Dokumentation

Selected Topics in Image Understanding
Prof. Tönnies
Sommersemester 2018

Group 4:
Benedikt Mayer
Lars Schnell
Alexander Wagner

Inhaltsverzeichnis

1. Einleitung

In diesem Projekt geht es um die Implementierung eines Algorithmus zur Klassifizierung von Bildern, beziehungsweise von Objekten auf Bildern, des CalTech101[1] Datensatzes. Der Datensatz besteht aus 101 verschiedenen Bildkategorien mit je 30-800 Bildern. Das Projekt umfasst die Auswahl von Bildfeatures, Methoden zur Reduzierung der Anzahl an Featuren, sowie einer Vorgehensweise zur Klassifizierung der Bilder.
Diese Ausarbeitung besteht aus 5 Kapiteln. In Kapitel 2, Konzept, wird die Organisation der Bilder für das Training und das Testen der implementierten Methode erläutert, die Auswahl der zu benutzenden Bildfeatures begründet, Möglichkeiten zur Reduzierung des Featurespace diskutiert und schlussendlich eine Methode zur Klassifizierung der Bilder (Featurevektoren der Bilder) vorgestellt. In Kapitel 3, Implementierung, wird die genaue Umsetzung des Konzeptes beschrieben. Dabei wird auf die verwendeten Tools eingegangen sowie die Parametrisierung der genutzten Algorithmen begründet. In Kapitel 4, Ergebnisse, werden die Resultate der implementierten Methode vorgestellt. Im Fazit, Kapitel 5, erfolgt ein Resümee dieses Projektes.

2. Konzept

Für die Umsetzung eines geeigneten Konzeptes zur Klassifizierung der Bilder des CalTech101 Datensatzes müssen verschiedene Punkte betrachtet werden. Die Bilder der verschiedenen Kategorien müssen so organisiert werden, dass sowohl für das Training, als auch das Testen eine repräsentative Verteilung der Bilder gewährleistet ist und eine Cross Validation (10-fold) durchgeführt werden kann. Dadurch soll ein mögliches Overfitting des Klassifikationsalgorithmus verhindert werden. Des Weiteren muss bei der Konzeption bedacht werden, dass der Datensatz für die verschiedenen Kategorien eine schwankende Anzahl an Beispielen aufweist. Außerdem muss über ein geeignetes Verfahren zum Umgang mit der Kategorie “Background Google” nachgedacht werden, da diese Kategorie willkürliche Bilder enthält. Ein weiteres Problem ist, dass es zum Teil sehr starke Ähnlichkeiten zwischen den verschiedenen Kategorien, zum Beispiel “Flamingo” und “Flamingo Head” gibt. Diese Probleme müssen bei der Auswahl der zu nutzenden Features sowie des Klassifikationsalgorithmus beachtet werden.

2.1. Bilddaten/Cross Validation

Die Ordnerstruktur des Datensatzes wurde direkt übernommen. Die einzige Vorverarbeitung, die im Bezug auf die Bilddaten stattgefunden hat, war eine Streckung der Bilder auf 200 x 200 Pixel, um eine einheitliche Grundlage für den zur Merkmalsgewinnung verwendeten HoG-Algorithmus nutzen zu können.
Entgegen unseres ersten Ansatzes, die Bilder den jeweiligen Folds entsprechend in Unterordnern zu speichern, haben wir uns im Verlauf des Projektes dazu entschieden, auf die split_train_test Funktion des Python Packages sci-kit-learn zurückzugreifen. Diese

[1] http://www.vision.caltech.edu/Image_Datasets/Caltech101/

ermöglicht, zu einer bestehenden Matrix, deren Zeilen die Feature Vektoren der einzelnen Bilder repräsentieren, entsprechend einer beliebigen Fold-Struktur in 10 Folds zu unterteilen, um diese für die Cross-Validation zu nutzen. Dieses Vorgehen war in unserem Fall angebracht, da wir dadurch zunächst insgesamt die Feature Vektoren zu allen Bildern (welche unabhängig von der Cross-Validation sind) mit Hilfe des HoG-Algorithmus berechnen konnten und im Anschluss PCA und SVM (welche von der Wahl der Folds abhängen) für die einzelnen Fold-Kombinationen durchführen konnten.

2.2. Feature Selection

Nach ausführlicher Betrachtung der Daten wurden zwei mögliche Verfahren zur Merkmalsgewinnung ausgewählt, welche möglicherweise gute Ergebnisse für die Bildklassifizierung liefern könnten.

2.2.1. Histogram of Oriented Gradients

Der HoG-Algorithmus zur Feature-Extraktion wurde aus den folgenden Gründen gewählt:

1. HoG arbeitet mit Kanten als Features. Diese schienen uns im Fall der vorliegenden Bilder als sehr aussagekräftig, im Gegensatz zu z.B. Farben, welche für unterschiedliche Darstellungen von Objekten derselben Klasse stark variieren.
2. Da die zu klassifizierenden Objekte jeweils zentriert und bis annähernd zur vollen Größe auf den Bildern dargestellt wurden, bot sich der HoG an. So ist dieser, im Gegensatz zu anderen Verfahren, nicht skalierungs- und translationsinvariant, sodass gut zwischen Klassen wie Flamingo und Flamingo Head unterschieden werden können sollte.
3. In der Vorlesung wurde darauf verwiesen, dass vor dem Aufkommen von CNNs der HoG Algorithmus eine sehr beliebte Technik zur Bildklassifizierung gewesen sei.

2.2.2. ORB (Orientated Fast and Rotated Brief)

Der ORB-Algorithmus zur Feature-Extraktion wurde aus den folgenden Gründen gewählt:

1. Unabhängig von Größe der dargestellten Objekte
2. Unabhängig der Rotation der dargestellten Objekte
3. Verwendung in Literatur für Objektklassifizierung

Es stellte sich allerdings heraus, dass diese Technik nicht sonderlich gut funktioniert. Die Anzahl der ORB Features und dementsprechend die Dimension des Feature Space unterscheiden sich zum Teil stark von Bild zu Bild und aufgrund der verschiedenen Feature Dimensionen zwischen den unterschiedlichen Kategorien war es für uns nicht möglich, mit Hilfe einer SVM eine Trennung der Klassen zu finden.

2.3. Feature Reduktion

Bei der Anwendung der beiden Feature Selektionsverfahren, beschrieben in Kapitel 2.2, werden sehr hochdimensionale Featurevektoren für jedes Bild berechnet. Eine Trennung der Klassen für solch hoch dimensionale Vektoren zu finden, kann sehr viel Rechenleistung benötigen beziehungsweise auch teilweise nicht sinnvoll möglich sein. Des Weiteren hilft die

Feature Reduktion einem potentiellen Overfitting entgegen zu wirken. Zur Reduzierung der Dimensionalität der Featurevektoren wird eine PCA (Principal Component Analysis) angewandt. Diese wurde in der Vorlesung als gängiges Verfahren zur Feature Reduktion vorgestellt und auch häufig bei der Bildklassifizierung verwendet.

2.4. Klassifikation

Für die Klassifizierung der Bilder wurde eine Support-Vector-Machine verwendet. Der Grund für diese Wahl ist, dass die Technik, besonders in Zeiten vor dem Aufkommen von neuronalen Netzen, sehr beliebt war und wir sie bereits in einer anderen Vorlesung (Intelligente Datenanalyse) als gute Technik kennengelernt haben. Außerdem war ein nicht-generatives Modell für die gestellte Aufgabe ausreichend.

2.5. Neuronales Netz

Für die Verwendung eines Neuronalen Netzes wurde Transfer Learning[2] verwendet. Dafür wurde das VGG16 Netz mit Gewichten, welche auf dem ImageNet Datensatz trainiert wurden, verwendet. Für die Bildklassifizierung wurde die Ausgabe des zweiten Fully-Connected Layer des Netzes extrahiert und der resultierende Feature Vektor der Länge 4096 für das Lernen einer SVM genutzt. In diesem Fall wurde keine zusätzliche Dimensionsreduktion angewandt. Das Verfahren erreichte eine durchschnittliche Accuracy von 90%.

3. Implementierung

Für die Umsetzung des Projektes wurde auf verschiedene Python-Bibliotheken zurückgegriffen, in denen die grundlegenden Funktionen zur Feature Extraktion und zur Klassifizierung bereits implementiert sind.

3.1. Tools

Für die Umsetzung des Projektes wird das folgende Setup genutzt:

- Jupyter 5.4.1 mit Python 3.6 innerhalb des Anaconda Navigators 1.8.2[3]
- Scikit Learn 0.19.1[4]
- OpenCV 3.3.1[5]
- Numpy 1.12.1
- Keras 2.1.5

---

[2] (orientiert an Paper A. S. Razavian, H. Azizpour, J. Sullivan and S. Carlsson, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition," *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Columbus, OH, 2014, pp. 512-519.)

[3] https://www.anaconda.com/download/

[4] http://scikit-learn.org/stable/

[5] https://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.3.1/opencv-3.3.1-vc14.exe/download

3.2. Konfiguration

Für die Implementierung der gesamten Bildverarbeitungspipeline mussten die Parameter für die verschiedenen Feature Selektions Verfahren, die Anzahl an Komponenten der PCA, sowie die Parameter für die genutzte SVM bestimmt werden.

HOG-Parameter:
Es ist anzumerken, dass die von uns getroffene Parameterwahl von den ursprünglich vorgeschlagenen Werten abweicht, da unsere Bilder zum einen größer sind und zum anderen versucht wurde, den zeitlichen Aufwand für die Berechnung der Feature Vektoren angemessen zu halten. Deshalb wurde ein Kompromiss zwischen Qualität und Berechnungsaufwand eingegangen. Es wurden nacheinander in einer For-Schleife verschiedene Parameterkombinationen getestet und im Anschluss die davon als am angemessensten befundene ausgewählt.
Die von uns gewählten Parameter lauten:

- Cell size: 25x25 (Größe der einzelnen Fensterzellen in Pixel)
- Cells per block: 3x3 (Größe der Normalisierungsblöcke)
- Normalization Norm: L1 (Norm bei der blockweisen Normalisierung)
- Number of orientations: 9 (Wie viele Orientierungs-Bins für die Gradientenrichtungen verwendet werden)
- Die sich daraus ergebende Länge der Feature Vektoren beträgt 2916.

Um die Bilder alle auf die gleiche Größe zu bringen, was für die Anwendung des HoG nötig war, haben wir alle Bilder auf 200x200 Pixel gestreckt. Die Befürchtung, dass etwaig entstehende Deformationen zu starken Einbußen in der Klassifizierungsgüte führen, hat sich nicht bestätigt.

PCA-Parameter:
Zur Ermittlung einer geeigneten Anzahl an Principal Components haben wir den kompletten Klassifizierungsablauf mehrfach mit einer Komponentenzahl von 5, 6, 7, …, 30 durchlaufen lassen. Die sich ergebende Accuracy verbesserte sich mit steigender Komponentenzahl kontinuierlich, sodass 30 als endgültiger Parameterwert festgelegt wurde. Interessanterweise war dieses Verhalten nicht zu erkennen, als wir für die im nächsten Abschnitt vorgestellte SVM noch einen RBF- anstatt eines linearen Kernels verwendet haben. Hier ergab sich als beste Komponentenanzahl für die PCA der Wert 14, wobei alle ganzzahligen Werte zwischen 5 und 20 getestet wurden.

SVM-Parameter:
Wie eben beschrieben, hatten wir zunächst einen RBF-Kernel für die SVM verwendet, dann jedoch festgestellt, dass ein linearer Kernel sowohl bessere Ergebnisse liefert, als auch eine kürzere Rechenzeit benötigt. Um mit den ungleichmäßigen Klassengrößen umzugehen, haben wir zunächst pro Klasse eine Balancierung aufgrund der Klassengröße vorgenommen. Anschließend entschieden wir uns jedoch für eine komplexere Variante, in

der sich eine starke Verstreutheit einer einzelnen Klasse über den Feature Space negativ auf ihr Klassengewicht auswirkt. Die Formel für die Klassengewichte lautet wie folgt:

$$w_i = 1 - \frac{c}{\max_j \sqrt[n]{|\Sigma_j|}} \sqrt[n]{|\Sigma_i|}$$

- w_i: *Klassengewicht*
- Σ_i: *Kovarianzmatrix der Klasse*
- n : *Größe des Feature Space (30)*
- c : *Grenzwert für das niedrigste mögliche Klassengewicht (0,5)*

Die Klasse mit dem höchsten Class-Scatter Wert und damit dem geringsten Gewicht (0,5) war wie erwartet BACKGROUND_Google.

4. Resultate

Bei der Anwendung der HOG Features in Kombination mit der PCA und dem SVM Klassifizierung Algorithmus, ergibt sich unter den oben genannten Bedingungen eine durschnittliche Accuracy von 64,4% über alle Folds mit einem Maximum von 67,7% und einem Minimum von 56,2% in dem Durchlauf, in welchem der größte Fold (Restfold) zum Testen verwendet wird. Die genauen Accuracies für jeden Fold der Cross Validation sowie die durchschnittliche Standardabweichung für die Vorhersage der einzelnen Klassen befindet sich im Anhang. Der Transfer Learning Ansatz erreicht eine durchschnittliche Accuracy von 90%, wobei bei dieser Technik mit den auf ImageNet trainierten Gewichten Informationen verwendet wurden, die eigentlich nicht Teil der Projektdaten sind.

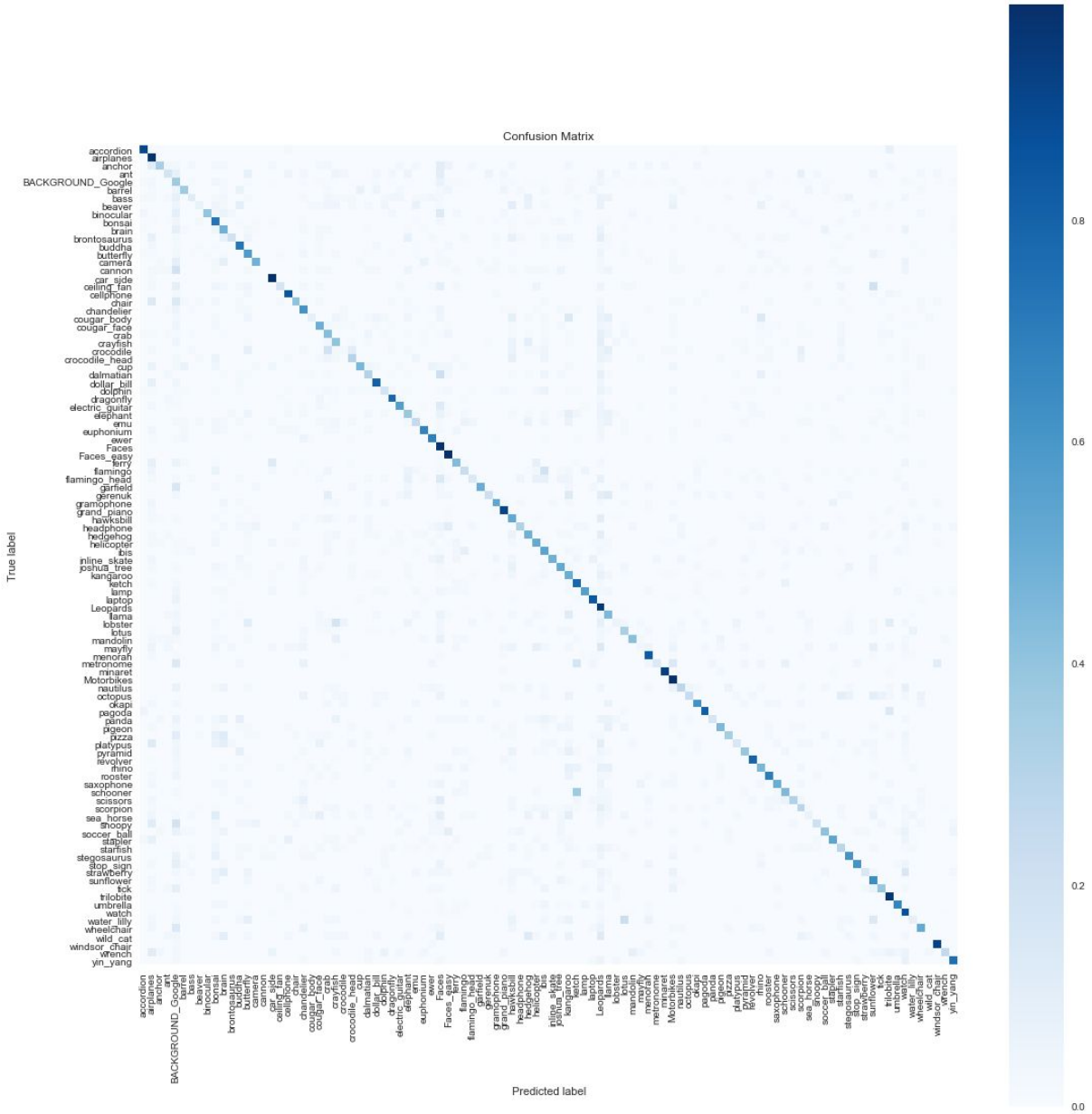
5. Fazit

Insgesamt haben wir bei der Bearbeitung des Projektes viel gelernt. Zum einen war es sehr interessant die theoretischen Grundlagen praktisch umzusetzen und zum anderen aber auch ein großer Lernfortschritt, da wir erst durch die tatsächliche Implementierung endgültig verstanden haben, wie die einzelnen Algorithmen eigentlich funktionieren (“Was ist mein Input? Was ist mein Output? Und wie bekomme ich daraus einen Feature Vektor?”)
Des Weiteren haben wir bei der Umsetzung des Projektes auch gemerkt, dass man sich im Vorfeld im Klaren darüber sein sollte, wie die zu verwendeten Techniken im Detail funktionieren, um frühzeitig abschätzen zu können, ob sie überhaupt funktionieren werden (zum Beispiel ORB-Features). Insgesamt sind wir der Meinung, dass unsere Ansätze eine akzeptable Vorhersagegenauigkeit haben. Die HoG-Feature in Kombination mit PCA und SVM liefern eine Genauigkeit von 64,4%. Während das neuronale Netz sogar eine Accuracy von 90% hat. Um eine ähnlich hohe Accuracy mit “herkömmlichen Verfahren” (der Bildverarbeitung) zu erreichen, müsste ein sehr großer Aufwand, in die Optimierung dieser Algorithmen für diesen Datensatz, investiert werden.

Eine mögliche Verbesserung unserer Lösung könnte eine Verbindung verschiedener Verfahren für die Bildklassifizierung sein (Boosting).

Anhang

Confusion Matrix



Class Accuracy und Standard Deviation Tabelle

	acc_1	**acc_2**	**acc_3**	**acc_4**	**acc_5**	**acc_6**	**acc_7**	**acc_8**	**acc_9**	**acc_10**	**std**
accordion	0,9	0,8	0,8	1	0,8	0,8	0,8	1	1	1	0,09
airplanes	1	1	1	1	0,95	0,98	0,94	1	0,99	0,9	0,03
anchor	0,17	0,25	0	0	0,25	0,5	0,75	0,75	0,5	0,25	0,26
ant	0,5	0,25	0	0	0	0,25	0	0,25	0	0,25	0,17
BACKGROUND_Google	0,34	0,35	0,35	0,3	0,39	0,39	0,33	0,3	0,46	0,41	0,05
barrel	0,45	0,5	0	0,25	0,25	0,5	0,25	0,25	0,25	0,75	0,2
bass	0,11	0,4	0	0,2	0,2	0	0	0	0	0,2	0,13
beaver	0	0	0	0	0,5	0	0	0	0	0,25	0,16
binocular	0	0	0	0	0	0	0	0,33	0,33	0	0,13
bonsai	0,85	0,83	0,75	0,75	0,67	0,75	0,67	0,58	0,75	0,67	0,08
brain	0,24	0,67	0,44	0,56	0,44	0,44	0,44	0,56	0,78	0,44	0,14
brontosaurus	0,14	0	0,25	0,25	0,25	0,25	0,25	0,5	0	0	0,15
buddha	0,69	0,38	0,63	0,63	0,88	1	0,63	0,75	0,88	0,75	0,17
butterfly	0,9	0,33	0,67	0,56	0,56	0,33	0,78	0,22	0,56	0,78	0,21
camera	0,4	0,8	0,4	0,2	0,6	0,4	0,6	0,6	0,6	0,2	0,18
cannon	0	0	0	0	0	0	0	0	0	0	0
car_side	1	1	0,92	1	1	1	1	1	1	1	0,03
ceiling_fan	0,18	0,25	0,25	0,25	0	0,25	0,25	0	0	0,25	0,11
cellphone	0,71	1	1	1	1	0,6	1	0,8	1	0,6	0,17
chair	0,25	0,17	0,5	0,5	0,17	0,67	0,67	0,33	0,67	0,17	0,21
chandelier	0,71	0,6	0,6	0,8	0,4	0,6	0,6	0,4	0,6	0,7	0,12
cougar_body	0,09	0,25	0	0,25	0	0	0	0	0	0,25	0,11
cougar_face	0,53	0,5	0,33	0,5	0,83	0,33	0,67	0,67	0,17	0,33	0,19

crab	0,3	0,14	0,57	0,71	0,57	0,71	0,43	0,29	0,57	0	0,23
crayfish	0,29	0,14	0,43	0,57	0,43	0,57	0,43	0,14	0,43	0,57	0,15
crocodile	0	0	0,2	0	0	0	0	0	0	0	0,06
crocodile_head	0	0,4	0,4	0,4	0,8	0,2	0,2	0,2	0	0,4	0,22
cup	0,5	0,2	0,4	0,4	0,4	0,8	0,2	0,8	0,6	0,2	0,22
dalmatian	0,15	0,5	0,33	0	0,5	0,17	0,33	0,33	0,33	0,5	0,16
dollar_bill	0,86	0,8	0,4	1	0,8	0,8	0,6	1	0,8	1	0,18
dolphin	0,18	0	0	0,17	0	0,33	0,17	0,5	0,17	0,17	0,15
dragonfly	0,71	0,83	0,83	1	0,67	0,83	0,5	0,67	0,83	1	0,15
electric_guitar	0,67	0,57	0,57	0,57	0,57	0,57	0,57	0,71	0,29	0,43	0,11
elephant	0,4	0,5	0,17	0,5	0,5	0,17	0,33	0,33	0,17	0,67	0,16
emu	0,25	0,4	0	0,2	0,4	0,2	0,4	0,2	0,4	0	0,15
euphonium	0,5	0,67	0,5	0,83	0,67	0,67	0,67	0,5	0,83	1	0,16
ewer	0,69	0,75	0,5	0,63	0,75	0,75	0,63	0,88	0,5	0,75	0,11
Faces	0,94	1	0,95	1	0,98	1	0,98	1	0,98	0,98	0,02
Faces_easy	0,96	0,98	0,98	1	1	1	1	0,98	1	1	0,01
ferry	0,46	0,5	0,83	0,67	0,5	0	0,33	0,5	0,33	0,17	0,23
flamingo	0,15	0,17	0,33	0,17	0,17	0,17	0,67	0,17	0,17	0,17	0,15
flamingo_head	0,33	0	0,25	0,5	0	0	0	0	0	0	0,18
garfield	0,86	1	0,67	0,33	0,33	0,33	0,33	0	0,33	0,33	0,28
gerenuk	0	0,33	0	0,33	0	0,33	0	0,33	0,33	0,33	0,16
gramophone	0,5	0,4	0,4	0,4	0,4	0,6	0,4	0,4	1	0,6	0,18
grand_piano	0,89	0,89	1	0,89	0,89	0,89	0,89	1	0,78	1	0,07
hawksbill	0,5	0,7	0,2	0,7	0,7	0,2	0,6	0,6	0,3	0,8	0,21
headphone	0,33	0,25	0,5	0,25	0,25	0,5	0,5	0,25	0,25	0	0,15
hedgehog	0,56	0,4	0,4	0,6	0,4	0,8	0,4	0,4	0,4	0,4	0,13
helicopter	0,56	0,5	0,5	0,75	0,25	0,5	0,75	0,38	0,63	0,25	0,17

ibis	0,63	0,5	0,13	0,75	0,25	0,25	0,63	0,88	0,75	0,63	0,24
inline_skate	0,75	0	0	0	0	1	1	1	1	0	0,48
joshua_tree	0,4	0,67	0,5	0,67	0,5	0,67	0,5	0,5	0,5	0,33	0,11
kangaroo	0,36	0,5	0,38	0,38	0,38	0,75	0,63	0,38	0,75	0,63	0,15
ketch	0,8	0,73	0,55	0,91	0,91	0,55	0,82	0,82	0,91	0,82	0,13
lamp	0,71	0,33	0,5	0,33	0,17	0,67	0,33	0,67	0,83	0,83	0,22
laptop	0,78	0,88	0,88	0,75	0,88	1	0,88	0,75	1	0,63	0,11
Leopards	0,95	0,9	1	1	1	0,9	0,85	1	1	1	0,05
llama	0,47	0,43	0,57	0,43	0,43	0,29	0,43	0,14	0,57	0,86	0,18
lobster	0	0	0	0	0	0,25	0	0	0,25	0	0,1
lotus	0,25	0,5	0,33	0,33	0,5	0,5	0,33	0,33	0,17	0,17	0,12
mandolin	0,57	0,25	0,5	0,5	0,75	0,25	0,5	0,25	0,25	0,25	0,17
mayfly	0	0	0	0	0	0	0	0	0,25	0	0,08
menorah	0,67	1	0,88	0,88	0,88	0,88	0,5	1	0,88	0,88	0,14
metronome	1	0,67	0	0,33	0	0	0	0	0	1	0,41
minaret	1	0,86	0,86	0,71	1	1	1	0,86	1	1	0,1
Motorbikes	1	1	0,95	1	1	1	1	1	1	1	0,02
nautilus	0,2	0,2	0,2	0,4	0,2	0,4	0,4	0,2	0,4	0,2	0,1
octopus	0,13	0	0	0	0,33	0	0	0	1	1	0,39
okapi	0	1	1	0,67	0,67	0,67	0,67	0	0,67	0,67	0,33
pagoda	0,73	0,75	0,75	0,75	1	1	0,75	0,75	1	0,75	0,12
panda	0,09	0,33	0,33	0	0,33	0	0	0	0,33	0,33	0,16
pigeon	0,44	0,5	0,25	0,75	0,75	0	0,25	0,25	0,5	0,75	0,24
pizza	0,25	0	0,2	0,4	0,4	0,6	0,4	0,2	0,6	0,4	0,18
platypus	0	0	0,33	0,33	0	0	0	0	0,67	0	0,22
pyramid	0,33	0,8	0,2	0,6	0,2	0,2	0	0,2	0,6	0,8	0,27
revolver	0,6	1	0,88	0,63	0,63	0,88	1	0,88	0,75	0,75	0,14

rhino	0,29	0,4	0,8	0,2	0,2	0,8	0,6	0,6	0,6	0,4	0,21
rooster	0,69	0,5	0,75	0,5	0,25	0,75	0,75	0,75	1	1	0,22
saxophone	0	0,25	0	1	1	0,75	0,75	0,5	0,5	0,25	0,35
schooner	0,44	0,67	0,5	0,5	0,17	0,83	0,5	0,33	0,33	0,17	0,2
scissors	0,42	0,67	0	0,33	0	0	0,33	0	0,33	0,67	0,25
scorpion	0,33	0,25	0,13	0,13	0,38	0,13	0,38	0,63	0,38	0,25	0,15
sea_horse	0,08	0	0,2	0,2	0	0	0	0	0	0	0,08
snoopy	0	0,67	0	0	0	0,33	1	0	0,33	0	0,33
soccer_ball	0,4	0,17	0,33	0,67	0,5	0,5	0,67	0,33	0,33	0,17	0,17
stapler	0,67	0,75	1	0,5	0	0,25	0,25	0,5	0,75	0,5	0,28
starfish	0,14	0,5	0,38	0,25	0,13	0,25	0,13	0,25	0,75	0,13	0,19
stegosaurus	0,43	0,4	0,6	0,6	0,8	0,8	1	0,8	0,4	0,6	0,19
stop_sign	0,7	0,67	0,5	0,33	0,83	0,83	0,33	0,33	0,83	0,67	0,2
strawberry	0	0	0	0,33	0	0	0	0,33	0	0,33	0,15
sunflower	0,62	0,63	0,75	0,88	0,63	0,38	0,63	0,5	0,88	0,5	0,15
tick	0,38	0,75	0,5	0,25	0,25	0,25	0,25	0,75	0,25	0,25	0,2
trilobite	1	1	1	0,88	1	0,88	1	1	0,88	1	0,06
umbrella	0,5	0,71	0,71	0,86	0,71	0,57	0,71	0,71	0,29	1	0,18
watch	0,63	0,91	1	0,96	1	0,78	0,87	0,96	0,91	0,74	0,12
water_lilly	0	0	0	0	0	0	0,33	0	0	0,33	0,13
wheelchair	0,43	0,8	0,6	0,2	0,6	0,4	0,8	0,2	0,4	0,8	0,22
wild_cat	0	0	0	0	0	0	0	0	0	0	0
windsor_chair	0,91	0,8	1	1	1	0,8	1	0,8	1	1	0,09
wrench	0	0,33	0,33	0,33	0	0,67	0,33	0,33	0,33	0,67	0,21
yin_yang	0,67	0,67	1	0,83	0,33	0,67	1	0,83	0,5	1	0,21