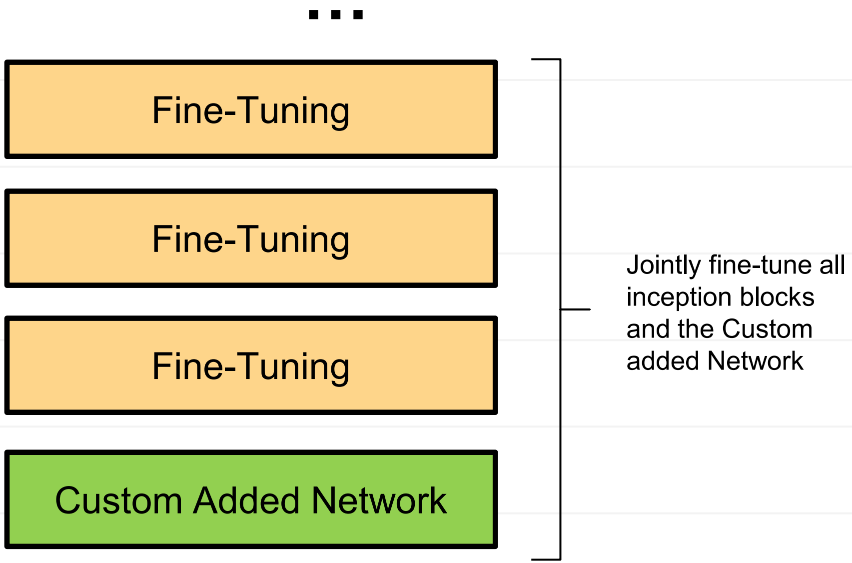
Neural Network Fine Tuning Structure



1. Add the custom network on top of an already-trained base network
2. Freeze the base network
3. Train the part we added.
4. Unfreeze two inception blocks in the base network.
5. Jointly train both these layers and the part we added.

Note: each of the model has been trained by adopting data augmentation to generate more samples from already existing training data and to prevent overfitting.

Neural Network Fine Tuning Structure

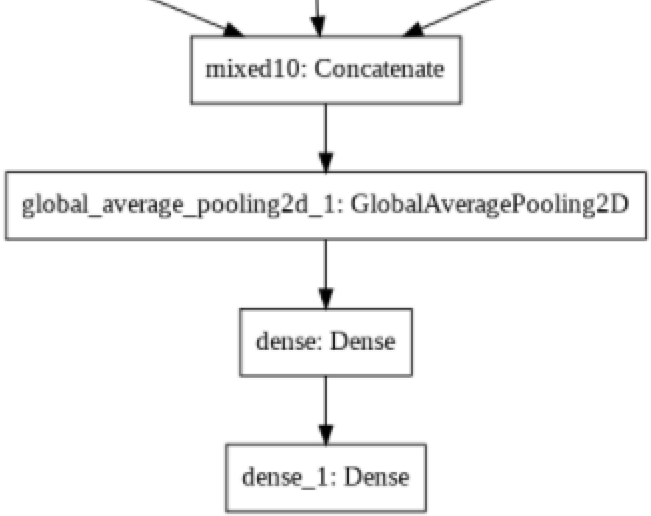


1. Add the custom network on top of an already-trained base network
2. Freeze the base network
3. Train the part we added.
4. Unfreeze the entire inception convolutional base network.
5. Jointly train both these layers and the part we added.

Note: each of the model has been trained by adopting data augmentation to generate more samples from already existing training data and to prevent overfitting.

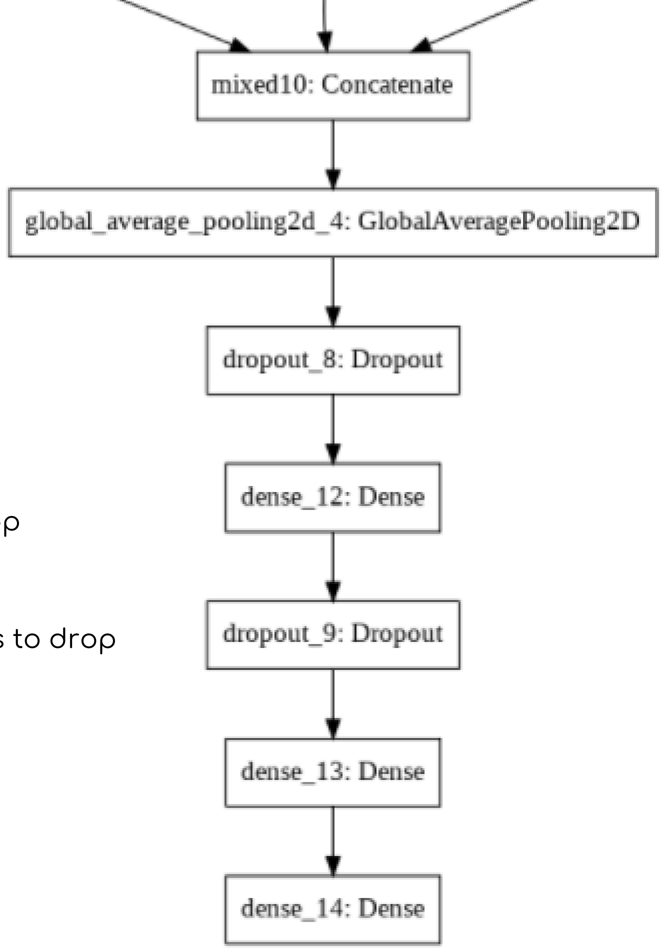
Model 1

- Model with the next custom layers:
    1. Pre-trained model Inception V3
    2. Global spatial average pooling layer
    3. First fully connected layer with 1024 dimensionality
    4. Second fully connected layer with 250 dimensionality



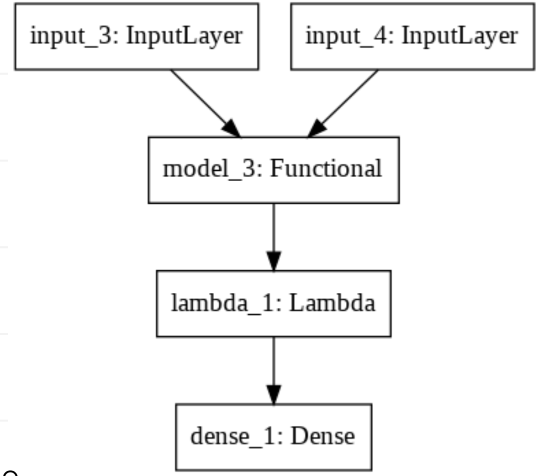
Model 2

- Model with the next custom layers:
    1. Pre-trained model Inception V3
    2. Global spatial average pooling layer
    3. First droupout layer with a fraction of 0.5 input units to drop
    4. First fully connected layer with 2048 dimensionality
    5. Second droupout layer with a fraction of the 0.5 input units to drop
    6. Second fully connected layer with 2048 dimensionality
    7. Third fully connected layer with 250 dimensionality



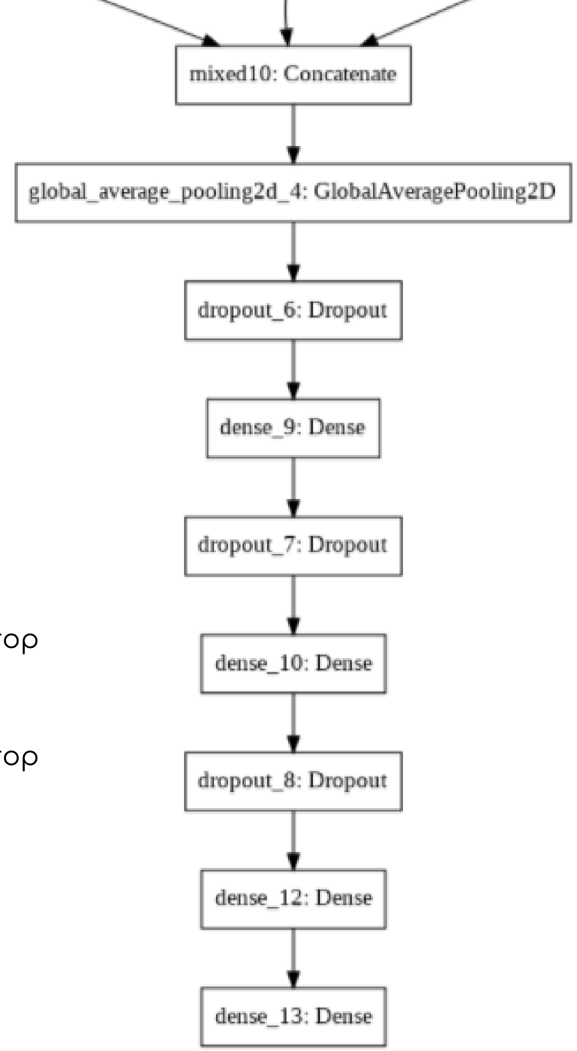
Model 3 (Siamese)

- Model with the next custom layers:
    1. Siamese neural network with the pre-trained custom model 3
    2. Lambda layer to compute the absolute difference between the encodings `(lambda tensors:K.abs(tensors[0] - tensors[1])`
    3. First fully connected layer with 1 dimensionality with a sigmoid unit to generate the similarity score



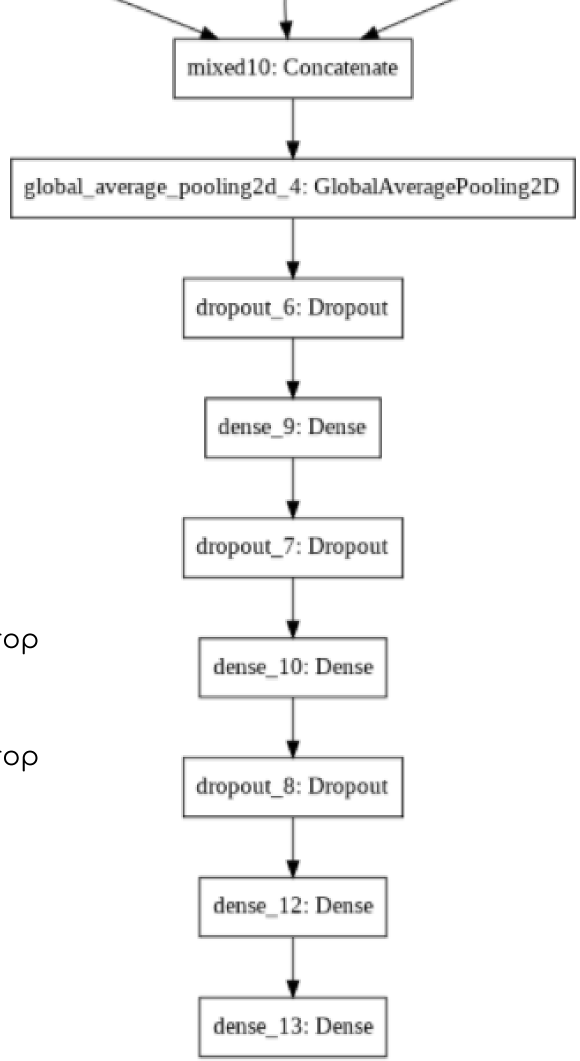
Model 4

- Model with the next custom layers:
    1. Pre-trained model Inception V3
    2. Global spatial average pooling layer
    3. First droupout layer with a fraction of 0.5 input units to drop
    4. First fully connected layer with 2048 dimensionality
    5. Second droupout layer with a fraction of the 0.5 input units to drop
    6. Second fully connected layer with 2048 dimensionality
    7. Second droupout layer with a fraction of the 0.5 input units to drop
    8. Third fully connected layer with 2048 dimensionality
    9. Fourth fully connected layer with 250 dimensionality



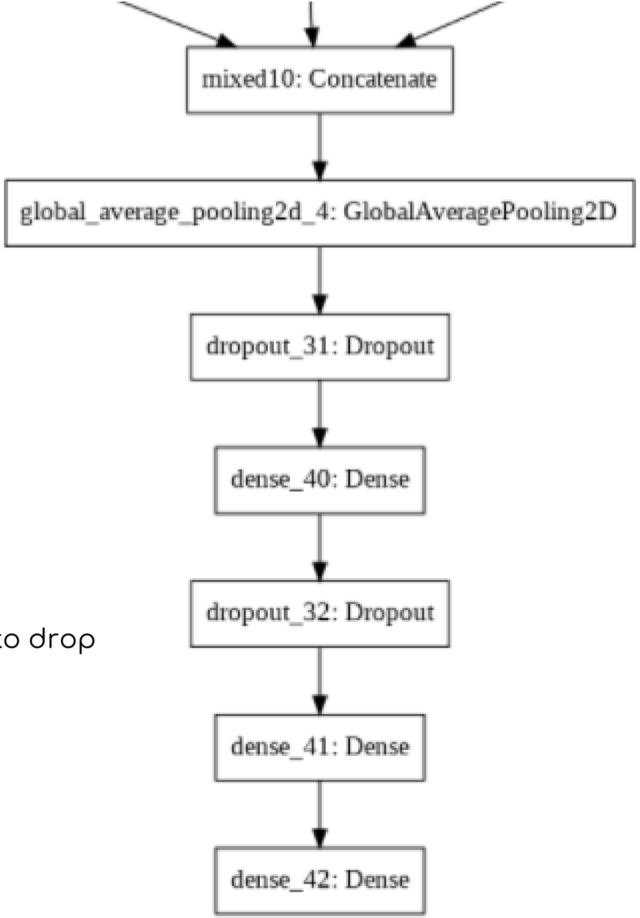
Model 5 (Best Model)

- Model with the next custom layers:
    1. Pre-trained model Inception V3
    2. Global spatial average pooling layer
    3. First droupout layer with a fraction of 0.5 input units to drop
    4. First fully connected layer with 2048 dimensionality
    5. Second droupout layer with a fraction of the 0.5 input units to drop
    6. Second fully connected layer with 2048 dimensionality
    7. Second droupout layer with a fraction of the 0.5 input units to drop
    8. Third fully connected layer with 2048 dimensionality
    9. Fourth fully connected layer with 250 dimensionality



Model 6

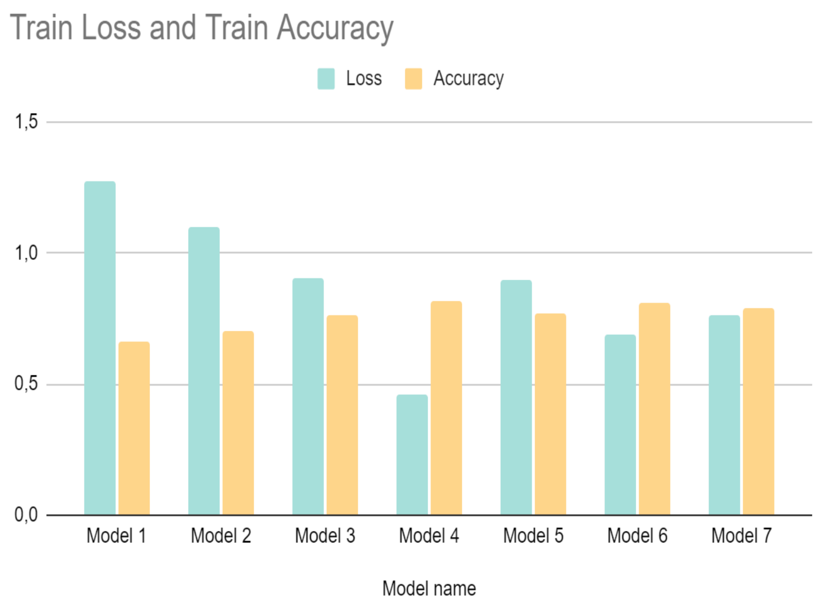
- Model with the next custom layers:
    1. Pre-trained model Inception V3
    2. Global spatial average pooling layer
    3. First droupout layer with a fraction of 0.5 input units to drop
    4. First fully connected layer with 4096 dimensionality
    5. Second droupout layer with a fraction of the 0.5 input units to drop
    6. Second fully connected layer with 4096 dimensionality
    7. Third fully connected layer with 250 dimensionality

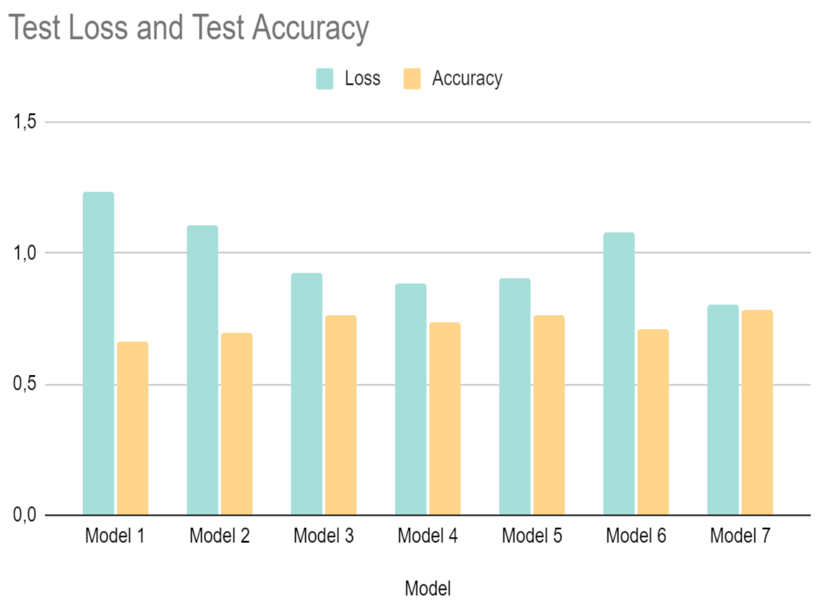


Model Test and Performance

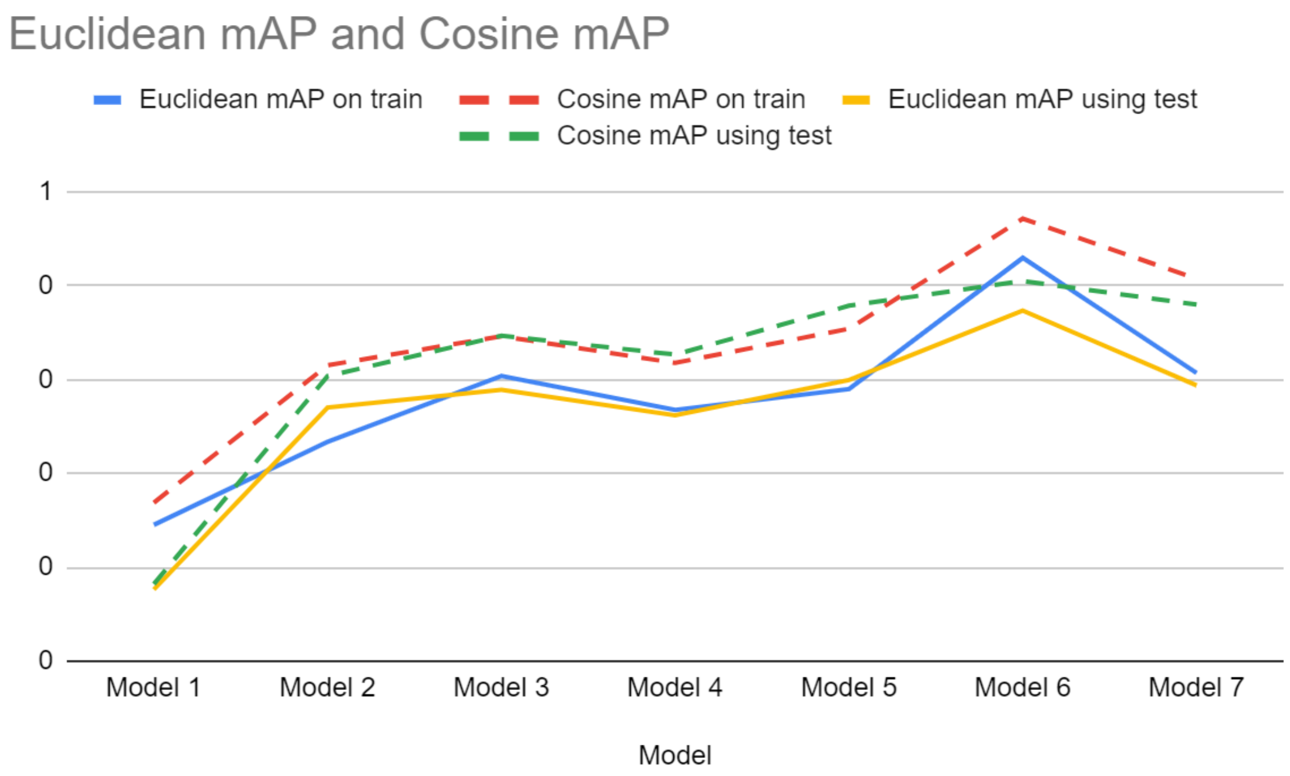
Model name	Train		Test		Euclidean mAP on train	Cosine mAP on train	Euclidean mAP using test	Cosine mAP using test
	Loss	Accuracy	Loss	Accuracy				
Model 1	1.2724	0.6617	1.2328	0.6648	0.1452	0.1687	0.0763	0.0821
Model 3 (Siamese)	0.4611	0.8157	0.8868	0.7373	0.2674	0.3175	0.2618	0.3264
Model 2	0.9033	0.7612	0.9272	0.762	0.3035	0.346	0.2889	0.3464
Model 4	0.894	0.7667	0.9065	0.761	0.2896	0.3539	0.2993	0.3784
Model 6	0.76	0.7867	0.8022	0.784	0.3068	0.4071	0.2935	0.3797
Model 5 (Best)	0.6899	0.8087	1.0764	0.709	0.4294	0.4711	0.3731	0.4047

Model Test and Performance



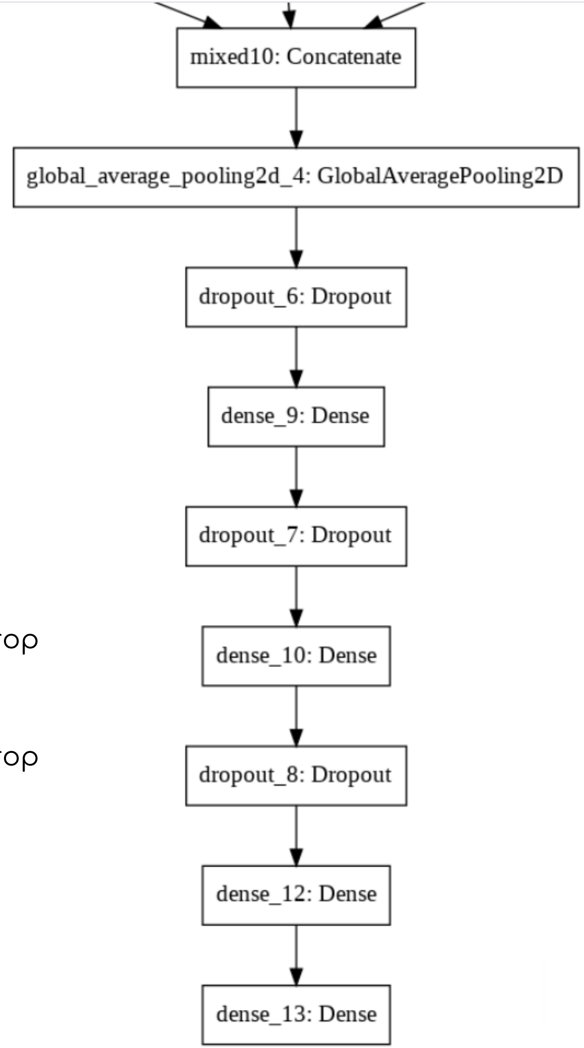


Model Test and Performance



Best Model Chosen: Model 5

- We choose the model with the next custom layers:
    1. Pre-trained model Inception V3
    2. Global spatial average pooling layer
    3. First droupout layer with a fraction of 0.5 input units to drop
    4. First fully connected layer with 2048 dimensionality
    5. Second droupout layer with a fraction of the 0.5 input units to drop
    6. Second fully connected layer with 2048 dimensionality
    7. Second droupout layer with a fraction of the 0.5 input units to drop
    8. Third fully connected layer with 2048 dimensionality
    9. Fourth fully connected layer with 250 dimensionality



LSH Tuning (G and H)

LSH	Test mAP		Test IE		Bucket Purity AVG	Bucket Purity STD	# Buckets	# Items	# AVG per BUCKET	# STD per BUCKET
	Euc	Cos	Euc	Cos						
G = 3, H = 5	0,01	0,02	3775,31	4000,96	0,87	0,24	24363	120000	4,93	36,3
G = 3, H = 3	0,06	0,06	387,62	269,23	0,72	0,32	4885	120000	24,5	183,97
G = 5, H = 2	0,18	0,19	6,68	9,31	0,61	0,35	1629	200000	122,77	642,32
G = 4, H = 2	0,17	0,20	11,2	14,3	0,62	0,34	1188	160000	134,68	677,15
G = 7, H = 2	0,24	0,26	5,8	3,9	0,62	0,34	2319	280000	120,7	722,83
G = 8, H = 2	0,25	0,27	1,9	2,69	0,62	0,35	2548	320000	125,59	688,5
G = 5, H = 1	0,31	0,37	0,52	0,46	0,52	0,35	150	200000	1333,3	3571,97

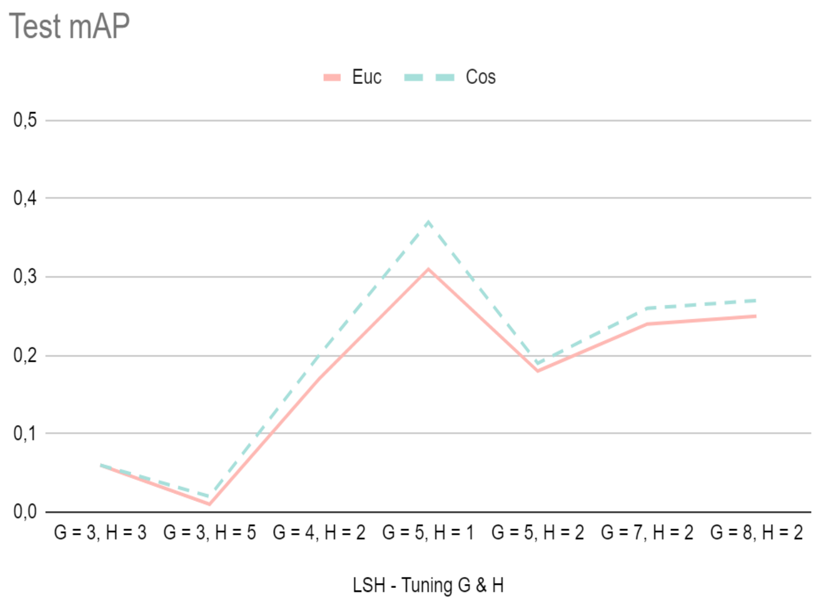
$Improvement\ Efficency = \frac{Cost\ No\ Index}{Cost\ with\ Index}$ Where $Cost = \#\ of\ Computed\ Distances$

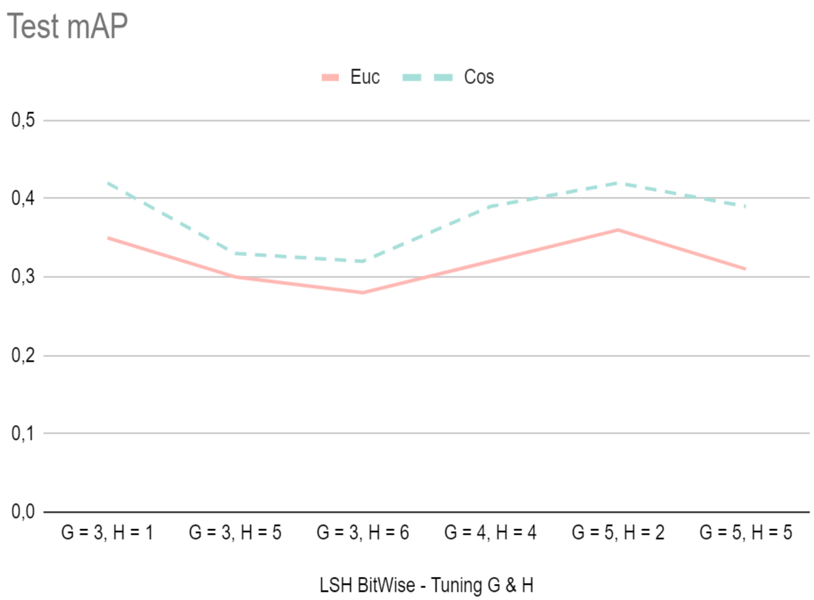
LSH BitWise - Tuning (G and H)

LSH BitWise	Test mAP		Test IE		Bucket Purity AVG	Bucket Purity STD	# Buckets	# Items	# AVG per BUCKET	# STD per BUCKET
	Euc	Cos	Euc	Cos						
G = 3, H = 6	0,28	0,32	8	8,2	0,3	0,25	192	120000	625	1502,5
G = 3, H = 5	0,30	0,33	5	4,67	0,26	0,27	96	120000	1250	2733
G = 4, H = 4	0,32	0,39	1,36	1,37	0,37	0,32	64	160000	2500	3220,32
G = 5, H = 5	0,31	0,39	2,23	2,2	0,29	0,26	160	200000	1250	2347
G = 3, H = 1	0,35	0,42	0,26	0,26	0,52	0,27	6	120000	20000	8807
G = 5, H = 2	0,36	0,42	0,31	0,31	0,35	0,36	20	200000	10000	8618,35

$Improvement\ Efficency = \frac{Cost\ No\ Index}{Cost\ with\ Index}$ Where $Cost = \#\ of\ Computed\ Distances$

Test mAP - Graph comparation





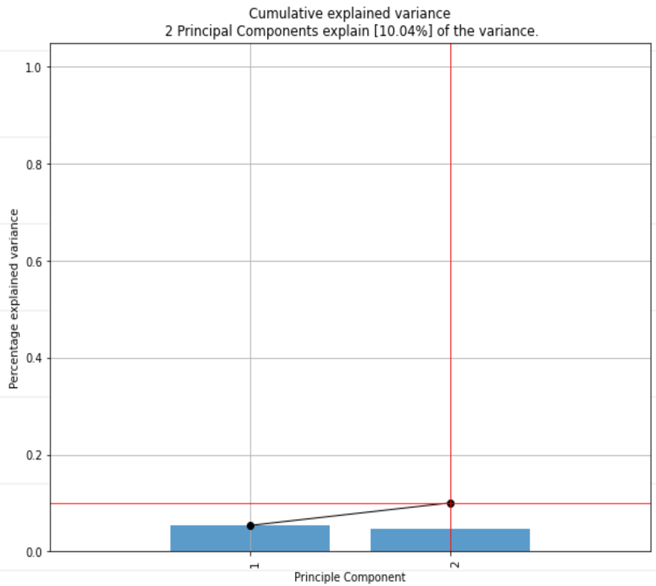
PCA – Principal Component Analysis

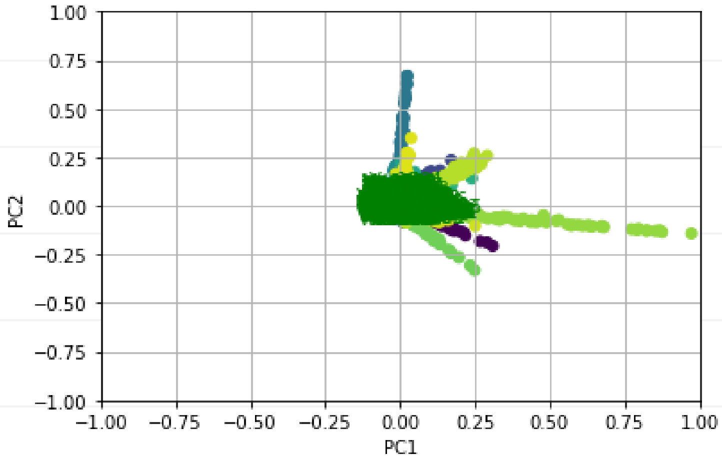
We tried to carry out the PCA in order to see:

- The possibility of represent features extracted in the new space obtained by using the first two principal components.

But, the first two principal components only explain 10% of the variance of our data:

- As you can see in the second graph, the features of the various classes are largely overlapped with no clear clusters distinguishable.





ACCURACY AND MAP ANALYSIS CLASS BY CLASS

01

For each class, iteratively query the sketches dataset, using each time one object of the class as query to retrieve all the others.

02

Compute the Average Precision for each query

03

For each class, compute mAP, min(AP), max(AP), std(AP), range(AP)

04

Return the Sketch for which the max(AP) score has been obtained and the sketch for which the mix(AP) score has been obtained

```
for each class c
    for each class sample s
        execute_query(s)
        compute_AP(s)
    compute_mAP(c)
    compute_max(AP, c)
    compute_min(AP, c)
    compute_range(AP, c)
    compute_std(AP, c)
    show_best_sketch(c)
    show_worst_sketch(c)
```

Mean Average Precision

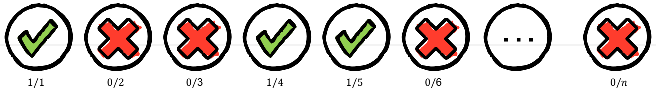
Our use case is that a user wants to query the sketches dataset by using a certain sketch as query object. So let q be the user query, m be the set of relevant sketches {s1, ..., sm} for q, Rk be the set of ranked retrieval results (from the most to the least similar according to some similarity measure). Then we compute the Average Precision AP(q) as

Average Precision: $AP(q) = \frac{1}{m}\sum_{k=1}^{m} Precision(R_K)$ $\quad Precision(R_K) = \begin{cases} 1 \; if \; d_K \; is \; relevant \\ 0 \;\; if \, d_K \; is \; not \; relevant \end{cases}$

Average precision computation example in which we have 3 relevant sketches:

AP = $^1/_3$ ($^1/_1 + {^0/_2} + {^0/_3} + {^2/_4} + {^3/_5} + {^0/_6} + 0$) = 0.7

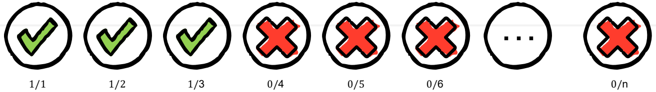
$\frac{True\ Positives}{Predicted\ positives}$ =



For another query, q, we could get a perfect AP of 1 if the retrieval results are ordered like this :

AP = $^1/_3$ ($^1/_1 + {^2/_2} + {^3/_3} + {^0/_4} + {^0/_5} + {^0/_6} + 0$) = 1.0

$\frac{True\ Positives}{Predicted\ positives}$ =

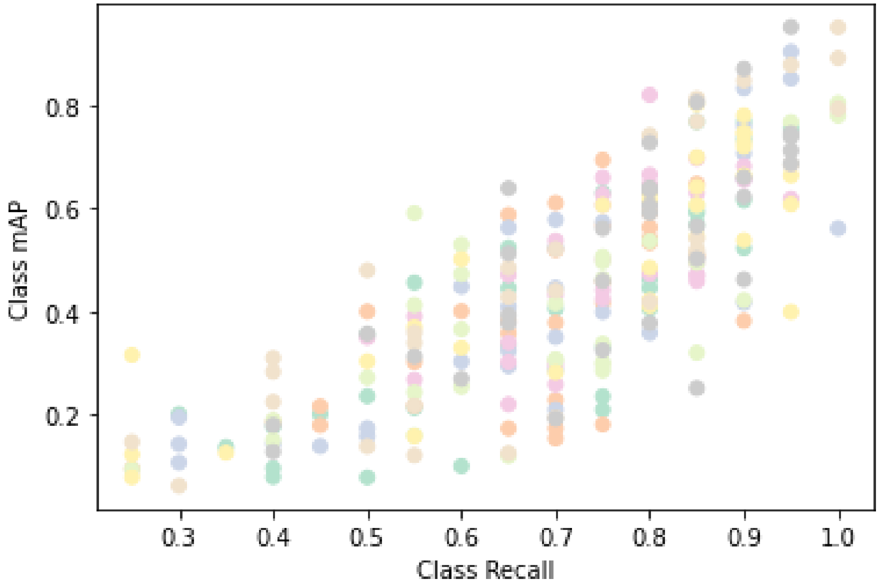


Suppose the user performs a set Q of queries q, then we compute the Mean Average Precision mAP(Q) as

Mean Average Precision: $mAP(Q) = \frac{1}{|Q|}\sum_{q=1}^{|Q|} AP(q)$

Recall Vs Mean Average Precision

Beside computing the mAP for each of the 250 sketches classes, we even computed the Recall for each class. What is the percentage of samples belonging to each class correctly recognized (labeled) by our deep learning model?
Then we carried out a **correlation analysis** between **Recall and mAP by class**, below the scatter plot is reported.

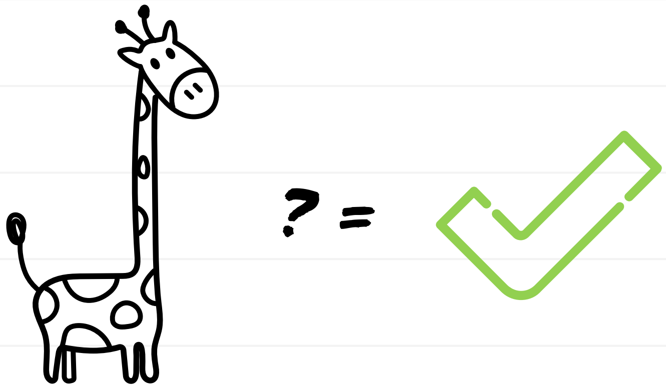


Correlation coefficient:

0.7863

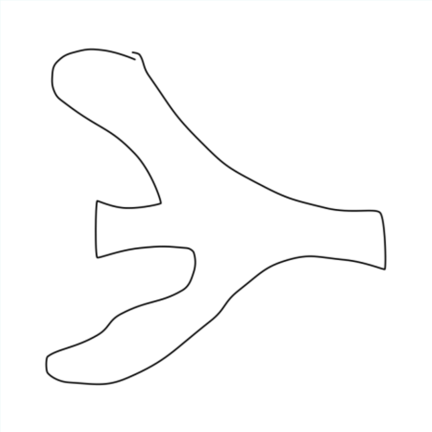
There is a significant positive correlation between Recall and mAP by class.

Results for some of the well recognized classes

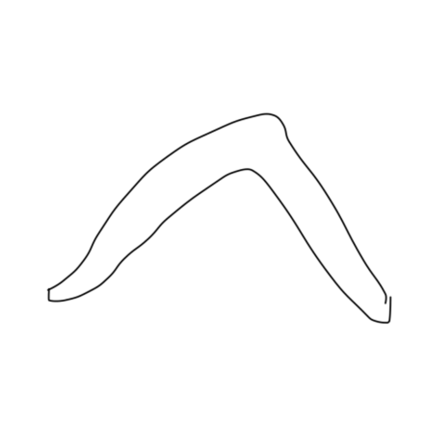


Class Boomerang: mAP 0.77 Recall 0.90

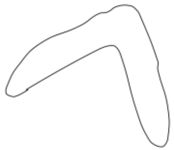

```
{
    "class": "boomerang",
    "label": 26,
    "mAP": 0.7673,
    "min AP": 0.0193,
    "max AP": 0.9367,
    "max-min range": 0.9166,
    "std": 0.2511
}
```

Worst sketch:
AP = 0.0125



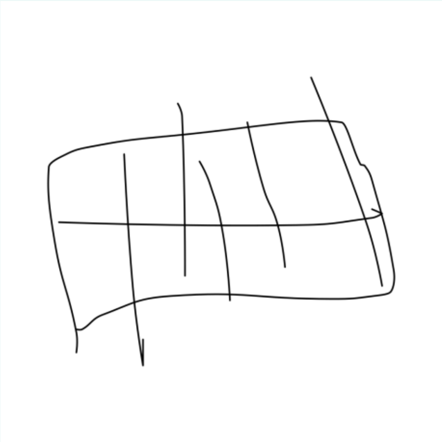
Best sketch:
AP = 0.936



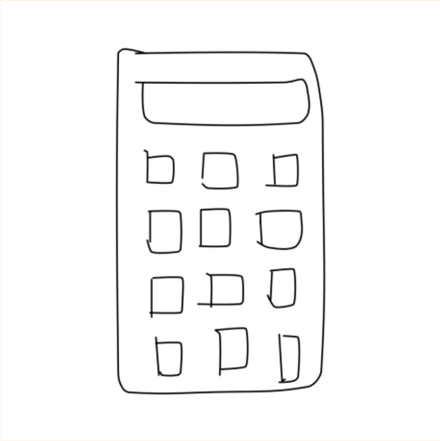
Some other sketches from the boomerang class:

Class Calculator: mAP 0.65 Recall 0.85


```
{
    "class": "calculator",
    "label": 39,
    "mAP": 0.6472,
    "min AP": 0.0130,
    "max AP": 0.8586,
    "max-min range": 0.8455,
    "std": 0.2788
}
```

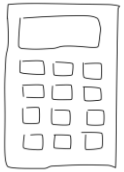



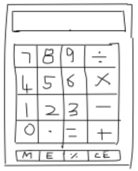
Worst sketch:
AP = 0.0131

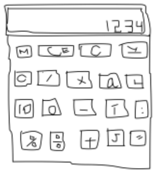


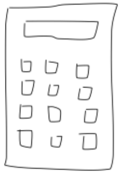
Best sketch:
AP = 0.8586

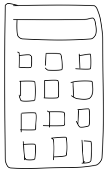
Some other sketches from the calculator class:

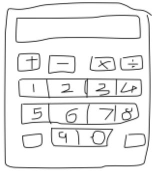




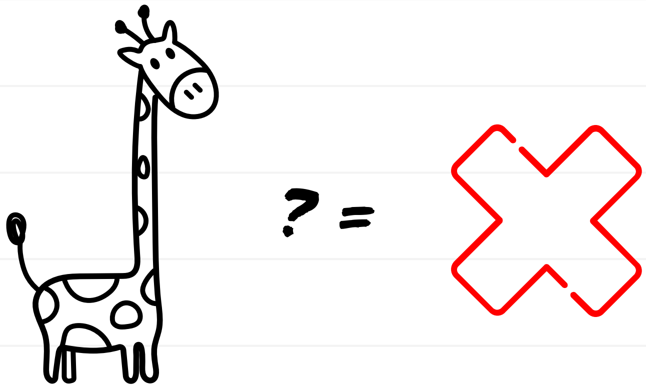






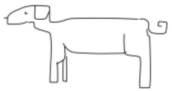


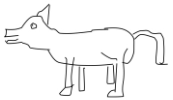
RESULTS FOR SOME OF THE BAD RECOGNIZED CLASSES

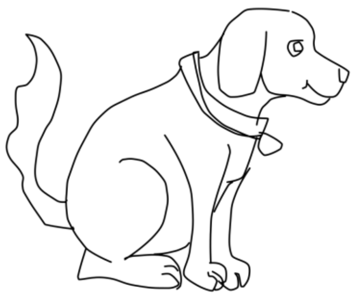


Class Dog: mAP 0.10 Recall 0.30

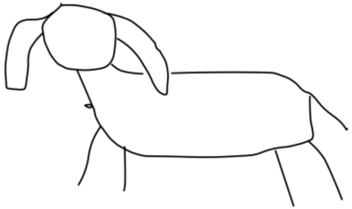
{
    "class": "dog",
    "label": 66,
    "mAP": 0.1077,
    "min AP": 0.0125,
    "max AP": 0.3210,
    "max-min range": 0.3085,
    "std": 0.0844
}



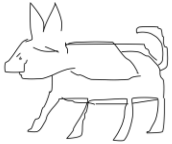




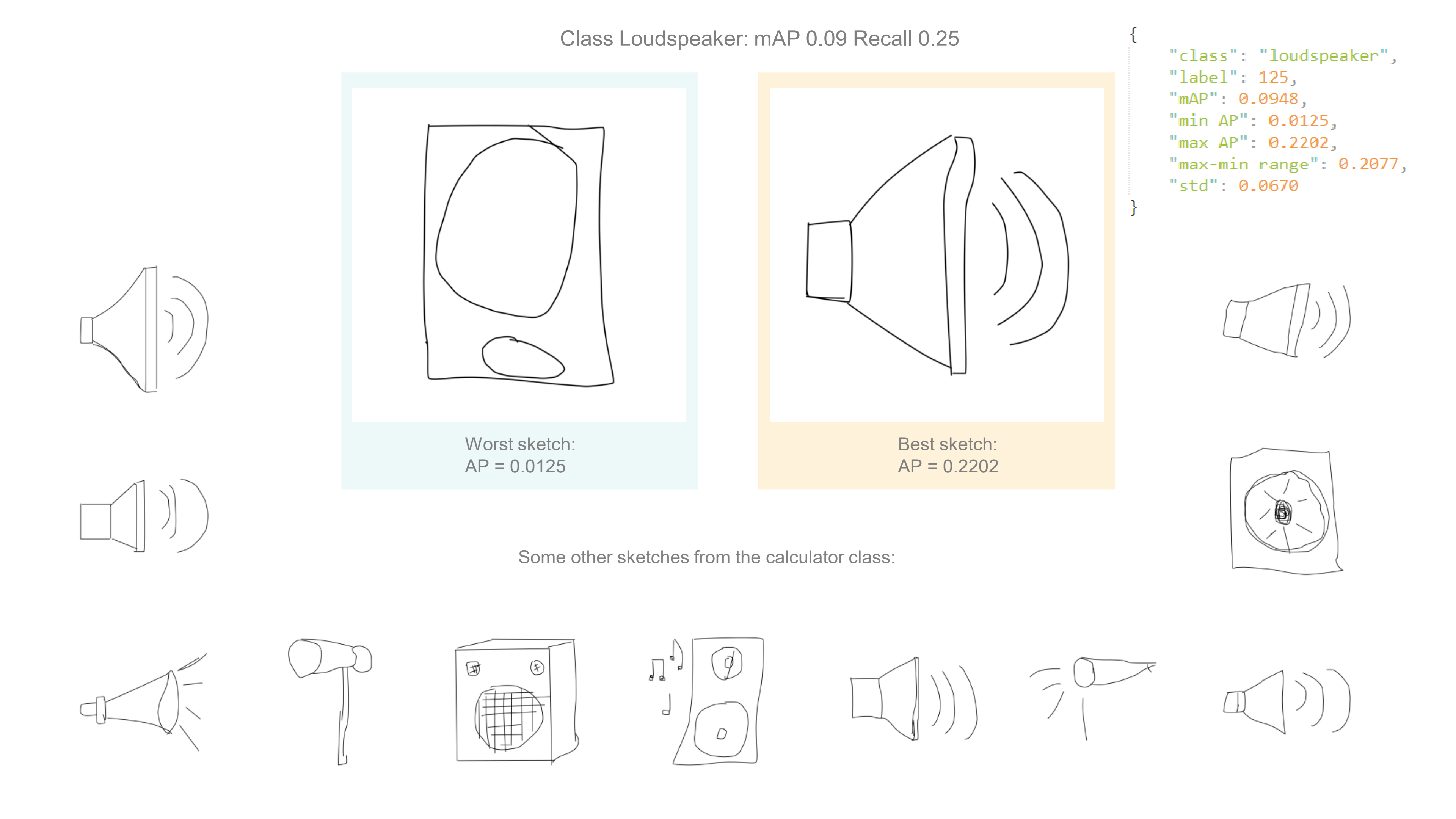
Worst sketch:
AP = 0.012



Best sketch:
AP = 0.3210

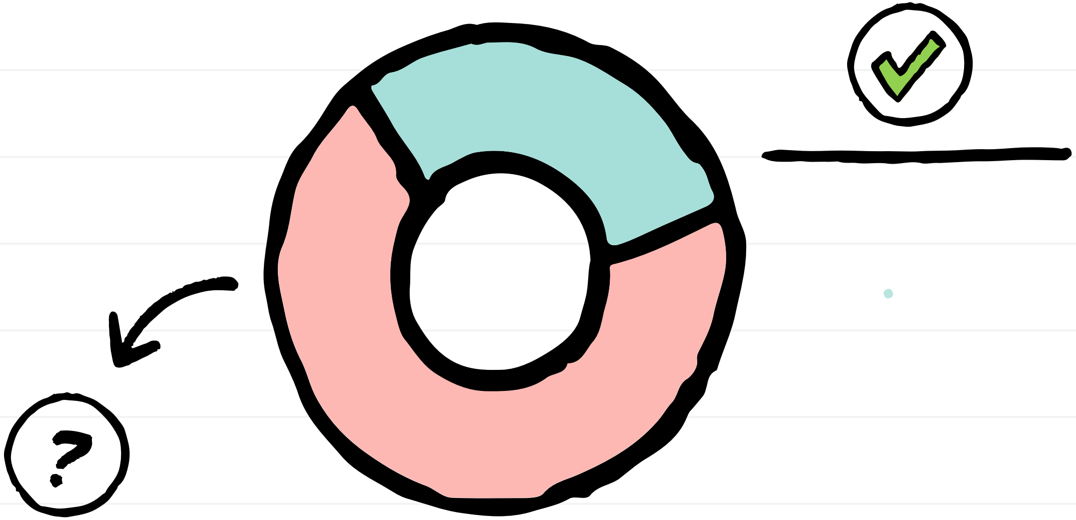


Some other sketches from the calculator class:


Class Loudspeaker: mAP 0.09 Recall 0.25
{
"class": "loudspeaker",
"label": 125,
"mAP": 0.0948,
"min AP": 0.0125,
"max AP": 0.2202,
"max-min range": 0.2077,
"std": 0.0670
}
Worst sketch:
AP = 0.0125
Best sketch:
AP = 0.2202
Some other sketches from the calculator class:

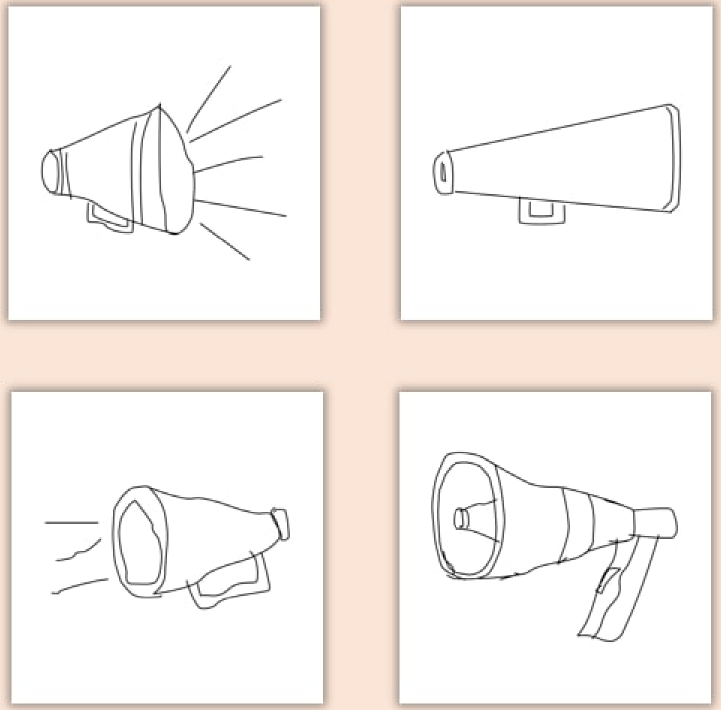
Most Frequent Misclassifications

For some of the worse recognized classes, we got the predictions in output from our deep learning model in order to check with which other classes the model makes confusion.

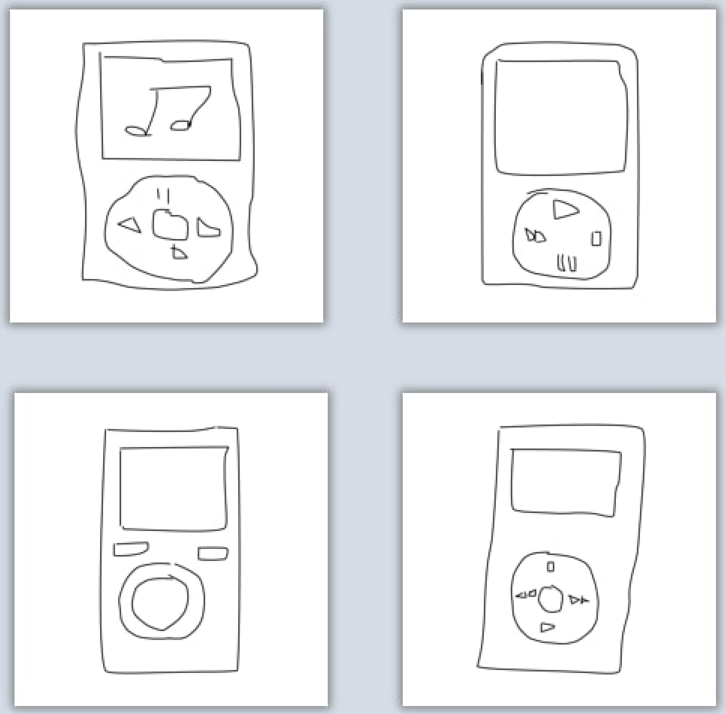


Loudspeaker Class: most frequent misclassifications

4/20 times misclassified as Megaphone:

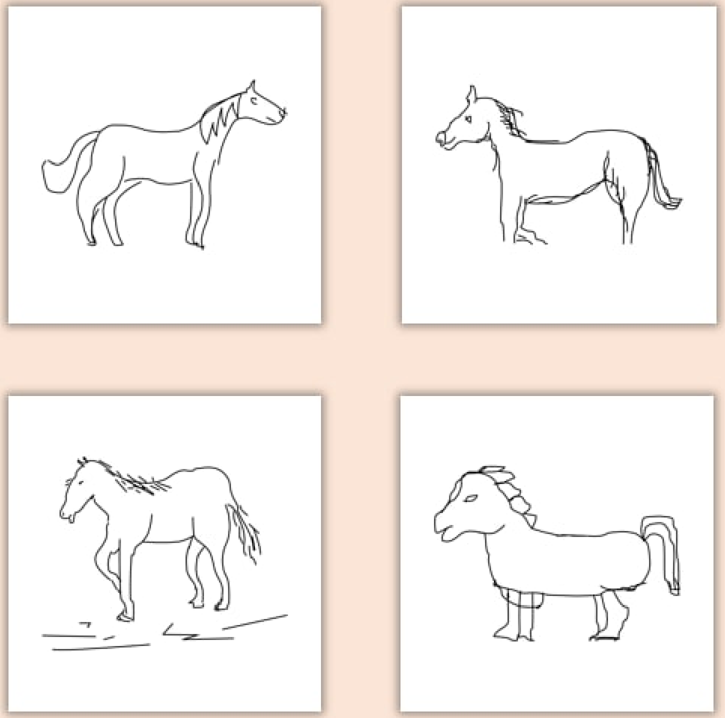


3/20 times misclassified as Ipod:

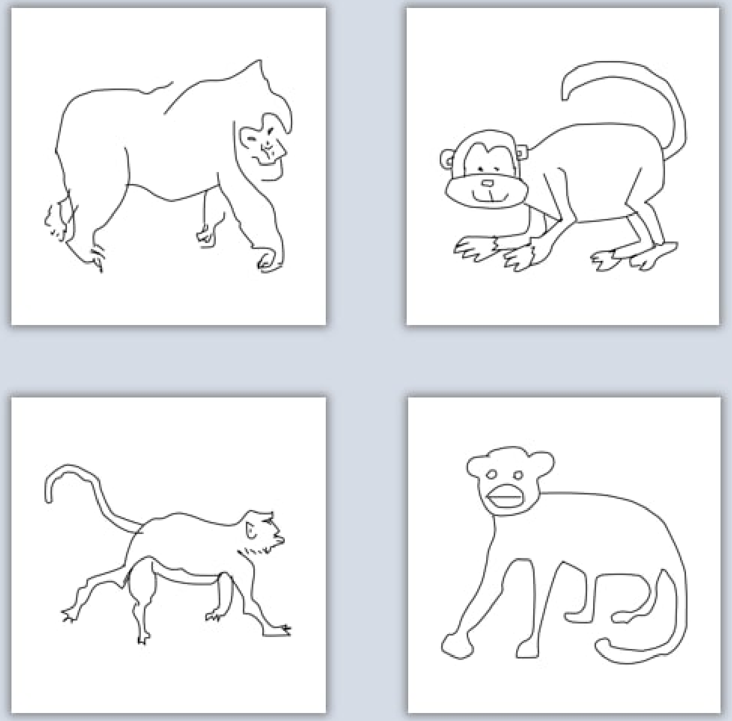


Dog Class: most frequent misclassifications

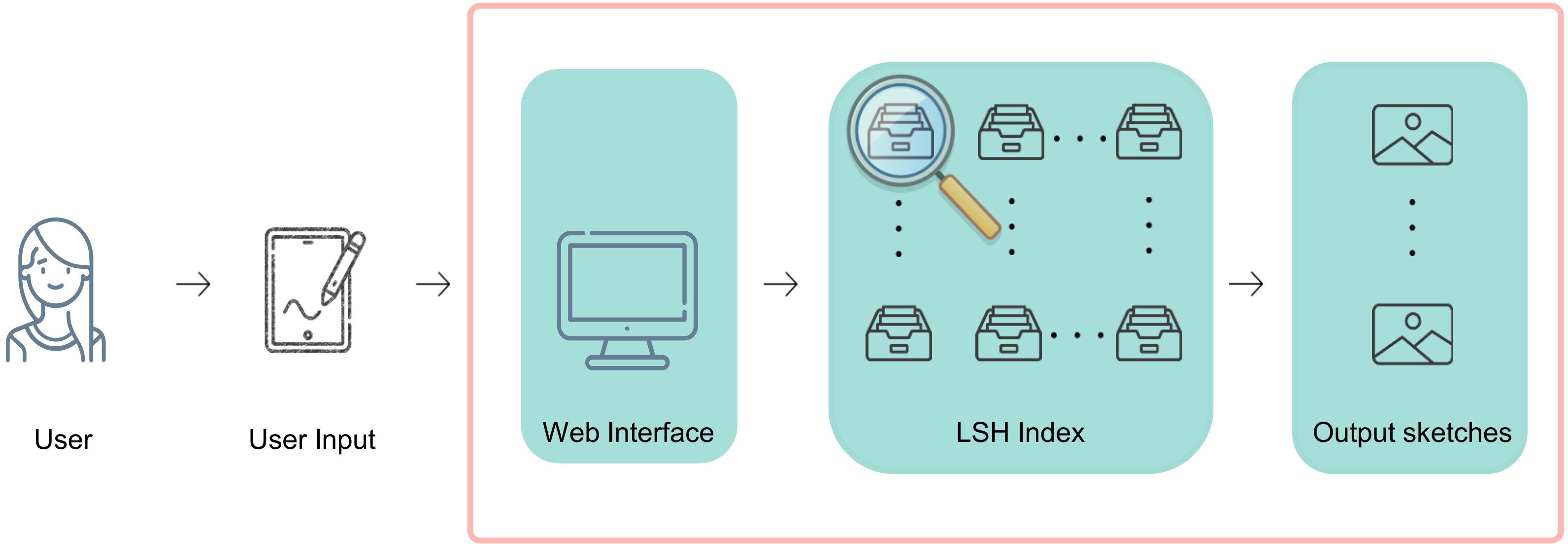
4/20 times misclassified as Horse:



3/20 times misclassified as Monkey:



Web Sketches Recognition Architecture



DEMO

Project Source Code

- Analysis_By_Class
- MIRCV
- MIRCV_Functions
- Pca_Try
- accuracy_assessment
- evaluate_models
- lsh_finetuning
- lsh_finetuning_bucket_dispersion
- mAP_by_class
- save_index
- siamese_finetuning
- sketches
- web

RESOURCES

Papers

- Rethinking the Inception Architecture for Computer Visual
- A Simple Guide to the Versions of the Inception Network
- A Neural Network for PCA and Beyond
- Theory and Applications of b-Bit Minwise Hasing
- A Neural Network for PCA
- Locality-sensitive hashing
- Locality-Sensitive Hashing Scheme Based on p-Stable Distributions
- Similarity Search in High Dimensions via Hashing
- How Do Humans Sketch Objects?
- One Shot Learning with Siamese Network using Keras

THANKS!