Universität Rostock

Traditio et Innovatio

Bachelorarbeit zum Thema

Analyse unterschiedlicher Ansätze für Content-based Image Retrieval (CBIR)

B.A.-Studiengang:	Informatik
Eingereicht an der:	Fakultät für Informatik der Universität Rostock
Vorgelegt von:	Artur Strelnikov
Matrikel-Nr.:	217203909
ErstgutachterIn:	Prof. Dr.-Ing. Thomas Kirste
ZweitgutachterIn:	Priv.-Doz. Dr.-Ing. habil. Dirk Joachim Lehmann
Betreuer:	Dr. rer. nat. Sebastian Bader
Lehrstuhl:	Mobile Multimedia Information Systems
Abgeabedatum:	02.06.2021

Bachelorarbeit

Fakultät für Informatik

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Content-based Image Retrieval (CBIR) bezeichnet die Herausforderung, eine Anfrage an eine Bilddatenbank mit Hilfe von nur einem Beispielbild zu stellen. Ein CBIR-System muss die Ähnlichkeit zwischen zwei beliebigen Bildern quantifizieren können, um eine Sortierung der Datenbank zu ermöglichen. Theoretisch ist das Qualitätsmaß dabei die subjektive menschliche Wahrnehmung von Ähnlichkeit. Die Aufgabe von dem System ist also, die semantische Lücke zwischen den Pixeln der Bilder und einer kompakten Repräsentation zu finden, in der eine Distanz (z.B. Euklidische Distanz bei Vektoren) dem menschlichen Verständnis von Ähnlichkeit entspricht.

CBIR wird für die Suche nach ähnlichen Bildern verwendet, wobei keine Annotation der Daten erforderlich ist. Dies kann zum Beispiel in medizinischen Anwendungen benutzt werden, um Ähnlichkeiten in Strukturen zu finden, ohne sich dabei auf die Diagnose eines Menschen verlassen zu müssen. CBIR kann auch mit der Organisation großer und vielfältiger Daten helfen, vor allem wenn die Metadaten nicht zu 100% korrekt und vollständig sind. Das bekannteste Beispiel dieser Anwendung ist „Google reverse image search“.

Das Ziel dieser Arbeit ist es, einen praktischen Vergleich der existierenden CBIR - Methoden aus der Literatur durchzuführen. Dafür werden die Tools zunächst in Gruppen eingeteilt (Kap. 3.3). Danach wird die Funktionsweise im Kap. 3.4 analysiert und die Ergebnisse mithilfe der Implementation reproduziert. Am Ende wird eine quantitative und qualitative Analyse der Implementationen (Kap. 4) auf 2 unterschiedlichen Datensätzen (ROxford5k und NUS-WIDE) durchgeführt, wobei vor allem auf die Unterschiede zwischen den Deskriptoren Wert gelegt wird. Bei dem Vergleich wird sowohl die Fähigkeit allgemeine Bilder (NUS-WIDE) zu unterscheiden, als auch die Optimierbarkeit für eine Spezielle Anwendung (Gebäudeerkennung mit ROxford5k) evaluiert.

Im Rahmen dieser Arbeit wurde auch ein Tool entwickelt, das ein gemeinsames grafisches Interface für die verwendeten Methoden bietet. Dieses wird in Kap. 5 vorgestellt.

2 Grundlagen

Während die genaue Funktionsweise der Tools einzeln behandelt wird, werden in diesem Kapitel zwei grundlegende Mittel erklärt, die später als Grundlagen verwendet werden.

Viele der in dieser Arbeit verwendeten Tools nutzen Convolutional Neural Networks (CNN), um Bilder zu beschreiben und am Ende als Vektor darzustellen. Während die genaue Umsetzung vom Tool abhängt, sind viele der grundlegenden Funktionen gleich. Ein CNN, auf deutsch „faltendes neuronales Netz“ besteht aus mehreren Convolutional Layers, in denen die Neuronen angeordnet sind. Jeder Neuron beschreibt dabei über einen Filterkernel (Faltungsmatrix) eine kleine Region seiner Eingabe. Die Ergebnisse von einem Neuron beeinflussen dabei benachbarte Ergebnisse, wodurch eine kleine Region beschrieben werden kann. Durch wiederholte Anwendung von unterschiedlichen Schichten mit unterschiedlicher Form und Werten können tiefe (deep) CNNs komplexe Bildstrukturen beschreiben.

Um die Ergebnisse von CNNs (und anderer Deskriptoren) besser für Distanzberechnung verwenden zu können, werden diese normalisiert. Die hierfür verwendete Methode ist die l_2 (l2) Normalisierung, bei der die Summe der Quadrate aller Werte des Vektors $(v_1, \ldots, v_n)$ 1 ergibt: $1 = \sum_{i=1}^{n} v_i^2$. Diese Normalisierung gleicht den Einfluss unterschiedlicher Features an, wobei Ausreißer beibehalten werden. Die l_2 Norm hat auch den Vorteil, dass sie eine Analytische Lösung hat und dadurch leicht berechenbar ist.

3 State of the art

Content-based Image Retrieval wird seit den frühen 1990er Jahren erforscht. In der Analyse von Zheng et al. [26] wird dieser Zeitraum grob in drei unterschiedliche Phasen eingeteilt. Wie man in Abb. 1 sehen kann, war die Forschung vor 2003 darauf fokussiert Bilder mit globalen Feature-Vektoren zu beschreiben. Von 2003 bis 2012 wurde primär SIFT (Scale Invariant Feature Transform) und ähnliche Algorithmen verwendet, um lokale Features in den Bildern zu finden und zu beschreiben. Diese Features werden danach so organisiert, dass sie eine kompakte und sinnvolle Repräsentation der Bilder darstellen. Die letzte in Abb. 1 dargestellte Phase sind die Convolutional Neural Network (CNN) - basierten Verfahren. Diese wurden durch die Publikation von „ImageNet classification with deep convolutional neural networks“ [8] in 2012 ermöglicht. Diese Verfahren versuchen mithilfe von CNNs eine Repräsentation von Bildern zu lernen.

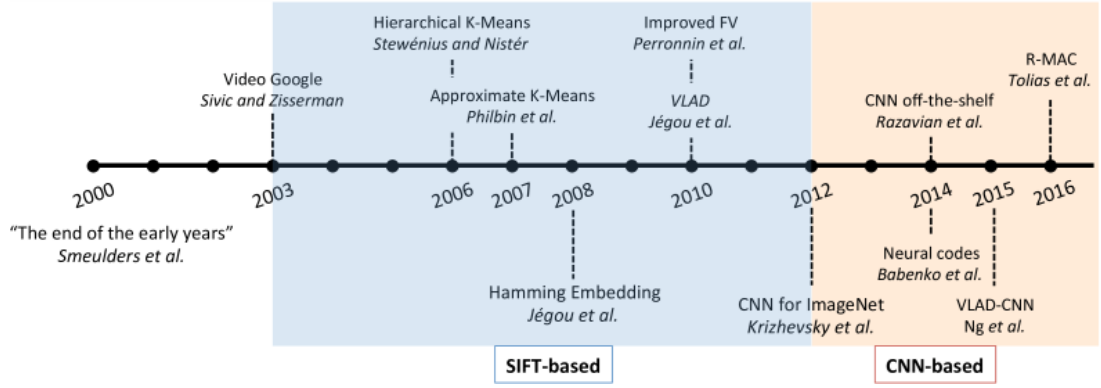


Abbildung 1: Zeitstrahl mit wichtigen Beiträgen für CBIR Forschung [26]

3.1 Literaturanalyse

Bei der Recherche für diese Arbeit lag der Fokus darauf, die besten reproduzierbaren CBIR Ansätze zu finden. Um die Reproduzierbarkeit in einer sinnvollen Zeit zu ermöglichen ist wichtig gewesen, dass Code für die untersuchten Paper veröffentlicht wurde. Ein weiteres notwendiges Kriterium bei der Suche nach relevanten Papers war, dass die Autoren eine quantitative Auswertung auf einem öffentlich verfügbaren Datensatz durchgeführt haben. Diese ermöglicht einen objektiven (nicht vollständigen) Vergleich, der eine Sortierung und somit eine Auswahl von den „besten“ Methoden ermöglicht. Die letzten Einschränkungen bei der Suche waren, dass die Publikation sich mit reinem CBIR ohne zusätzliche Informationen (z.B. Tags) beschäftigt und kein interdisziplinäres Wissen (z.B. Medizin, Biologie) für das Verständnis der Bilder benötigt. Die durch diese Suche ausgewählten Tools werden in den Kapiteln 3.4.2 bis 3.4.8 noch genauer vorgestellt.

3.2 kommerzielle Lösungen

In der Geschichte von Content-based Image Retrieval wurden viele unterschiedliche kommerzielle Lösungen für dieses Problem entwickelt. Eines der ersten Tools war „QBIC“ (Query By Image Content), das von IBM in 1995 entwickelt wurde. Heutzutage ist CBIR eher dadurch bekannt, dass jede große Suchmaschine (Google, Bing usw.) ein Feature hat, dass eine Anfrage mit einem Beispielbild erlaubt. Offensichtlich steckt dahinter eine leistungsstarke CBIR Implementation, die mit sehr vielen unterschiedlichen Bildern umgehen kann. Diese Tools bieten dem Nutzer aber nur die Möglichkeit, Bilder im Internet zu suchen, weshalb sie für eine interne Anwendung nicht nützlich sind.

Außer den Suchmaschinen existieren noch viele anwendungsspezifische Umsetzungen, die Content-based Image Retrieval für eine bestimmte Anwendung, wie z.B. Medizin oder Textilindustrie, optimiert haben. Diese sind aufgrund ihrer Natur nicht brauchbar, wenn man nicht dieselbe Art von Bildern hat, wie von den Entwicklern vorgesehen wurde.

Die letzte Gruppe an CBIR Entwicklungen sind Erweiterungen für bestimmte Datenbankmanagementsysteme (DBMS), wie Elasticsearch. Diese verbinden einige der existir die Datensicherheit und Kompatibilität, fügt aber auch zusätzliche Datenbank-Komponenten hinzu, die Rechenzeit und Komplexität erhöhen ohne einen Vorteil für die untersuchten CBIR-Methoden bieten, weshalb sie für diese Arbeit irrelevant sind

Deshalb wird sich in dieser Arbeit nur auf den Vergleich der wissenschaftlichen Literatur konzentriert.

3.3 Gruppierung

Nach dem Vorbild von Zheng et al. [26] wurden die in dieser Arbeit untersuchten Publikationen in einzelne Gruppen eingeteilt. Die Einteilung hierfür wurde anhand des verwendeten Deskriptors getroffen. Hierbei wurden 2 Kriterien definiert, wovon jedes 2 mögliche Antworten hat. Das erste Kriterium unterscheidet daran, ob die Bilder mit einem Deep-Learning Deskriptor oder einem hand-crafted Deskriptor beschrieben werden. Das zweite Unterscheidungskriterium befasst sich mit der Art der Features und trennt die Methoden in lokal und global. Ein lokales Feature beschreibt im Unterschied zu einem globalen nicht das Bild, sondern nur eine interessante, kleine

Region. Ein Bild kann mit vielen lokalen Features beschrieben werden, welche in einem späteren Schritt zusammengefasst werden können.

Mit den beiden vorgestellten Unterscheidungen kommt man auf vier unterschiedliche Gruppen. In jeder dieser Gruppen wurden zwei Tools ausgesucht, reproduziert und miteinander verglichen. Bei der Auswahl wurden Tools mit einem hohen mean average precision (mAP) Wert (genauer erklärt in Kap. 4.1) auf Oxford5k [12] (und ROxford5k [13]) bevorzugt. Dieser Datensatz wurde in der Literatur am häufigsten verwendet und die darauf erreichten mAP-Werte stellen ein einfaches Vergleichskriterium dar, das bessere Tools bei der Auswahl bevorzugt.

3.3.1 Globale Hand-crafted CBIR Methoden

Die Kategorie globaler Hand-crafted Methoden ist die älteste und einfachste der untersuchten Gruppen. Die Methoden dieser Gruppe verwenden zum Beispiel Farb-Histogramme und Farbmomente, die ein komplettes Bild algorithmisch beschreiben können. Diese Methoden wurden hauptsächlich vor 2003 (Zeitangabe von Zheng et al. [26]) erforscht, weshalb im Rahmen dieser Arbeit nur das Paper „Evaluating Color Descriptors for Object and Scene Recognition“ ([22]) nach den vorgestellten Suchkriterien gefunden wurde. Bei genauerer Analyse der Implementation, hatte sich herausgestellt, dass die Autoren nur ein kompiliertes Programm zur Verfügung gestellt haben. Dieses Programm bietet die Möglichkeit, für ein Bild lokale oder globale Features zu beschreiben. Die Implementation bietet aber keine Möglichkeit, eine Datenbank zu erstellen und Anfragen zu stellen und somit ist auch die Verwendung dieser Implementation für Content-based Image Retrieval nicht möglich. Aus diesen Gründen wird dieses Paper als nicht reproduzierbar betrachtet und es können keine Beispiele aus der Kategorie „Global Hand-crafted CBIR“ für den Vergleich verwendet werden.

3.3.2 Lokale Hand-crafted CBIR Methoden

Die Gruppe der lokalen Hand-crafted Methoden verwendet existierende Algorithmen um „Keypoints“ (wichtige Punkte) im Bild zu finden und mit einem Vektor zu beschreiben. In dieser Arbeit werden die Deskriptoren „Scale-Invariant Feature Transform (SIFT)“ [9] und „Oriented FAST and rotated BRIEF (ORB)“ [16] verwendet. Die Herausforderung von den CBIR-Systemen in dieser Gruppe ist es, die lokalen Vektoren zu einer kompakten Repräsentation zusammenzufassen. Das erste Modell, das dies erfolgreich geschafft hat, ist „Bag of Visual Words (BOVW)“ [17], welches in dieser Arbeit vorgestellt aber nicht praktisch untersucht wird. Die ausgewählten Tools heißen „Vector of local aggregated descriptors (VLAD)“ [7] (Kap. 3.4.2) und „Aggregated Selective Match Kernels (ASMK)“ [20] (Kap. 3.4.3) und werden auch noch genauer untersucht.

3.3.3 Globale Deep-Learning CBIR Methoden

In dieser Gruppe werden Tools betrachtet, die direkt eine Bildrepräsentation mithilfe von CNNs generieren. Das verwendete Neuronale Netz kann entweder nur vortrainiert (idr. auf dem ImageNet Datensatz) oder speziell für die Anwendung getunt worden sein. Für das Finetuning der Netze wird in der Regel entweder der Datensatz selbst oder ein ähnlicher Datensatz verwendet. Bei dem Anwendungsbeispiel von Gebäuden sind zum Beispiel der Google Landmarks Datensatz [1], [4] und der SfM-120k [14] Datensatz verfügbar. Bei diesen Datensätzen handelt es sich um

sehr große annotierte Bildersammlungen, die man für Training und Fine-tuning von CNN Deskriptoren verwenden kann, um vor allem die Erkennung von ähnlichen Gebäuden zu trainieren.

Für das Training wird hier ein Tool mit dem Namen „Deep Metric Learning (DML)“ (Kap. 3.4.4) verwendet. Es werden 2 mögliche loss-Funktionen genauer beschrieben und die beste in der Evaluation verwendet. Zusätzlich wird eine zweite Implementation in den Vergleich hinzugefügt (Kap. 3.4.5), die mit höherer Auflösung arbeitet und für Gebäude vortrainierte Modelle beinhaltet. Es wird außerdem versucht, diese Bildbeschreibungen in einem Diffusions - Graphen (Kap. 3.4.6) zu speichern, um möglicherweise bessere Ergebnisse zu erzielen.

3.3.4 Lokale Deep-Learning CBIR Methoden

Die Verwendung von lokalen Features, welche mit Deep Learning Methoden gelernt, gefunden und beschrieben werden, ist die neueste der hier untersuchten Klassen und hat dadurch eine relativ geringe Menge an Implementationen. Die erste Publikation mit diesem Ansatz war das Paper „Large-Scale Image Retrieval with Attentive Deep Local Features“ [11], welches mithilfe eines Neuronalen Netzes lokale Features aus den Bildern extrahiert hat. Die Features werden danach durch ein weiteres Neuronales Netz gefiltert. Die Anwendung dieser Features wird in dem weiterführenden Paper „Detect-to-Retrieve: Efficient Regional Aggregation for Image Search“ [19] beschrieben. In dem Paper wird unter anderem die Verwendung der existierenden Methoden zum Umgang mit lokalen Features vorgeschlagen (genauer ASMK).

Das zweite in dieser Gruppe untersuchte Tool wurde von den Autoren von ASMK vorgestellt und kombiniert das ASMK-Verfahren mit einem Deep-Learning Deskriptor, der „HOW“ genannt wird.

3.4 Funktionsweise der untersuchten Tools

Im folgenden Kapitel wird die Funktionsweise von jedem Tool genauer untersucht, um besser zu verstehen, wie sie funktionieren und um einige der Ergebnisse besser erklären zu können.

3.4.1 Bag of Visual Words (BOVW), (lokal Hand-crafted)

Das BOVW Modell [17] wird zwar nicht direkt im Rahmen dieser Arbeit betrachtet, aber es wird oft als Grundlage für die anderen lokalen Methoden verwendet. Der Grund dafür ist, dass BOVW das erste Modell war, das viele lokale Features zu einer kompakten (und Sinnvollen) Bildrepräsentation für CBIR zusammenzufassen konnte.

BOVW basiert auf dem „Bag of Words“ Modell aus dem Bereich der Textsuche. Die Ähnlichkeit zwischen 2 Texten wird hierbei über die Häufigkeit der vorkommenden Worte definiert. Um diese Idee in der Bildverarbeitung mit lokalen Deskriptoren umzusetzen, muss zunächst ein Codebook mit K Schwerpunkten („visuellen Wörtern“) erstellt werden. Dieses wird mithilfe eines clustering-Verfahrens aus einer Vielzahl lokaler Deskriptoren erstellt. Der clustering Algorithmus fasst dabei die vielen n-dimensionalen Vektoren von den einzelnen Features zu einer vordefinierten Anzahl von Gruppen zusammen, die im Falle von Codebooks durch einen n-dimensionalen Mittelpunkt (Cluster Zentrum) repräsentiert werden.

Das Codebook wird danach verwendet, um die Lokalen Deskriptoren zu einem der visuellen Wörter zu quantisieren. Die Beschreibung eines Bildes im BOVW Modell besteht somit aus

einem Histogramm mit den Häufigkeiten jedes visuellen Wortes, wie auch in Abb. 2 dargestellt wird.

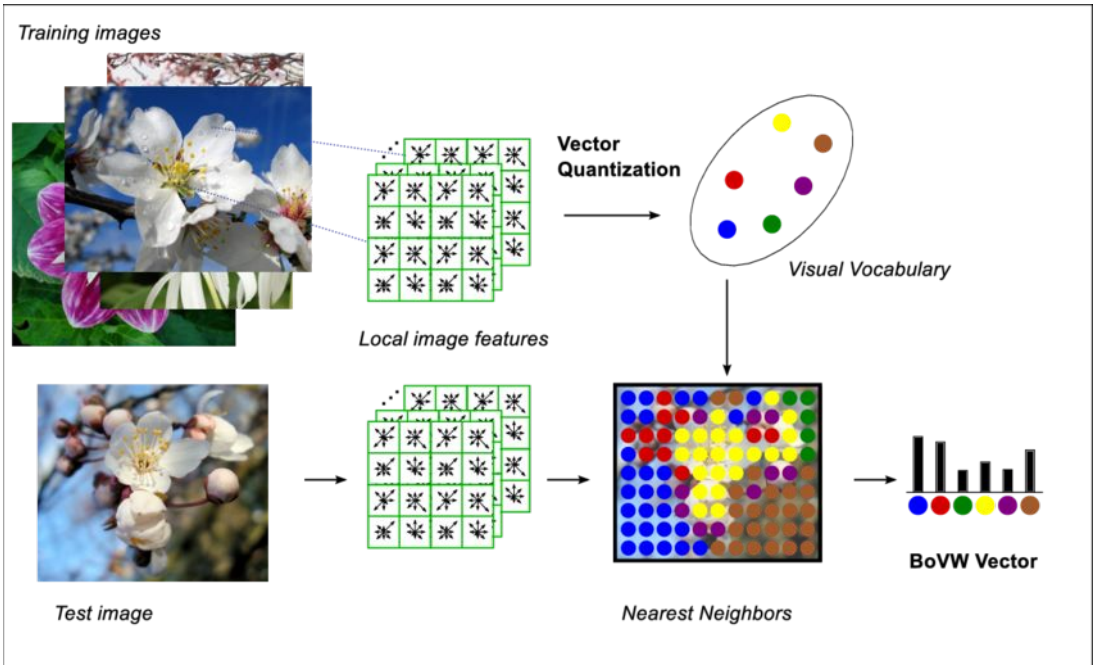


Abbildung 2: Erstellung eines BOVW Vektors mithilfe eines Codebooks (Visual Vocabulary).

3.4.2 Vector of local aggregated descriptors (VLAD), (lokal Handcrafted)

Ein Schritt zur Verbesserung der Qualität der BOVW – Bildbeschreibungen ist die Verwendung von einem Fischer Vektor, um nicht nur die visuellen Wörter, sondern auch die beobachtete Abweichung zu den Worten zu speichern. Dieser Vektor stellt zwar die theoretische Grundlage dar, ist aber sehr zeitaufwändig zu berechnen, weshalb in dieser Arbeit die weiter verbreitete nicht-probabilistische Approximation vom Fischer Vektor betrachtet wird: der Vector of local aggregated descriptors (VLAD) [7].

Bei VLAD wird, genauso wie bei BOVW ein Codebook $\{\mu_1 \dots \mu_K\}$ erstellt und jeder d-dimensionale lokale Deskriptor x_t wird dem nächsten visuellen Wort $NN(x_t)$ zugewiesen. Für jedes Codewort wird die Differenz $x_t - \mu_i$ zwischen dem Codewort und allen zugewiesenen Deskriptoren x_t berechnet und akkumuliert:

$$v_i = \sum_{x_t : NN(x_t)=i} x_t - \mu_i$$

Die VLAD Beschreibung für ein Bild ergibt sich dabei durch Konkatenation der d-dimensionalen Vektoren v_i.

Durch diese Zusammenfassung erreicht VLAD eine höhere Genauigkeit mit einem kleineren Codebook als BOVW. Die Größe der verwendeten Codebooks in der Publikation variiert dabei zwischen 16 und 256 Wörtern, im Gegensatz zu Millionen bei BOVW.

3.4.3 Aggregated Selective Match Kernels (ASMK), (lokal Hand-crafted)

Die nächste Weiterentwicklung bei der Zusammenfassung lokaler Features für CBIR ist die Publikation von „Aggregated selective match Kernel“ (ASMK) [20]. In diesem Modell wird VLAD [7] mit BOVW so kombiniert, dass ein Größeres Codebook verwendet werden kann. Bei dem ASMK Modell wird das Bild zunächst in Zellen eingeteilt, und jede Zelle wird wie bei VLAD aggregiert und l_2-normalisiert. Bei der Zusammenfassung wird, ähnlich wie bei BOVW die Spärlichkeit der visuellen Worte ausgenutzt und dadurch kann ASMK mit größeren Codebooks zurechtkommen. Dies liegt vor allem daran, dass ASMK nur die visuellen Wörter speichert, die auch für die jeweilige Region interessant sind (und nicht alle, wie bei VLAD). Die Autoren von ASMK [20] verwendeten hauptsächlich eine Codebook-Größe von 65536 Wörtern, was in [26] als mittelgroßes Codebook eingestuft wird (im Gegensatz zu kleinen Codebooks bei VLAD).

Eine Besonderheit von ASMK ist die Verwendung einer eigenen Selektivitäts–Funktion, die die Wichtung der Zellen anpassen kann. Die Verwendete Form von ASMK geht auch noch einen Schritt weiter und bildet die zusammengefassten Deskriptoren als binäre Codes ab. Dies ermöglicht eine viel kompakteren Darstellung der Bilder.

3.4.4 Deep Metric Learning (DML), (global Deep-Learning)

In dem Bereich von Deep Metric Learning wird versucht, ein Neuronales Netz so feinzutunen, dass es die wichtigen Eigenschaften der Bilder erlernt (im Bezug auf die Anwendung) und dadurch eine sinnvolle Darstellung finden kann. Die Bild–Repräsentation ist ein Vektor im euklidischen Raum, der in der verwendeten Implementation eine Länge von 128 Werten hat. Das Ziel von Deep Metric Learning ist es, dass die Vektoren von ähnlichen Bildern nah beieinander liegen, sodass bei einer CBIR Anfrage eine KNN - Suche (k-nearest neighbors) die Ergebnisse liefern kann.

Die in dieser Arbeit untersuchten Deep Metric Learning Methoden verwenden als Grundlage ResNet50 [5], eine Form von dem CNN ResNet mit 50 Schichten. Die letzte softmax pooling Schicht wird dabei durch eine fully connected Schicht ersetzt, um den angestrebten Vektor, statt einer Klassenzuordnung zu bekommen.

Der interessanteste Aspekt von DML ist das Training Nach der Definition von dem Ziel des Trainings und einer Betrachtung der Architektur wird hier das Training von DML untersucht. Um das besser zu verstehen, werden zwei verwendete loss-Funktionen betrachtet. Die Implementation unterstützt die Verwendung von Triplet ([23], [6]), Margin ([24]), ProxyNCA ([10]), N-Pair ([18]) und FastAP ([15]) loss. Besonders interessant davon sind der Triplet loss, weil er die älteste und einflussreichste der verfügbaren Funktionen ist und dadurch viele Nachfolger auf dem Triplet Loss aufbauen oder sich mit ihm vergleichen. Bei Experimenten im Rahmen der Bachelorarbeit wurden alle aufgelisteten loss-Funktionen ausprobiert und dabei hat der ProxyNCA loss [10] die besten Ergebnisse geliefert, weshalb diese Funktion ebenfalls ein besonderes Interesse darstellt.

Beide untersuchten loss-Funktionen gehen davon aus, dass die Trainingsbilder nur mit einfachen Labels annotiert sind und somit wird Ähnlichkeit als „Bilder mit einem gemeinsamen Label“ definiert. Durch diese Annahme können Trainingsdatensatze relativ einfach erstellt werden, obwohl spezielle CBIR-Annotationen potentiell genauere Ergebnisse liefern könnten (was

zum Zeitpunkt dieser Arbeit nie versucht wurde, aufgrund der benötigten Datensatzgröße, der Komplexität der Annotationen und der Subjektivität von Ähnlichkeit).

Die Triplet loss-Funktion ([23], [6]) verwendet eine Gruppe aus 3 Bildern: $t = (x, x^+, x^-)$. Dieses Triplet besteht aus einem Referenzbild x, einem positiven (mit gleichem Label wie Referenz) x^+, und einem negativen Beispielbild x^-. Das Ziel der loss-Funktion ist dabei, die Distanz zum positiv $d(x, x^+)$ kleiner als die Distanz zum negativ $d(x, x^-)$ zu gestalten:

$$l(x, x^+, x^-) = max\{0, d(x, x^+) - d(x, x^-)\}$$

Die Distanz ist hierbei mit dem quadratischem Euklidischen Abstand berechnet und l_2-normalisiert. Für die Auswahl der Triplets werden dabei theoretisch alle passenden Kombinationen verwendet, was zu einer Komplexität von $O(n^3)$ Tripeln führt, die viele ineffiziente Beispiele beinhaltet (zu einfache Gruppen, die vorher korrekt waren und nichts zum Lernprozess beitragen), wodurch der Lernprozess relativ langsam ist.

Um dieses Problem zu verbessern wurde die ProxyNCA [10] Methode vorgestellt. Hierbei wird jedes Konzept (Label in der Annotation) durch einen vorher ermittelten Vektor, einen sogenannten Proxy repräsentiert. Dadurch dass jeder Proxy jeweils ein Konzept darstellt, können diese das positive und negative Beispielbild ersetzen. Wie man in Abb. 3 sehen kann, reduziert dieser Ansatz sehr stark die Anzahl der möglichen Tripel, und somit den Rechenaufwand.

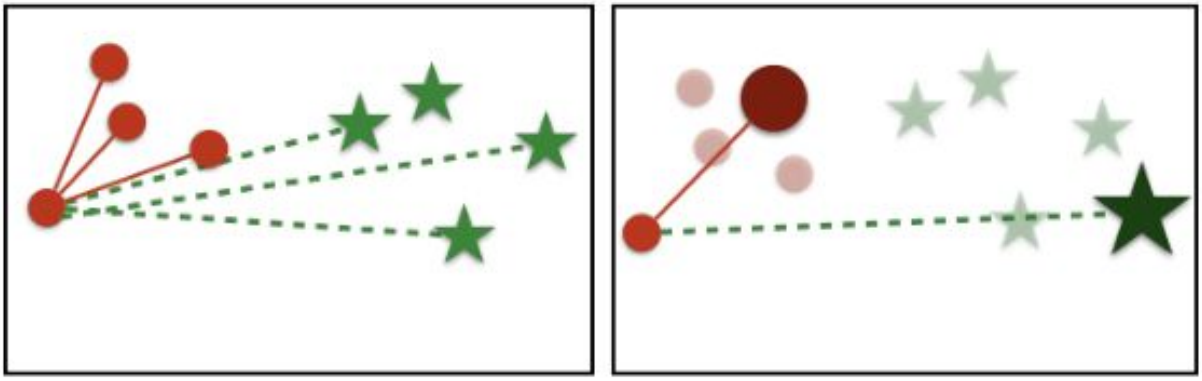


Abbildung 3: Beispiel für Komplexitäts-Reduktion von Proxies. [Links] Es sind 48 Triplets möglich (Kombinationen aus 2 Kreisen und einem Stern). [Rechts] Proxies repräsentieren Konzepte (grosser Kreis/Stern), es sind nur 8 Vergleiche nötig [10]

3.4.5 Learning with average precision: GeM(AP), (global Deep-Learning)

Bei dieser Methode handelt es sich ebenfalls um einen globalen Deep-Learning Deskriptor, weshalb grundlegende Funktionsweise genauso aufgebaut ist, wie bei Deep Metric Learning (Kap. 3.4.4). In dem hier untersuchten Paper [15] wird ein vortrainiertes ResNet (ResNet50 und ResNet101) durch ein „generalized-mean pooling“ (GeM) layer erweitert. Das Netz wird im Unterschied zu DML so trainiert, dass Kosinusähnlichkeit statt Euklidischer Distanz verwendet wird. Dies ermöglicht eine einfachere Berechnung vom Abstand zwischen den Bildern I_i und I_j mithilfe folgender Formel für die Ähnlichkeit:

$$sim(I_i, I_j) = x_i^T x_j \in [-1, 1]$$

Das Training für das vorgestellte Tool nutzt eine loss-Funktion, die Average Precision (AP) als Qualitätsmaß verwendet. Average Precision ermöglicht, die Sortierung des kompletten Datensatzes bei einer Anfrage zu evaluieren und vermeidet somit das heraussuchen von Bildergruppen (Triplets). In dem Paper wird außerdem ein hoher Fokus darauf gelegt, dass das Training mit hochauflösenden Bildern (800×800 Pixel) im begrenzten Grafikkarten-Speicher möglich gemacht wird. Aber dadurch dass die Implementation nur die Evaluation vortrainierter Modelle erlaubt, ist dieser Teil weniger relevant für diese Bachelorarbeit. Die Modelle wurden auf dem Landmarks-clean Datensatz ([1], [4]) mit einer Batchsize von B = 4096 trainiert.

3.4.6 Diffusion, (global Deep-Learning)

Bei diesem Tool wird, im Unterschied zu allen anderen, keine eigene Bildbeschreibung formuliert. Bei Diffusion [25] werden die Bildbeschreibungen von einem Tool wie z.B. GeM(AP) (Kap. 3.4.5) verwendet und Diffusion ersetzt lediglich die k-NN Suche. Bei einem Diffusion–Graphen werden Bild-Features (in dieser Untersuchung globale Features) durch Knoten repräsentiert. Jeder Knoten ist mit seinen k-nächsten Nachbarn verbunden und für jeden dieser Nachbarn wird die Ähnlichkeit gespeichert. In Abb. 4 kann man einen Teilgraphen sehen, bei dem jeder Knoten 3 Verbindungen hat. Die Ähnlichkeitswerte der verbundenen Bilder-Knoten können danach stochastisch normalisiert werden, sodass die Kanten nicht mehr einer absoluten Ähnlichkeit im Raum entsprechen, sondern die Wahrscheinlichkeit darstellen, der ähnlichste von den k–Nachbarbildern zu sein. In Abb. 5 kann man sehen, wie die Normalisierung den Graphen von Abb. 4 ändert.

Bei einer Anfrage wird der neue Bildvektor zum Graphen hinzugefügt (Nachbarn werden über k-NN Suche gefunden). Danach wird mithilfe der Kanten und Wahrscheinlichkeiten der Graph zufällig durchlaufen, wobei in jedem Schritt eine 1% Chance besteht (der 1% Wert ist anpassbar, wird aber in dem Paper und der Implementation nicht geändert), dass die Suche zum Anfangsknoten zurückkehrt. Dieser Prozess konvergiert irgendwann zu einem Endzustand, den man auch direkt berechnen kann. Die Autoren stellen in dem Paper eine optimierte Lösung dieser Berechnung vor, die es möglich macht, den Online-Teil von Diffusion zu minimieren (stattdessen wird mehr Rechenaufwand bei der Erstellung der Datenbank benötigt).

Den Vorteil bei der Suche mithilfe von Diffusion kann man in Abb. 6 sehen. Dadurch dass man nicht die Absolute Distanz vergleicht, werden Bilder in einer engen Struktur höher eingeordnet, als Bilder aus benachbarten Strukturen.

3.4.7 Deep Local Features (DELF), (lokal Deep-Learning)

Die Idee, lokale Features mit Deep Learning Verfahren zu kombinieren ist an sich nicht neu, aber die Umsetzung von DELF [11] ist die erste, die es geschafft hat lokale Deep Learning Deskriptoren zu verwenden, um gute Ergebnisse nach dem aktuellen Stand der Technik zu erzielen. Frühere hybride Ansätze hatten versucht, lokale SIFT Features anzulernen um somit die Bilder zu beschreiben. Aber durch die Weiterentwicklung von Deep Learning (und Deep Metric Learning) konnten CNNs sinnvollere Features erlernen und somit besser die Bilder beschreiben als SIFT. Diese Features werden danach bei globalen Methoden, wie in DML und GeM(AP) (Kap. 3.4.4, 3.4.5) innerhalb des Neuronalen Netzes gewichtet zu einem einheitlichen Bildvektor zusammengefasst.

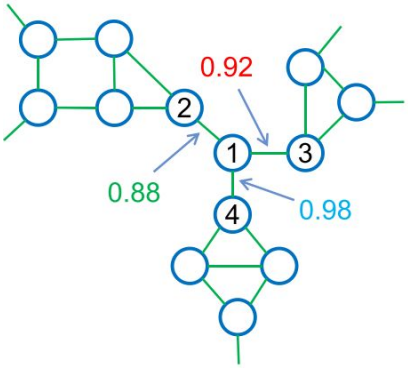


Abbildung 4: Beispiel eines gewichteten Diffusion-Graphen. Für die Erstellung wurde die KNN-Suche mit 3 Nachbarn verwendet. Die Kantengewichte stellen Ähnlichkeitswerte dar.

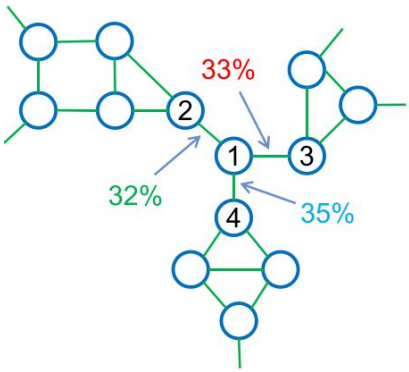


Abbildung 5: Beispiel des Graphen 4 nach Normalisierung. Die Kantengewichte wurden im Verhältnis zu den anderen Nachbarn angepasst.

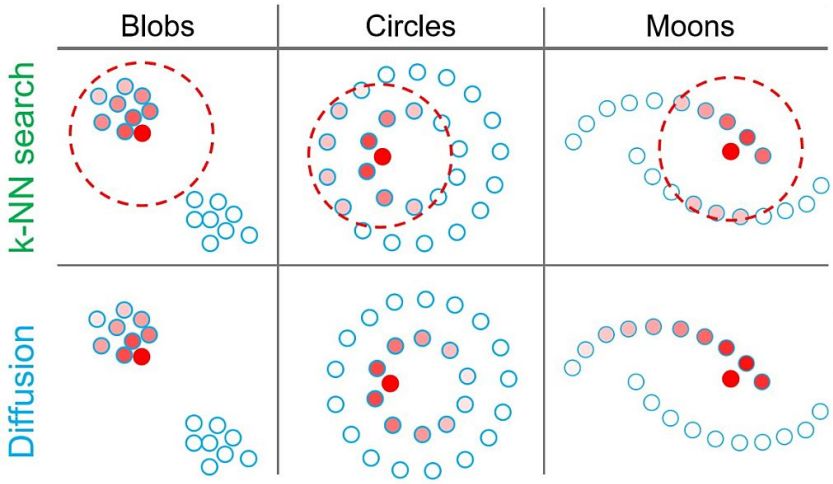


Abbildung 6: Beispiel des Nutzens der Diffusions-Suche im Unterschied zu KNN. Die rot–Färbung der kleinen Punkte stellt die Ähnlichkeit zum (komplett) roten Punkt dar.

Der theoretische Vorteil von DELF [11] ist, dass es die Vorteile von Deep Learning Features mit den Vorteilen, die durch Lokalität dieser Features entstehen, kombinieren kann. Hierfür verwendet DELF, genauso wie die anderen untersuchten Deep Learning Tools, ResNet [5] (genauer ResNet50), das für Klassifikation trainiert wurde. Für die DELF Pipeline verwendet das Tool das Ergebnis von dem conv4_x Layer von ResNet. Dies ist, wie man in Abb. 7 sehen kann, ein dichtes Netz an lokalen Feature–Vektoren, die alle Bildregionen beschreiben. Da aber viele dieser Regionen unwichtig sind und eher störend wirken würden, muss das Netz der lokalen Features im nächsten Schritt gefiltert werden. Hierfür verwendet DELF ein eigenes CNN mit 2 Schichten, das separat vom ResNet trainiert wird (siehe 7).

Obwohl das Vorgehen bei DELF sich sehr stark von SIFT unterscheidet, ist das Ergebnis dieser Schritte strukturell ähnlich, von der Hinsicht, dass man eine bestimmte Anzahl interessanter Punkte gefunden und mit einem Vektor beschrieben hat. Dementsprechend können für DELF Features auch die gleichen Methoden für den Umgang mit lokalen Features (Zusammenfassung, Suche usw.) verwendet werden. Deshalb werden DELF Features mit ASMK (Kap. 3.4.3)

verwaltet.

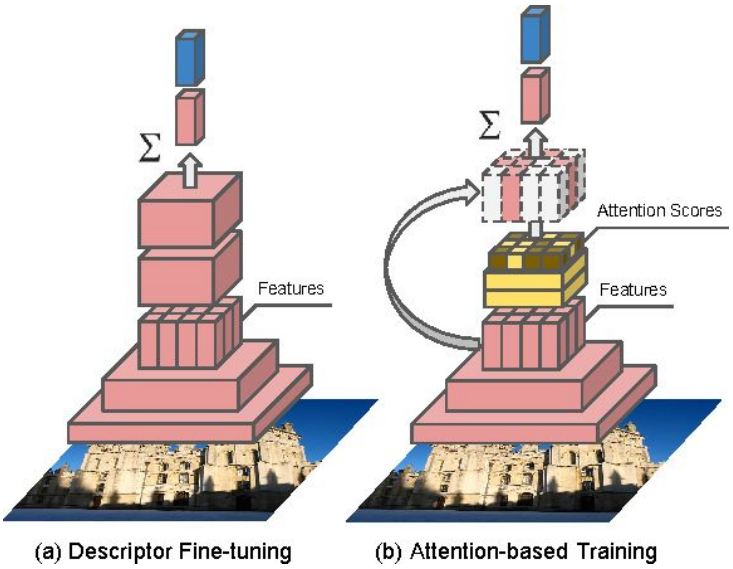


Abbildung 7: DELF Bildbeschreibung und die für das Training notwendigen Stufen. Die lokalen Features werden mit ResNet (rot) extrahiert und mit einem zweiten CNN (gelb) gefiltert.

3.4.8 ASMK mit HOW-Deskriptor (ASMK-HOW), (lokal Deep-Learning)

Das Modell von Tolias et al. [21] ist eine Weiterentwicklung von ASMK [20] und verwendet das oben vorgestellte ASMK-Modell (Kap. 3.4.3) um lokale Features für CBIR zu verwenden. Konzeptionell ist dieses Tool ähnlich zu DELF (Kap. 3.4.7), von der Hinsicht dass auch hier ResNet [5] für die Erzeugung lokaler Features verwendet wird und dass die gefilterten lokalen Features mit ASMK (Kap. 3.4.3) verwaltet werden. Das Training vom ResNet erfolgt hier über globales Metric Learning (Kap. 3.4.4), wodurch globale Label ausgenutzt werden können, um die Netzparameter beim Training zu optimieren. Das Training erfolgt explizit mit einem globalen Deskriptor, weil hierbei dieselben Parameter wichtig sind, wie auch bei der lokalen Beschreibung. Außerdem würde beim Training mit lokalen Features eine zusätzliche Abhängigkeit von einem Codebook entstehen (was Zeit kostet).

Einer der größten Unterschiede zu DELF (Kap. 3.4.7) ist, dass der HOW-Deskriptor nicht separat die Wichtung (Attention Scores) der lokalen Features lernen muss und stattdessen eine statische Funktion verwendet. Diese verlässt sich darauf, dass unwichtige lokale Features eine geringere Wichtung (dargestellt durch l_2 Norm) bekommen. Diese Eigenschaft wird durch den globalen Deskriptor gelernt, um bessere Ergebnisse beim Sum-pooling zu erzielen. Wie man in Bild 8 sehen kann, wird diese Wichtungsfunktion $w(\cdot)$ beim Training für die Feature-Vektoren angewandt, um den Fokus der Features zu verbessern. Dieselbe Funktion wird danach auch beim Testen verwendet, um die lokalen Features zu filtern.

4 Evaluation der Tools

In diesem Abschnitt werden die Ergebnisse der Tools ausgewertet und miteinander verglichen. Hierfür werden die Datensätze ROxford5k ([12], [13]) und NUS-WIDE [3] verwendet. Diese Da-

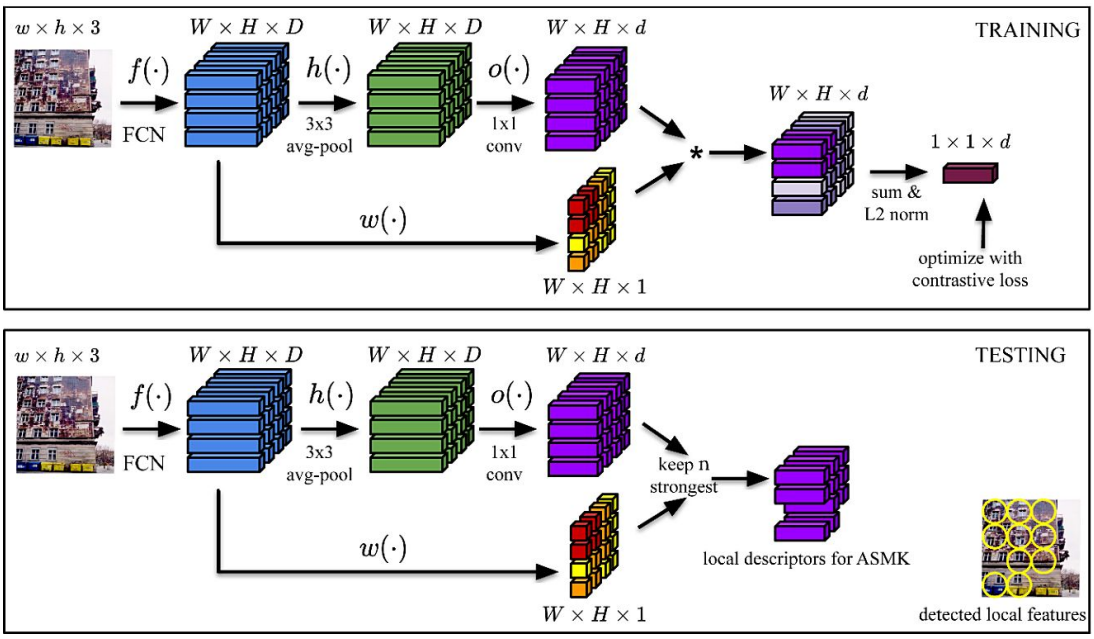


Abbildung 8: Unterschied der Architektur zwischen globalem Training und Tests mit lokalen Features. Die Beschreibung und Wichtung der Features kann global gelernt und lokal angewandt werden.

tensätze wurden ausgewählt, um zwei unterschiedliche Domänen von CBIR vergleich zu können. ROxford5k (und die vorherige Version Oxford5k) kommt in der untersuchten Literatur am häufigsten vor. Er repräsentiert die Fähigkeit eines Verfahrens, sehr spezielle Bildeigenschaften anhand von Gebäude-Bildern zu erkennen um die Gebäude voneinander trennen zu können. Diese Spezialisierung sollte theoretisch besser durch Neuronale Netze erreichbar sein, da man diese auf solche Spezialfälle feintunen kann.

Als zweiten Datensatz wurde NUS-WIDE [3] verwendet, um im Unterschied zu Oxford, die Fähigkeit der Tools zu testen, mit allgemeinen und unterschiedlichen Bildern umzugehen. Die meisten untersuchten Publikationen betrachten nicht diesen Aspekt und konzentrieren sich stattdessen auf eine spezifische Anwendung (z.B. Gebäudeerkennung). Aus diesem Grund ist es besonders interessant herauszufinden, wie die Qualität der Ergebnisse mit diesem Datensatz aussieht.

4.1 Methoden

Für die Evaluation der Ergebnisse werden zunächst die Annotationen der Datensätze zusammen mit vordefinierten Anfragebildern verwendet. Mithilfe dieser Daten können vergleichbare Qualitätsmaße für die Tools bestimmt werden. Als Maß für ROxford5k wird der mAP (mean average precision) verwendet. Dieses Maß berechnet für jede Anfrage und jede mögliche Anzahl von Ergebnisbildern den Precision Wert (Anteil richtiger Ergebnisse in der Ergebnismenge) und gibt den Durchschnitt zurück. Der mAP ist vor allem praktisch, weil er ein direktes Maß für die Sortierung der Ergebnisbilder darstellt und dadurch alleine die durchschnittliche Qualität der Tools auf dem Datensatz bewerten kann.

Aufgrund der Größe von NUS-WIDE, ist es nicht plausibel den Datensatz komplett zu sortieren. Deshalb wird hier statt des mAP nur der Precision und Recall Wert berechnet. Der Recall

bestimmt hier den Anteil der richtigen Ergebnisse im Verhältnis zu allen korrekten Ergebnissen. Dieser Wert wird vor allem bei Ergebnismengen interessant, die größer sind als die Anzahl aller richtigen Ergebnisse ist. Hier hängt die Qualität nicht nur davon ab, wie viele der Ergebnisse gut sind (Precision), sondern auch wie viele der guten Ergebnisse gefunden wurden (Recall).

Für die Evaluation wurden 1000 Anfragen definiert und es wurde für jede Anfrage der Precision und Recall Wert für 10, 1000 und 30000 Ergebnisse berechnet. Danach wurden die Qualitätsmaße über alle Anfragen zusammengefasst, wodurch wir zu dem Maß mP@k und mR@k kommen (mean Precision/Recall bei k Ergebnisbildern).

Zusätzlich zu den quantitativen Analysen der Ergebnisbilder mithilfe der bereits beschriebenen Precision und Recall Werte, wird in dieser Arbeit zusätzlich noch eine qualitative Analyse einiger Ergebnisbilder durchgeführt. Hierdurch verlässt man sich nicht auf die reine Semantik der Datensatz–Annotationen, stattdessen kann man zusätzlich ein Subjektives Maß verwenden. Mithilfe der qualitativen Analyse kann man auch genauer untersuchen, welche Stärken und Schwächen die unterschiedlichen Tools und Gruppen von Tools haben. Außerdem kann man bei einer genauen Betrachtung auch herausfinden, welche Faktoren zu den quantitativen Ergebnissen beitragen haben könnten.

Neben der der Evaluation der Ergebnisse werden in dieser Arbeit auch andere Faktoren untersucht, wie die Laufzeiten der Implementationen, deren Speicherbedarf auf der Festplatte und einige Probleme in der Anwendung, die beim Testen aufgefallen sind, und Hardware–abhängig einige Kompromisse (z.B. Parameter-Anpassungen) erzwungen haben.

4.1.1 Revisited Oxford5k

ROxford5k besteht aus 4993 Bildern und zusätzlich 70 Anfragen. Zu jedem dieser Anfragebilder haben die Autoren [13] definiert, welche Datenbank-Bilder einfach, schwer oder nicht erkennbar sind. Somit sind alle nicht in der Annotation vorhandenen Bilder als falsch zu betrachten. Anhand dieser Annotationen kann man mit dem Datensatz drei Schwierigkeitsstufen: „Easy“, „Medium“ und „Hard“ definieren. Bei „Easy“ werden nur einfache von falschen Bildern unterschieden, bei „Hard“ wird nur anhand der schweren Ergebnisse evaluiert und bei „Medium“ sind sowohl einfache, als auch schwere Bilder wichtig.

Evaluation bei ROxford5k erfolgt über den mAP (mean average precision). Dieser ist der Durchschnitt über alle Anfragen, wobei bei jeder Anfrage der Precision-Wert jede mögliche Anfragelänge berechnet wird (P@1 bis P@4993). Diese Evaluation ermöglicht es, die Sortierung des kompletten Datensatzes direkt zu bewerten.

In Abb. 9 sind einige Beispielbilder aus dem Oxford5k Datensatz dargestellt, um

4.1.2 NUS-WIDE

Der zweite verwendete Datensatz, NUS-WIDE [3] musste zunächst auf 208516 Bilder (statt 269648) reduziert werden, weil einige Bilder aus dem originalen Datensatz nicht mehr verfügbar sind. NUS-WIDE wurde mit 81 „Labels“ annotiert, wobei jedes Label einem Konzept aus der realen Welt entspricht. Die Anzahl der Labels ist dabei nicht vorbestimmt und dadurch können einige Bilder keine oder mehrere Labels haben.

Weiterhin wurde der Datensatz durch eine Liste aus 1000 Anfragebildern erweitert, um eine standardisierte Evaluation durchführen zu können. Die Anfragebilder sind zufällig ausgewählt,

Abbildung 9: Beispiel - Anfragebilder aus dem Oxford5k Datensatz

wobei für jedes Label mindestens ein Bild in dieser Liste vorhanden ist.

Diese Art der Evaluation unterscheidet sich von der Methodik von He et. al [5] und nachfolgenden dadurch, dass nicht nur die 21 am häufigsten verwendeten Labels, wie „sky“ und „animal“ verwendet werden, sondern auch die selteneren Konzepte in den Anfragen vertreten sind. Die Ergebnisse der Anfragen können beliebige Bilder aus dem Datensatz sein. Hierdurch ist das Ziel nicht, wie in [5], wenige Konzepte voneinander zu trennen, stattdessen müssen die Tools ein Label in den Bildern erkennen und im Datensatz wiederfinden. Bei der Evaluation gilt ein Ergebnis als richtig, wenn mindestens ein Label in Anfrage und Ergebnis gleich sind. Es wird Precision und Recall für 10, 1000 und 30000 Ergebnisse berechnet (mR@10 und mR@1k werden nicht untersucht, weil bei 1000 Ergebnissen der Precision Wert bis zu 99,78 % erreichten kann und dadurch für die Bewertung der Tools ausreichend ist).

Abb. 10 stellt einige Beispielbilder mit den dazugehörigen Annotationen dar.

4.2 Quantitative Ergebnisse

In diesem Abschnitt werden quantitative Auswertungen der Ergebnisse von den einzelnen Tools aufgeführt und miteinander vergleichen. Es werden auch erste Hypothesen zur Begründung dieser Werte gebildet, die bei der qualitativen Analyse überprüft werden.

4.2.1 ROxford5k

Die quantitativen Ergebnisse spiegeln im allgemeinen die zeitliche Entwicklung der untersuchten Tools wieder. Man kann sehen, dass Deep Learning Features viel bessere Ergebnisse erzielen können als Hand-crafted Features mit SIFT oder ORB.

Bei den Hand-crafted Features lässt sich auch ein sehr großer Unterschied zwischen SIFT und ORB beobachten, was eine Begründung dafür sein könnte, wieso SIFT in der Literatur so häufig verwendet wird. Die Untersuchung vom Einfluss der Codebook-Größe zeigt auch, dass VLAD von einem größeren Codebook profitieren kann. Für ASMK ist in diesem Experiment die

Abbildung 10: Beispiel - Bilder aus dem NUS-WIDE Datensatz mit Annotation

in der Literatur verwendete Codebook Größe von 65536 Wörtern optimal.

Im Vergleich mit beiden Deep Learning Gruppen sieht man, dass die Hand-crafted Tools einen viel kleineren mAP erreichen. In der zweiten Gruppe, den globalen Deep-Learning Methoden ist der Unterschied zwischen den einzelnen Tools sehr groß. GeM(AP) (Kap. 3.4.5) erreicht hier sehr gute Ergebnisse. Man sieht, dass die Verwendung von dem Größeren ResNet101 die Retrieval-Qualität bemerkbar verbessert. In dieser Gruppe ist vor allem der Vergleich zwischen Gem(AP) und DML interessant. Wie man in der Tabelle 1 sehen kann, haben beide DML Modelle einen viel kleineren mAP Wert. Der Grund dafür könnte darin liegen, dass DML nur mit einer Auflösung von 224×224 Pixeln arbeitet. Vor allem bei einem Benchmark wie ROxford5k, wo das erkennen feiner Eigenschaften wichtig ist, kann dieser Nachteil zu großen Unterschieden in der Qualität der Ergebnisse führen. Ein anderer wichtiger Faktor sind die Trainingsdaten und die verwendeten Parameter, wobei von den letzteren hier abstrahiert wird, in der Annahme dass die bestmöglichen Parameter für das Training verwendet wurden (es wurde nur DML trainiert, wobei alle Parameter getestet und die besten gewählt wurden).

Für das Training von DML werden die Datensätze Oxford und Sfm verwendet. Das Training auf Oxford entspricht zwar nicht einer realistischen Anwendung, liefert aber vor allem bei den schweren Bildern bessere Ergebnisse. Das Training auf SfM120k Sfm ([14]) (Gebäude Datensatz mit 120.000 Bildern von 3D Modellen von Gebäuden aus unterschiedlichen Winkeln, er wurde für das Training vom HOW Deskriptor in [21] verwendet) liefert im Durchschnitt schlechtere Ergebnisse, ist aber bei der leichten Herausforderung besser, was darauf hindeutet, dass die Erkennung kompletter Gebäude besser trainiert wurde (mit dem Größeren Datensatz). Bei GeM(AP) wurde für das Training der Landmarks [1] Datensatz verwendet, der im Unterschied zu SfM aus vielfältiger ist.

In der Gruppe globaler Deep-Learning Methoden wurde auch Diffusion (Kap. 3.4.6) untersucht. Hierfür wurden die Features von GeM(AP) mit ResNet101 verwendet. Die Verwendung

Ergebnisse ROxf5k			
Toolname	Easy	Medium	Hard
VLAD - SIFT (2048 W.)	40,35%	32,36%	14,83%
VLAD - SIFT (128 W.)	17,67%	13,13%	2,96%
VLAD - ORB (2048 W.)	10,80%	7,73%	1,18%
ASMK - SIFT (65536 W.)	40,35%	31,67%	13,14%
ASMK - SIFT (2048 W.)	35,37%	28,25%	11,61%
ASMK - SIFT (1048576 W.)	35,95%	27,98%	11,65%
ASMK - ORB (65536 W.)	19,17%	12,99%	2,06%
DML (ResNet50, trained on Sfm)	48,25%	36,27%	11,28%
DML (ResNet50, trained on Oxf)	47,31%	40,02%	18,07%
GeM(AP) (ResNet50)	79,28%	64,51%	37,64%
GeM(AP) (ResNet101)	86,57%	71,59%	46,95%
GeM(AP) (ResNet101) + diffusion	91,17%	77,09%	54,76%
DELF (ResNet50, 65536 W.)	85,96%	72,58%	45,64%
ASMK–HOW (ResNet50, 65536 W.)	92,71%	78,53%	54,56%

Tabelle 1: Evaluation Ergebnisse ROxford5k. Es werden alle Tools mit allen 3 Herausforderungen verglichen.

von Diffusion führt zu einer Verbesserung der Ergebnisse, vor allem bei der schweren Challenge von ROxford5k. Wenn man mit den Vorteilen aus Abb. 6 vergleicht, sieht man auch, dass Diffusion hauptsächlich zu Verbesserungen führen kann, wenn die Features suboptimal (für kNN Suche) angeordnet sind, wie es vermutlich beim schweren ROxf5k der Fall ist.

Die letzte und in diesem Benchmark beste Gruppe sind die lokalen Deep Learning Tools. Beide Tools in dieser Gruppe haben mit ResNet50 Ergebnisse erreicht, die mit GeM(AP) mit ResNet101 vergleichbar sind. Vor allem der HOW Deskriptor erreicht sehr gute Ergebnisse, und der Aspekt, dass hierfür ein kleineres CNN ausreicht, deutet darauf hin, dass lokale Features bessere Ergebnisse liefern können, vor allem wenn man spezifische Eigenschaften unterscheiden muss. Beide Tools arbeiten in dieser Gruppe mit hohen Auflösungen, was die Hypothese unterstützt, dass die Ergebnisse von DML mit der Auflösung begründet werden können.

4.2.2 NUS-WIDE

Bei der Evaluation von NUS-WIDE wurde eine Liste aus 1000 Anfragen vordefiniert. Für jedes der 1000 Anfragebilder muss ein Ergebnis mindestens ein Label mit der Anfrage gleich haben, um bei der Evaluation als korrekt bewertet zu werden. Ausgehend aus dieser Regel von Korrektheit/Ähnlichkeit, sind ca. 21,99% aller Bilder ähnlich zu den vordefinierten 1000 Anfragen. Ausgehend aus dieser Zahl, kann man bestimmen, dass der Erwartungswert für Precision von einer zufälligen Ergebnismenge beliebiger Größe 21,99% beträgt.

Für die Evaluation der Tools wurden die mean Precision Werte bei 10, 10000 und 30000 Ergebnissen herausgesucht. Die mP@10 (mean precision bei 10 Ergebnissen) repräsentiert dabei die Qualität der besten Ergebnisse. Diese Ergebnisse zeigen die Fähigkeit des Tools, die Anfrage zu erkennen und die ähnlichsten Bilder herauszusuchen, ohne größere Probleme mit der Verfügbarkeit ähnlicher Bilder zu haben. In der praktischen Anwendung würden die ersten 10 Bilder auch den höchsten Wert für den Nutzer haben, weshalb diese Metrik auch für die Anwendung interessant ist. Wichtig zu beachten beim mP@10 ist auch der Fakt, dass die Anfragebilder

in der Datenbank vorhanden sind, wodurch das kleinstmögliche Ergebnis bei 10% liegt (für ein funktionierendes CBIR System, das zumindest ein Bild in der Datenbank wiederfinden und als sehr ähnlich bewerten kann). Bei den anderen Metriken, mP@1k und mP@30k werden immer größere Ergebnismengen untersucht wobei auch andere Aspekte Einfluss auf die Evaluation haben. Bei mP@30k liegt das bestmögliche Ergebnis bei 70,42%, weshalb hier zusätzlich der mean Recall Wert (mR@30k) angegeben wird (mR@30k kann maximal 70,26% erreichen). Diese Grenzen sind dadurch bedingt, dass einiger der Anfragen weniger als 30 tausend korrekte Ergebnisse haben und somit den Durchschnitt herunterziehen.

Die Ergebnisse aus Tabelle 2 zeigen, dass Globale Deep Learning Features besonders gute Ergebnisse auf NUS-WIDE erreichen. DML und ASMK haben dabei die besten Werte in den jeweiligen Gruppen, was vor allem damit zu begründen ist, dass die Deep Learning Modelle auf NUS-WIDE feingetuned wurden. Bei GeM(AP) war ein Training mit der verfügbaren Implementation nicht möglich, aber es wurden trotzdem relativ gute Ergebnisse erreicht. Hier wurde auch ein Vergleich zwischen ResNet101 und ResNet50 durchgeführt, bei dem ResNet101 zwar schlechtere Precision Ergebnisse hatte, aber dafür einen der besten Recall-Werte erreichen konnte. Diese Beobachtung deutet darauf hin, dass ResNet101 vor allem bei spezifischen Anfragen (mit wenigen korrekten Ergebnissen) sehr gute Ergebnisse liefern konnte (aber dafür schlechter mit allgemeinen Anfragen umgehen).

Bei ASMK-HOW wurden zwei unterschiedliche Ergebnisse angegeben, die sich darin Unterscheiden, mit wie vielen Skalierungen die lokalen Features gefunden wurden. ASMK-HOW und DELF verwenden nativ eine Auflösung von 1024×1024 Pixeln und Skalieren die Bilder auf mehrere unterschiedliche Größen (vom 0,25 bis 2,0 - fachen der Originalen Auflösung). Bei Oxford werden 7 unterschiedliche Skalierungen verwendet, die für NUS-WIDE auf 1 und 5 reduziert wurden. Die Verwendung von mehr Skalierungen führt zu besseren Ergebnisse, wie in Tabelle 2 gezeigt wird. Allerdings führt das auch zu schlechteren Laufzeiten und bei der Verwendung von 7 Skalierungen in NUS-WIDE kam es vermutlich Hardwarebedingt zu Fehlern, weshalb hier nur 5 verwendet werden.

Diffusion hat auf NUS-WIDE nicht zu einer quantitativen Verbesserung der Ergebnisse geführt.

Bei den Hand-crafted Features konnten SIFT-Features wieder bessere Ergebnisse erreichen, als ORB. Der Unterschied bei NUS-WIDE ist aber viel geringer. Bei VLAD musste aufgrund der Größe der Datenbank die Anzahl der Wörter reduziert werden. Dadurch dass in Tabelle 1 ein Vergleich der Codebook Größe durchgeführt wurde, ist anzunehmen, dass VLAD mit 2048 Wörtern bessere Ergebnisse erreichen könnte, es aber nicht Möglich war (genauere Daten sind in Tabelle 4). Die Hand-crafted Deskriptoren haben insgesamt ziemlich schlechte Ergebnisse gezeigt, vor allem bei vielen Ergebnissen (mP@30k) haben sie die untere Grenze nur sehr knapp übertroffen (mit nur 1,32% - 3,71%).

4.3 Vergleich Ressourcen/Randbedingungen

Dieser Abschnitt vergleicht andere wichtige Faktoren, die für die Evaluation der Tools wichtig sein können. Zu diesen Faktoren zählt vor allem Speicherbedarf und Laufzeiten. Dies ist vor allem wichtig, weil ein zu hoher Speicherbedarf dazu führen kann, dass bestimmte Kompromisse auf Kosten der Qualität eingegangen werden müssen. Die Laufzeiten sind auch ein wichtiger

Ergebnisse NUS-WIDE (1k Anfragen)				
Toolname	mP@10	mP@1k	mP@30k	mR@30k
baseline (zufällig)	21,99%			14,39%
VLAD - SIFT (128 W.)	10,33%	27,86%	25,30%	17,46%
VLAD - ORB (256 W.)	33,85%	24,93%	23,37%	15,71%
ASMK - SIFT (65536 W.)	34,58%	27,39%	24,63%	16,78%
ASMK - SIFT (2048 W.)	36,95%	31,12%	25,70%	18,05%
ASMK - SIFT (1048576 W.)	33,44%	25,52%	24,05%	16,25%
ASMK - ORB (65536 W.)	32,32%	24,23%	23,31%	15,78%
DML (ResNet50)	87,24%	81,82%	62,79%	49,36%
GeM(AP) (ResNet50)	74,05%	60,29%	36,93%	28,47%
GeM(AP) (ResNet101)	71,81%	55,20%	34,04%	36,48%
GeM(AP) (ResNet50) + diffusion	73,49%	59,73%	32,51%	24,48%
DELF (ResNet50, 65536 W.)	67,86%	52,75%	36,04%	25,50%
ASMK–HOW (cale) (ResNet18, 65536 W.)	66,67%	57,75%	41,14%	30,28%
ASMK–HOW (5 scales) (ResNet18, 65536 W.)	70,68%	58,51%	39,96%	29,65%

Tabelle 2: Evaluation Ergebnisse NUS-WIDE. Es wurden 1000 Bilder zufällig ausgewählt und die Durchschnittlichen Precision und Recall Werte verglichen. Bei den gewählten Bildern sind durchschnittlich 21,99% aller möglichen Ergebnisse korrekt (beinhalten mindestens ein Label vom Anfragebild). Daraus ergibt sich die untere Schranke von 21,99% Precision und 14,39% Recall@30000, gegen die die Tools abgeglichen werden.

Aspekt und können für die Praktikabilität eines Tools entscheidend sein.

4.3.1 Speicherverbrauch

Hier wird der Speicherverbrauch aller Tools mit einigen Parametern gegenübergestellt. Es wird der benötigte Arbeitsspeicher für die Erstellung der Datenbank sowie für eine Anfrage untersucht. Mit diesen Daten kann man die Grenzen der Tools finden, um z.B. eine Hardwareanforderung oder maximale Datenbank-Größe formulieren zu können. Alle Experimente wurden auf einem PC mit 32 GB Arbeitsspeicher durchgeführt und die Parameter wurden dementsprechend angepasst (konkrete Anpassungen werden beschrieben). Manche kurze Überschreitungen der 32 GB werden mithilfe von Windows Pagefiles ermöglich, was aber zu einer erhöhten Absturzwahrscheinlichkeit geführt hat.

Außerdem wird hier der benötigte Festplattenspeicher untersucht. Dieser ist vor allem im Kontext zu der Größe der Bilder interessant und kann im Zusammenhang mit den qualitativen Ergebnissen eine Auskunft darüber geben, wie effizient die Methoden Bilder zusammenfassen und beschreiben können. Die Größe Der Bilder beträgt für Oxford5k: 1,84 GB und für NUS-WIDE: 35,55 GB.

Speicher Oxford5k In Tabelle 3 sieht man, dass der Oxford Datensatz mit 5063 Bildern zu klein ist, um Probleme mit dem Arbeitsspeicher zu haben. Trotzdem fällt hier VLAD auf, weil es bei 2048 Wörtern und SIFT Features über 21 GB Arbeitsspeicher benötigt. Die Größe der Datenbank auf der Festplatte ist hierbei auch 10 GB, die für jede Anfrage wieder zu 20 GB entpackt werden. Das liegt daran, dass VLAD Features (siehe Abschnitt Kap. 3.4.2) für jedes Bild

Speicherbedarf ROxf5k			
Toolname	RAM (Erstellung)	Festplatte	RAM (1 Anfrage)
VLAD - SIFT (2048 W.)	21,26 GB	10,13 GB	20,73 GB
VLAD - SIFT (128 W.)	1,33 GB	648 MB	1,30 GB
VLAD - ORB (2048 W.)	5,32 GB	2,53 GB	5,18 GB
ASMK - SIFT (65536 W.)	3,74 GB	108 MB	243 MB
ASMK - SIFT (2048 W.)	3,71 GB	75 MB	85 MB
ASMK - SIFT (1048576 W.)	4,25 GB	275 MB	1,97 GB
ASMK - ORB (65536 W.)	1,03 GB	50 MB	168 MB
DML (ResNet50)	35 MB	3 MB	9 MB
GeM(AP) (ResNet50)	142 MB	40 MB	142 MB
GeM(AP) (ResNet101)	174 MB	40 MB	174 MB
GeM(AP) (ResNet101) + diffusion	312 MB	79 MB	173 MB
DELF (ResNet50, 65536 W.)	200 MB	162 MB	334 MB
ASMK–HOW (ResNet50, 65536 W.)	5,45 GB	103 MB	233 MB

Tabelle 3: Speicherbedarf der Tools für Erstellung und Anfrage auf dem ROxford5k Datensatz.

Informationen für jedes Wort speichern. Dadurch ist die Größe der Datenbank direkt proportional zur Anzahl visueller Worte im Codebook. Dies wird auch durch das zweite Experiment von VLAD, mit 128 Wörtern und SIFT Features, bestätigt. Dadurch dass das Codebook in dem zweiten Experiment 16× kleiner ist, benötigt das Tool auch 16× weniger Arbeits- und Festplattenspeicher.

Im Vergleich zwischen den Gruppen sieht man auch, dass die Tools mit lokalen Features im Allgemeinen mehr Arbeitsspeicher benötigen, als die Tools mit globalen Features. Der Grund hierfür ist, dass die Implementation für ASMK und ASMK–HOW zunächst unkomprimierte lokale Features extrahiert, und diese danach quantisiert und zusammenfasst. Dadurch befinden sich zu einem Zeitpunkt bei der Erstellung der Datenbank alle unkomprimierten Features im Arbeitsspeicher.

Bei der Implementation von DELF wird dieses Problem umgangen, indem jedes Bild einzeln beschrieben und aggregiert wird. Die Ergebnisse von jedem Schritt werden bei dieser Implementation auf der Festplatte zwischengespeichert.

Bei den globalen Tools sieht man, dass DML eine kompaktere Speicherform der Features verwendet, als GeM(AP). Man kann auch sehen, dass ResNet101 etwas mehr Speicher benötigt, als ResNet50. Und bei Diffusion ist vor allem eine zusätzliche Belastung der Festplatten zu sehen, die dadurch kommt, dass Diffusion nicht die eigentlichen Features ersetzt, sondern diese nur durch einen Graphen erweitert.

Speicher NUS-WIDE Im allgemeinen ist der Speicherbedarf für NUS-WIDE in Tabelle 4 vergleichbar mit den Messwerten für Oxford in Tabelle 3 (hochgerechnet auf die Anzahl der Bilder). Alle globalen Tools und DELF erreichen eine kompakte Bildrepräsentation und benötigen dafür relativ wenig Arbeitsspeicher. Den genauen Wert für die Erstellung mit DELF zu messen war aufgrund einer sehr hohen Laufzeit (in Tabelle 6 ist eine Schätzung) nicht möglich. Außerdem haben viele Programmabstürze während der Laufzeit eine genaue Messung unmöglich gemacht.

Speicherbedarf NUS-WIDE			
Toolname	RAM (Erstellung)	Festplatte	RAM (1 Anfrage)
VLAD - SIFT (128 W.)	55,53 GB	26,47 GB	53,40 GB
VLAD - ORB (256 W.)	27,81 GB	13,24 GB	26,71 GB
ASMK - SIFT (65536 W.)	13,10 GB	3,90 GB	4,84 GB
ASMK - ORB (65536 W.)	4,63 GB	1,79 GB	1,97 GB
DML (ResNet50)	1,35 GB	112 MB	345 MB
GeM(AP) (ResNet50)	185,01 MB	1,60 GB	1,74 GB
GeM(AP) (ResNet50) + diffusion	12,15 GB	3,15 GB	3,36 GB
DELF (ResNet50, 65536 W.)	<10 GB	4,18 GB	6,88 GB
ASMK–HOW (1 scale) (ResNet18, 65536 W.)	14,82 GB	1,30 GB	1,48 GB
ASMK–HOW (5 scales) (ResNet18, 65536 W.)	6,17 GB	1,73 GB	1,93 GB

Tabelle 4: Speicherbedarf der Tools für Erstellung und Anfrage auf dem NUS-WIDE Datensatz. Parameter von VLAD wurden im Vergleich zu ROxf5k angepasst, um die Arbeitsspeicherkapazität von 32 GB (soweit wie möglich) nicht zu überschreiten.

Da bei VLAD, wie in Kapitel Kap. 3.4.2 dargestellt, die Größe der finalen Datenbank und dadurch auch der Arbeitsspeicherbedarf von der Anzahl der Wörter abhängt, musste diese bei NUS-WIDE auf 128 Wörter für SIFT und 256 Wörter für ORB gesenkt werden, um mit den verfügbaren 32 GB RAM (und Windows Pagefile) ein Ergebnis erreichen zu können. Hier musste ein Kompromiss auf Kosten der Qualität eingegangen werden. Somit sind die genauen Messwerte schon weniger relevant, da Parameter angepasst wurden um diese Werte zu erreichen. Demnach kann man folgern, dass VLAD nicht mit großen Codebooks funktionieren kann, diese sind aber wichtig für die Qualität der Ergebnisse (siehe Tabelle 1). Diese Probleme werden aber bei ASMK gelöst, weshalb man hier schließen kann, dass ASMK allein aufgrund der Größe der Datenbank besser ist als VLAD.

Um auf die Ergebnisse von ASMK (auch ASMK-HOW) zu kommen, musste die Pipeline so angepasst werden, dass man die Bilder in einzelnen Batches beschreibt. Bei der Verarbeitung eines Batches werden alle lokalen Features für ein Batch extrahiert. Danach werden diese mit ASMK zusammengefasst und in den Index hinzugefügt, bevor der nächste Batch angefangen wird. Durch dieses Vorgehen ist es möglich, ein große Datenbank zu erstellen, ohne alle lokalen Features im Arbeitsspeicher behalten zu müssen. Dieser Ansatz ist dabei schneller als DELF, bei dem das Arbeitsspeicher - Problem gelöst wird, indem die Features auf der Festplatte zwischengespeichert werden. DELF hat außerdem dadurch den Nachteil, dass während der Erstellung der Datenbank viel Festplatten - Speicher temporär benötigt wird.

Die Batch-Verarbeitung von ASMK ist auch ein wichtiger Faktor bei der Erstellung von einem Codebook. Dieses muss mit den unkomprimierten lokalen Features gebildet werden, wodurch die Menge der Bilder, die das Codebook beeinflussen, eingeschränkt ist. In der Literatur würde dies keine besonderen Probleme bedeuten, weil die Codebooks dort vorher auf anderen Daten generiert werden. Die Qualität von einem Codebook scheint auch zufällig zu sein, wobei ein Paar im Rahmen dieser Arbeit durchgeführte Versuche darauf deuteten, dass mehr Bilder zu besseren Codebooks führen. In der Literatur wird auch erwähnt, dass das Training vom Codebook auf dem Datensatz zu gute Ergebnisse liefern würde, weshalb dafür ein anderer Datensatz verwendet werden soll. Für die Praxis würde dies aber bedeuten, dass man einen zweiten Daten-

satz benötigt, nur um ein Codebook zu erstellen, was sehr unpraktisch ist. Aus diesem Grund wurden alle Codebooks für NUS-WIDE auf dem Datensatz selbst erstellt.

Für die Erstellung vom Codebook wurden zwei unterschiedliche Verfahren versucht, die das Problem versuchen zu umgehen, dass nicht alle lokalen Features gleichzeitig im RAM sein können. Bei dem ersten Ansatz wurde das Codebook nur auf dem ersten Batch trainiert (geclustert). Bei dem zweiten Ansatz wurde jeder Batch geclustert (auf die gleiche Anzahl Zentren, wie die angestrebte Codebook-Größe) und die daraus erzielten Cluster-Zentren wurden am Ende zu dem finalen Codebook geclustert. Dieser Ansatz ist zwar aufwändiger, aber er konnte bei der Evaluation bessere Ergebnisse erzielen, weshalb er für die in Tabelle 2 gezeigten Ergebnisse verwendet wurde.

Es wurde auch eine Batch - Vorgehensweise für VALD versucht, aber es war aufgrund von Fehlern und der Größe der finalen Datenbank nicht möglich.

4.3.2 Laufzeit

In diesem Abschnitt werden die Laufzeiten der Tools untersucht. Der erste Messwert ist die benötigte Zeit zum Erstellen der Bilddatenbank. Der zweite (und dritte bei Oxford) Messwert ist eine Zeitmessung für eine einfache Anfrage. Bei Oxford wird hierbei der Einfluss der Ergebnisgröße auf die Laufzeit untersucht. Der letzte Messwert ist die Laufzeit von 100 Anfragen (mit jeweils 100 Ergebnissen). Bei diesem Experiment können einige Tools Optimierungen von Batch-Anfragen ausnutzen und es werden Verzögerungen, die durch das laden der Modelle und Datenbanken entstehen, ausgeglichen.

Die in diesem Kapitel vorgestellten Laufzeiten beinhalten nicht die benötigte Zeit, um Deep Learning Modelle zu trainieren, oder um die Codebooks für lokale Features zu erstellen.

Für die Messung aller Laufzeiten wurde ein Computer mit einem Intel Core i7 6700k (4 Kerne, 8 Threads mit 4,6GHz) und einer Nvidia Geforce GTX 1080 verwendet. Die GPU wurde von den Tools benutzt, um Deep Learning Features der Bilder zu beschreiben, und um lokale Features bei ASMK zu quantisieren (VLAD ist das einzige Tool, das keine Vorteile durch die GPU hat).

Laufzeiten ROxford5k In Tabelle 5 sind die Laufzeiten dargestellt, die die einzelnen Tools (mit den Parametern) gebraucht haben.

Für die Erstellung der Datenbanken haben globale Deep-Learning Tools, am wenigsten Zeit benötigt. Der Grund dafür ist, dass ein Großteil der Zeit dafür verwendet wird, die Bilder zu beschreiben. Dabei ist zwar der Rechenaufwand für ResNet viel höher als für SIFT oder ORB, aber dieser kann auf der GPU ausgeführt werden, wodurch die Laufzeit im Endeffekt kürzer sind. Bei der Beschreibung von lokalen Deep-Learning Features verwenden die Tools eine hohe Auflösung (1024×1024 Pixel), wobei die untersuchten Bilder in mehrere Unterschiedliche Auflösungen skaliert werden (von 0,25 bis 2,0). Das Beschreiben der Bilder mit 7 unterschiedlichen Auflösungen führt dazu, dass ASMK-HOW die angegebenen 28 min 42 s, statt 3 min48 s (bei nur 1 Skalierung) für die Erstellung der Datenbank benötigt.

Von allen Tools sticht aber DELF am meisten heraus. Vom Konzept ähnelt DELF ziemlich stark ASMK-HOW (dem zweit-langsamsten Tool) und DELF verwendet auch Features, die aus vielen Skalierten Versionen hochauflösender Bilder bestehen. Aber im Unterschied zu ASMK-

HOW ist DELF schlechter für Laufzeit optimiert. Während die Implementation von ASMK alle lokalen Features im Arbeitsspeicher behält, speichert DELF die Features und die ASMK-Aggregation auf der Festplatte. Dieser Unterschied lässt sich auch in Tabelle 3 beobachten. Hierdurch hat die DELF Pipeline viel mehr Festplatten-Zugriffe (Bilder lesen, Features schreiben, Features lesen, Aggregation schreiben). Außerdem kann DELF mit dieser Pipeline nicht von Batch-Verarbeitung profitieren. Ein weiterer wichtiger Unterschied zwischen der Implementation von DELF und ASMK ist die Speicherform der aggregierten Features. ASMK verwendet hierfür einen „inverted file index“, der die Spärlichkeit der Features ausnutzt um die kNN Suche zu vereinfachen. Außerdem wird durch diese Speicherform die komplette Datenbank in einer Datei gespeichert, wodurch das laden der Datenbank von der Festplatte viel weniger Zeit in Anspruch nimmt, als bei DELF (DELF sollte laut dem Paper eine IVF Struktur verwenden, die nicht implementiert wurde, wodurch DELF das einzige Tool ist, dass Features für jedes Bild einzeln speichert). Die Effekte dieser Speicherform sieht man auch bei den Laufzeiten von den Anfragen. Eine DELF Anfrage auf Oxford5k benötigt dabei über 2 min, um die Datenbank zu laden und ca. 13 s-14 s für die kNN Suche. Die Laufzeit ist dabei kaum bis gar nicht davon abhängig, wie viele Ergebnisse erwartet werden. Diese Beobachtung trifft dabei laut Tabelle 5 auf alle Tools zu.

Bei den Anfragen von ASMK-HOW und bei den globalen Deep-Learning Tools ist, ähnlich wie bei DELF erkennbar, dass das laden der Datenbank und vom Deep-Learning Modell relativ viel Zeit kostet. Diese Zeit wird bei der Verarbeitung von 100 Anfragen wieder ausgeglichen, weil die Beschreibung und die kNN Suche im Vergleich weniger Zeit benötigen. Besonders stark ist das bei den globalen Tools zu beobachten. Bei der Untersuchung von Diffusion sieht man auch, dass es bei Oxford zu keinen zusätzlichen Kosten bei den Anfragen gibt, und dass die Erstellung vom Diffusions-Graphen nur 22 s benötigt.

Bei den lokalen Hand-crafted Tools wurden neben den normalen Experimenten auch Vergleiche durchgeführt, die den Einfluss der Codebook-Größe auf die Laufzeiten untersuchen. Dabei haben größere Codebooks mehr Zeit benötigt, um erstellt zu werden (nicht gemessen), und um Quantisation der lokalen Features durchzuführen. Außerdem führen sehr große Codebooks (siehe „VLAD - SIFT (2048 W.)“ und „ASMK - SIFT (1048576 W.)“ in Tabelle 5) dazu, dass die Ladezeit der Datenbank für eine Anfrage viel langsamer wird. Dafür haben die Tools mit größeren Codebooks im Verhältnis eine höhere Verbesserung der Laufzeit bei 100 Anfragen. Bei VLAD liegt das vermutlich an der Größe (21,26 GB laut Tabelle 3) der Datenbank, wodurch allein das laden von der Festplatte ein Großteil der Zeit darstellt. Bei ASMK spielt das Codebook vermutlich eine größere Rolle. Das Codebook mit 1 Million Wörtern ist 500 MB groß, wodurch die Zeit für 1 Anfrage stark beeinflusst wird. Allerdings führt das Codebook auch dazu, dass die Repräsentationen noch Spärlicher verteilt sind, als bei 65 tausend Wörtern. Durch diese Verteilung könnte der Index von kürzeren Zeiten für die kNN Suche profitieren, die bei Oxford aber nicht zu beobachten ist.

Bei dem Vergleich zwischen den Hand-crafted Deskriptoren hat ORB eine viel bessere Laufzeit als SIFT, die aber mit schlechterer Qualität der Ergebnisse in Tabelle 1 erreicht wird.

Laufzeiten NUS-WIDE In Tabelle 6 sind die gemessenen Laufzeiten für die Arbeit der Tools mit NUS-WIDE. Die Zeiten zur Erstellung können größtenteils als eine Skalierung der

Laufzeit ROxf5k				
Toolname	Erstellung	1 Anfrage 100 Ergebnisse	1 Anfrage 1000 Ergebnisse	100 Anfragen 100 Ergebnisse
VLAD - SIFT (2048 W.)	24 min	24,39 s	25,09 s	3 min 04 s
VLAD - SIFT (128 W.)	12 min 4 s	0,71 s	0,82 s	24,93 s
VLAD - ORB (2048 W.)	8 min 35 s	2,18 s	2,66 s	46,51 s
ASMK - SIFT (65536 W.)	9 min 38 s	1,25 s	1,30 s	51,59 s
ASMK - SIFT (2048 W.)	8 min 40 s	0,85 s	0,82 s	1 min 03 s
ASMK - SIFT (1048576 W.)	14 min 18 s	10,38 s	10,68 s	1 min 20 s
ASMK - ORB (65536 W.)	5 min 25 s	1,14 s	1,33 s	45,56 s
DML (ResNet50)	46,20 s	6,73 s	6,87 s	7,58 s
GeM(AP) (ResNet50)	3 min 23 s	12,21 s	12,31 s	15,04 s
GeM(AP) (ResNet101)	5 min 26 s	12,33 s	12,16 s	17,84 s
GeM(AP) (ResNet101) + diffusion	5 min 41 s	12,41 s	12,27 s	17,73 s
DELF (ResNet50, 65536 W.)	20 h 08 min	2 min 40 s	2 min 39 s	25 min 18 s
ASMK–HOW (ResNet50, 65536 W.)	28 min 42 s	8,28 s	8,30 s	83,67 s

Tabelle 5: Vergleich Laufzeiten ROxf5k.

Ergebnisse aus Tabelle 5 betrachtet werden. Die Geschwindigkeit (in Bildern/Sekunde) von allen Hand-crafted Tools, bis auf ASMK-SIFT mit 2048 Wörtern, lag bei NUS-WIDE innerhalb von ±10% der Geschwindigkeit von Oxford5k. ASMK-SIFT mit 2048 Wörtern war bei NUS-WIDE 19,9% langsamer, was man damit erklären könnte, dass die Anzahl der Wörter zu gering ist, um eine Sinnvolle IVF-Struktur zu erstellen (mit so vielen Bildern). Diese Vermutung wird auch bei den Anfragen bestätigt, wo das Experiment mit 2048 Wörtern die längste Laufzeit bei einer und noch offensichtilicher bei 100 Anfragen hat.

Die Deep-Learning Tools haben bei der Erstellung von NUS-WIDE (DELF ausgeschlossen, da genaue Zeit Fehlt) alle eine Messbare Verbesserung der Geschwindigkeit gezeigt. Diese Tools haben auch bei einzelnen Anfragen eine mit Oxford vergleichbare Laufzeit. Bei 100 Anfragen werden die Zeitunterschiede etwas größer, hier können vor allem die globalen Tools eine sehr gute Skalierbarkeit demonstrieren.

Bei Diffusion hat die Vergrößerung des Datensatzes zu sehr großen Veränderung der Geschwindigkeit geführt. Die Erstellung des Diffusions-Graphen ist bei NUS-WIDE $5,83\times$ langsamer als bei Oxford (beim Vergleich der Zusatzkosten durch Diffusion) und auch die Zeit für eine Anfrage ist viel höher im Vergleich zu GeM(AP). Aber Diffusion hat bei NUS-WIDE kaum Zusatzkosten für die Verarbeitung von 100 Anfragen (im Vergleich zu 1 Anfrage). Ausgehend aus dieser Beobachtung kann man vermuten, dass ein Großteil der Zeit bei einer Anfrage für das laden vom Diffusions-Graphen verbraucht wird.

4.4 Qualitative Analyse

Dieser Abschnitt untersucht mithilfe einiger Beispiele die genauen Unterschiede zwischen den Tools. Für den Vergleich werden die ersten Ergebnisse verwendet, und wenn nicht explizit erwähnt, wird das Anfragebild aus Ergebnissen entfernt. Die gefundenen Unterschiede werden basierend auf der Funktionsweise erklärt. Es wird zunächst die Spezialisierung auf Gebäude

Laufzeit NUS-WIDE			
Toolname	Erstellung	1 Anfrage 100 Ergebnisse	100 Anfragen 100 Ergebnisse
VLAD - SIFT (128 W.)	7 h 56 min	5 min 05 s	12 min 04 s
VLAD - ORB (256 W.)	2 h 41 min	5,18 s	1 min 20 s
ASMK - SIFT (65536 W.)	6 h 04 min	17,28 s	5 min 38 s
ASMK - SIFT (2048 W.)	7 h 26 min	24,09 s	23 min 19 s
ASMK - SIFT (1048576 W.)	10 h 27 min	23,35 s	2 min 56 s
ASMK - ORB (65536 W.)	3 h 59 min	6,65 s	5 min 44 s
DML (ResNet50)	13 min 25min	7,10 s	12,03 s
GeM(AP) (ResNet50)	1 h 45 min	14,11 s	20,17 s
GeM(AP) (ResNet50) + diffusion	2 h 45 min	2 min 25 s	2 min 28 s
DELF (ResNet50, 65536 W.)	>400 h	37 min 57min	3 h 03h
ASMK–HOW (1 scale) (ResNet18, 65536 W.)	3 h 25 min	9,45 s	1 min 39 s
ASMK–HOW (5 scales) (ResNet18, 65536 W.)	7 h 22 min	10,63 s	1 min 54 s

Tabelle 6: Vergleich Laufzeiten NUS-WIDE.

mit Oxford5k, und danach die allgemeine Objekt-/Konzept-Erkennung mit NUS-WIDE untersucht. Am Ende von diesem Abschnitt wird eine Zusammenfassung der Stärken, Schwächen und Besonderheiten der Gruppen und Tools beschrieben.

Abbildungen bestehen aus einem blau umrandeten Anfragebild, an das von links nach rechts sortiert die entsprechenden Ergebnisse angehängt wurden.

4.4.1 Beispiele Oxford

Das erste Beispiel in Abb. 11 zeigt eine einfache Anfrage, bei der jedes Tool 10 ähnliche Bilder finden konnte. Solch eine Anfrage sagt wenig über die Tools aus, aber anhand von ihr kann man sehen, dass alle Tools Bilder finden können, die ein Zentrales Objekt aus ähnlichen Winkeln mit guter (und gleicher) Beleuchtung beinhalten. In den Abbildungen 12, 13 und 14 wird auch dargestellt, wie die Tools mit einfachen Transformationen, wie Rotation und Skalierung, sowie mit Bildausschnitten von derselben Anfrage zurecht kommen. Von den Bildern kann man sehen, dass alle Tools mit einem repräsentativen Bildausschnitt auskommen können.

Bei der Untersuchung von Rotation haben die lokalen Tools keine Probleme mit der Wiedererkennung des Gebäudes im Rotierten zustand gehabt. Die globalen Methoden hingegen haben bei dieser Anfrage nur eine Ähnlichkeit mit dem gesuchten Gebäude sehen, weshalb viele (aber nicht alle) Ergebnisse das Richtige Gebäude beinhalten. Allerdings scheinen diese Methoden nicht vollständig Rotationsinvariant zu sein.

Bei der Skalierung wurde die Auflösung von jeder Seite der Anfrage halbiert. Dadurch wurden die Ergebnisse vieler Tools beeinflusst (DML skaliert intern auf eine geringere Auflösung, weshalb es das einzige Tool war ohne Änderungen), wobei alle Tools, immer noch fehlerfrei dasselbe Gebäude erkennen konnten. Die Ergebnisse durch globale Tools haben dabei mehr Wert darauf gelegt, dass das Gebäude in der Mitte vom Bild ist, als die Tools mit lokalen Features.

Abb. 15 ist eine Demonstration davon, dass die Annotationen von ROxford5k nicht direkt Ähnlichkeit darstellen, und dass die quantitativen Ergebnisse aus der Tabelle 1 als alleiniger Indikator für die Qualität der Ergebnisse nicht ausreicht. In dieser Abbildung sind links (A und C) Beispiele von zwei Anfragen mit subjektiv ähnlichen Ergebnissen. Auf der rechten Seite sind

Ergebnisse für dieselben Anfragen, die von der Semantik des Datensatzes als korrekt bewertet werden. Die Besonderheit dieser Beispiele kommt davon, dass es sich um einen kleinen Teil des Bildes handelt und die Tools können ihren Fokus auf das „falsche“ Objekt richten. Dieser Fokus ist auch besonders interessant in Abb. 16, bei dem eine Anfrage untersucht wird, bei der eine Blumen-ähnliche Pflanze im Vordergrund ist und eine Turmspitze hinter einem Gebüsch zu sehen ist. ASMK-SIFT hat bei dieser Anfrage den Turm gefunden, es hat aber auch vermutlich die vielen organischen Kanten aus dem Gebüsch verwendet, um eine Ähnlichkeit zu den Bildern mit Himmel und Wasser zu finden. Bei DML waren die Unterschiede so groß, weshalb zwei Ergebnisse in Abb. 16 übernommen wurden. Das Erste Ergebnis wurde von einem Modell gefunden, das ein relativ kurzes Feintuning auf Oxford gemacht hat. Bei der Anfrage kommt das Modell dazu, die Pflanzen im Vordergrund wiederzufinden. Das zweite DML Modell hingegen, das auf einem reinen Gebäudedatensatz (Sfm120k) trainiert wurde, hat einen viel stärkeren Fokus auf die Gebäudespitze. Das Modell findet aber danach nicht das richtige Gebäude, sondern eins mit einer ähnlichen Spitze. Dieser Fehler kann mit der internen Auflösung von DML (224×224 Pixel) erklärt werden.

Solche Probleme sind auch in Abb. 17 zu beobachten, wo DML möglicherweise vertikale Strukturen, wie eine Tür oder eine Leiter mit den Säulen von einem Gebäude verwechselt.

Das letzte Beispiel aus Abb. 16 ist das Ergebnis von ASMK-HOW. Bei dem Tool wurde das Modell und das Codebook auf Sfm erstellt, wodurch ein sehr starker Fokus auf die Turmspitze zu beobachten ist. ASMK-HOW arbeitet auch mit mehreren hohen Auflösungen, wodurch der Turm richtig wiedererkannt werden konnte.

In Abb. 18 kann man sehen, wie die Verwendung eines Diffusion-Graphen die Ergebnisse beeinflusst. Diffusion hat im Durchschnitt den mAP verbessert, allerdings hatte Diffusion auch einen negativen Einfluss auf den mP@10 Wert. Beim betrachten des Beispiels 18 sieht man, dass Diffusion einen sehr starken Fokus auf einen Aspekt des Bildes legt. Das Bild beinhaltet Äste und das Gebäude, aber Diffusion konzertiert sich dabei aber nur auf das Gebäude, sodass sogar das Anfragebild nicht wiedergefunden wird. Dieser Fokus kann theoretisch dazu führen, dass Diffusion sich auf das falsche Objekt konzentriert, aber bei Oxford ist es häufiger das gesuchte Objekt.

Ein Weiterer Unterschied zwischen den untersuchten Gruppen ist der Umgang mit unterschiedlichen Lichtverhältnissen. In Abb. 19 wird die Beobachtung demonstriert, dass Hand-crafted Tools keine sinnvollen Ergebnisse mit anderer Beleuchtung finden können. Dieses Problem wird aber sehr gut von allen Deep-Learning Methoden gelöst.

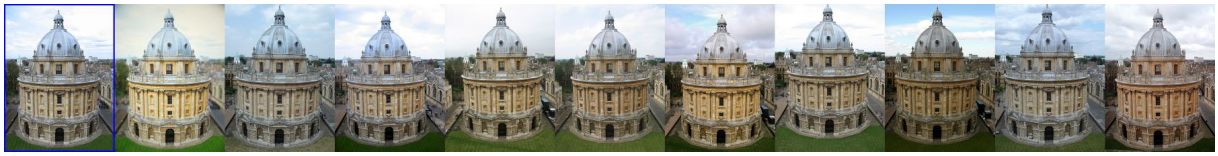
Das letzte Beispiel aus dem Oxford Datensatz zeigt ein weiteres Problem, das lokale Hand-crafted Deskriptoren haben. In der Anfrage von Abb. 20 ist vor dem Gebäude eine große Anzahl an Ästen, die aus vielen Kanten bestehen, auf die ein lokaler Hand-crafted Deskriptor besonders achtet. Dieser Fokus auf Kanten führt dazu, dass alle Ergebnisse auch viele Kanten beinhalten. Deep Learning Verfahren hingegen können viel besser bestimmte Objekte erkennen und vom Rausch trennen.

(a) VLAD - SIFT (2048 W.)

(b) ASMK - SIFT (65536 W.)

(c) DML (ResNet50, trained on Sfm)

(d) GeM(AP) (ResNet101)

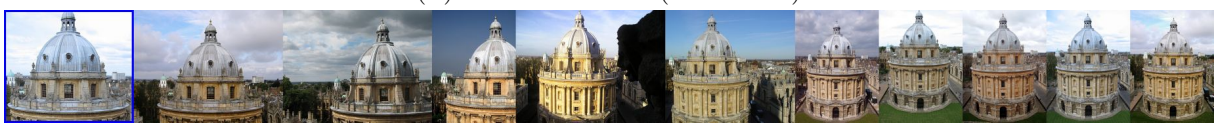


(e) DELF (ResNet50, 65536W.)

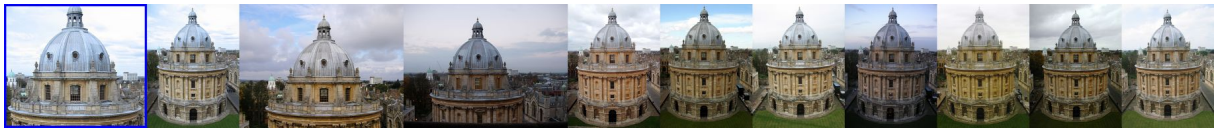
(f) ASMK-HOW (ResNet50, 65536W.)

Abbildung 11: Vergleich aller Tools anhand einer einfachen Anfrage, bei der alle untersuchten Tools eine P@10 von 100% haben.

(a) ASMK - SIFT (65536 W.)



(b) DML (ResNet50, trained on Sfm)



(c) ASMK - HOW (ResNet50, 65536 W.)

Abbildung 12: Ausschnitt aus Abb. 11

(a) ASMK - SIFT (65536 W.)

(b) DML (ResNet50, trained on Sfm)

(c) ASMK - HOW (ResNet50, 65536 W.)

Abbildung 13: Rotation von Abb. 11

(a) ASMK - SIFT (65536 W.)

(b) GeM(AP) (ResNet101)

(c) ASMK - HOW (ResNet50, 65536 W.)

Abbildung 14: Skalierung von Abb. 11 auf ein Viertel der Auflösung

(a) DML (ResNet50, trained on Sfm), ähnliche falsche Ergebnisse

(b) ASMK - SIFT (65536 W.), weniger ähnliche korrekte Ergebnisse

(c) GeM(AP) (ResNet50), ähnliche falsche Ergebnisse

(d) ASMK - HOW (ResNet50, 65536 W.), weniger ähnliche korrekte Ergebniss

Abbildung 15: Semantische Unterschiede zwischen visueller Ähnlichkeit und Korrektheit im Sinne der Annotation von ROxf5k.

(a) ASMK - SIFT (65536 W.)

(b) DML (ResNet50, trained on Oxf)

(c) DML (ResNet50, trained on Sfm)

(d) ASMK - HOW (ResNet50 65536 W.)

Abbildung 16: Voller Vergleich Fokus auf Objekte (Bestimmung relevanter Objekte)

Abbildung 17: Demonstration von Fehlern, die vermutlich durch geringe Auflösung bei DML entstehen (4. und 8. Ergebnis haben vertikale Objekte, die einer Säule ähneln könnten).

(a) GeM(AP) ohne Diffusion

(b) GeM(AP) mit Diffusion

Abbildung 18: Unterschiede, die durch die Verwendung von Diffusion entstehen. Bei diesem Beispiel wird das Anfragebild nicht aus den Ergebnissen entfernt, da es bei Difffusion nicht zu den ersten 5 Ergebnissen gehört.

(a) ASMK - SIFT (65536 W.)

(b) ASMK - HOW (ResNet50 65536 W.)

Abbildung 19: Umgang mit wechselnden Lichtverhältnissen. Deep Learning Tools können im Unterschied zu Hand-crafted Tools das Gebäude am Tag wiedererkennen.

(a) ASMK - SIFT (65536 W.)

(b) ASMK - HOW (ResNet50 65536 W.)

Abbildung 20: Vergleich Verdeckung durch kantige Äste

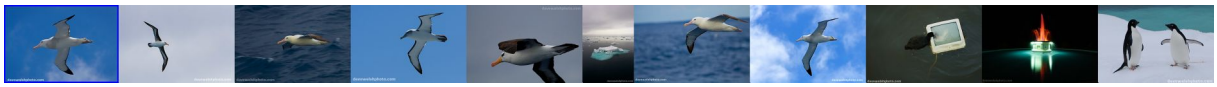
4.4.2 Beispiele NUS-WIDE

Bei NUS-WIDE ist die Herausforderung, ein Objekt zu erkennen und aus aus einer sehr großen Datenbank herauszusuchen. Im Vergleich zu Oxford sind die Bilder vielfältiger, wodurch auch viele einfache Bilder verfügbar sind. Dadurch müssen die Tools für NUS-WIDE in der Lage sein, unterschiedliche Konzepte zu beschreiben und mit einer hohen Sicherheit von allen anderen zu unterscheiden.

Das erste Beispiel hierfür ist die Anfrage aus Abb. 21, bei der nur eine Möwe auf größtenteils blauem Hintergrund zu sehen ist. Alle globalen Deep-Learning Methoden konnten diese Möwe zu anderen fliegenden Vögeln zuordnen. Bei den lokalen Methoden kam es aber häufiger zu Verwechslungen. ASMK-HOW zum Beispiel hat neben den gesuchten Bildern auch Bilder gefunden, die Flugzeuge oder Wolken beinhalten. Am weitesten von den gesuchten Bildern waren die Hand-crafted Tools entfernt. Das in Abb. 21 verwendete Ergebnis war das beste aus seiner Gruppe (hand-crafted), und auch in dem Beispiel werden Bilder von Eisschollen, Pinguinen und einer Fontäne gefunden, wobei die Ergebnisse im Schnitt noch verwendbar sind.

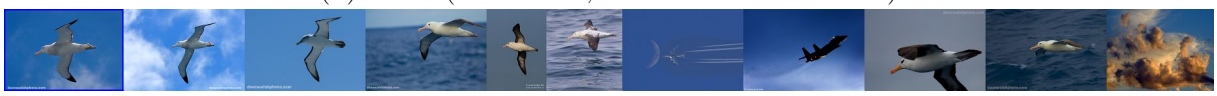
Anders sieht das bei den Beispielen aus Abbildungen 22 und 23 aus. In 22 hat das Bild einen schwarzen Rand, der von den Hand-crafted Features als starke Kante gesehen wird, wodurch fast alle Ergebnisse auch einen schwarzen Rand haben, aber abgesehen davon keine Ähnlichkeit zur Anfrage existiert. Ein ähnliches Problem ist auch bei ASMK-HOW in dem zweiten Beispiel von Abb. 22 mit dem weißen Rand zu beobachten.

In der nächsten Abb. 23 mit dem FLugzeug kann man besonders gut sehen, dass SIFT Features bei diesen Bildern nicht die Objekte erkennen können. Dadurch werden häufig Bilder als ähnlich identifiziert, obwohl sie ein anderes Objekt enthalten und somit für einen menschlichen



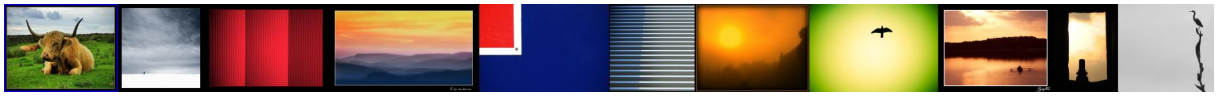
(a) ASMK - SIFT (65536 W.)

(b) DML (ResNet50, 30 Epochen Training)



(c) ASMK - HOW (65536 W.)

Abbildung 21: Objekterkennung mit einfachen Hintergrund



(a) ASMK - SIFT (65536 W.)

(b) ASMK -HOW (ResNet18, 65536 W.)

Abbildung 22: Probleme mit Rand im Bild

Betrachter unterschiedlich aussehen.

Bei dem Beispiel in Abb. 24 wird untersucht, wie gut ein Bild und das abgebildete Objekt anhand von einem Ausschnitt erkannt werden kann. Beide lokalen Methoden haben dabei das ursprüngliche Bild wiederfinden und an erster Stelle einsortieren können. Bei den globalen Methoden war die Wiedererkennung der Anfrage etwas schlechter, aber diese Methoden hatten, genauso wie in Abb. 23 insgesamt mehr Flugzeuge gefunden als die anderen Gruppen. Bei einer genaueren Betrachtung der Ergebnisse von DML und ASMK-HOW sieht man auch, dass DML einen höheren Fokus auf die Flugzeugnase legt.

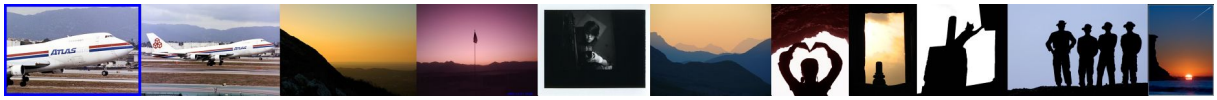
Abb. 25 zeigt den Einfluss von längerem fein-tuning. In dem Vergleich sind die Ergebnisse von einem Modell, das für 3 Epochen auf NUS-WIDE trainiert wurde, sowie ein Modell, das für 30 Epochen trainiert wurde. Beide Modelle wurden auf Imagenet vortrainiert. In der untersuchten Anfrage wird ein Bild mit einer Libelle gesucht, welches in NUS-WIDE als „animal“ annotiert wurde. Durch diese ungenaue Annotation hat das DML Modell mit mehr Training eine schlechtere Beschreibung für das Anfragebild gefunden, die aber immer noch ein perfektes Ergebnis bei der quantitativen Analyse bekommen würde.

Die letzte Abb. 26 aus NUS-WIDE untersucht das Verhalten der Tools bei einer ungewöhnlichen Anfrage. Die meisten Tools haben dabei Bilder mit runden Objekten gefunden, aber GeM(AP) mit ResNet101 konnte besonders gute Ergebnisse mit dem selben „Little Planet“ Effekt finden, wie die Anfrage (es wäre ziemlich unwahrscheinlich, dass das Modell mit solchen Effekten trainiert wurde).

(a) ASMK - SIFT (65536 W.)

(b) ASMK - HOW (ResNet18, 65536 W.)

Abbildung 23: Beispiel SIFT-Features erkennen keine Objekte, wodurch sich Ergebnisse stark von der Anfrage unterscheiden können.

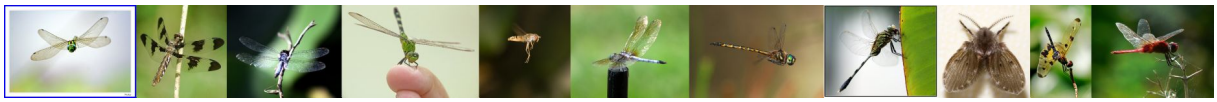


(a) ASMK - SIFT (65536 W.)

(b) DML (ResNet50, 30 Ep. Training)

(c) ASMK - HOW (ResNet18, 65536 W.)

Abbildung 24: Beispiel Ausschnitt von Anfrage aus Abb. 23



(a) DML (ResNet50, 3 Ep. Training)

(b) DML (ResNet50, 30 Ep. Training)

Abbildung 25: Training von DML mit sehr groben Annotationen von NUS-WIDE

Abbildung 26: Positivbeispiel Abstrakte Bilder (GeM(AP) ResNet101)

4.5 Zusammenfassung Evaluation

Im Laufe der Untersuchung der Ergebnisse von den unterschiedlichen Tools sind Deep-Learning Features als besonders gut aufgefallen, weil sie besser einzelne Objekte erkennen können. Dadurch sind die gefundenen Bilder näher an einem Menschlichen Verständnis von Ähnlichkeit als Ergebnisse, die mit Hand-crafted Features gefunden werden. Ein Beispiel hierfür ist in Abb. 16 zu sehen, wo die „falschen“ Bilder von Deep Learning Methoden ähnlicher Aussehen, als die „falschen“ Bilder der Hand-crafted Methoden. Bei ROxford5k wird dieses Problem minimiert, indem der Fokus nur auf bestimmte anorganische Features von Gebäuden gerichtet wird. Ohne diesen Vorteil erreichen Hand-crafted Methoden, wie man in Tabelle 2 keine guten Ergebnisse.

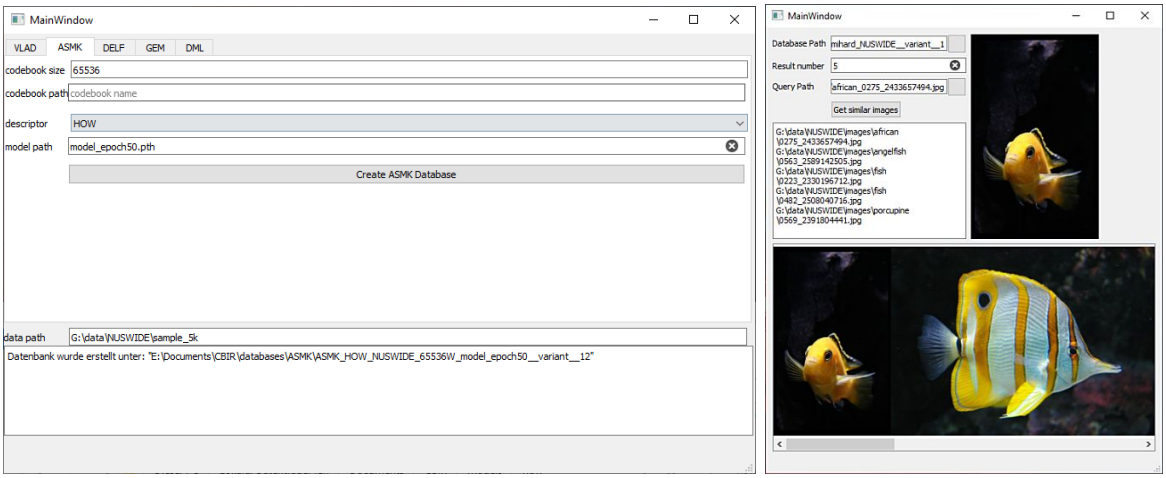
Bei dem Vergleich zwischen globalen und lokalen Deskriptoren haben lokale Features tendenziell bessere Ergebnisse erreicht, wenn nur ein Teil des Bildes wichtig ist. Diese Fähigkeit ist vor allem gut, um die Qualität innerhalb einer bestimmten Anwendung zu verbessern. Lokale Methoden können dadurch zum Beispiel besser mit Verdeckungen umgehen, ohne auf das verdeckende Objekt abgelenkt zu werden. Bei der allgemeineren Herausforderung von NUS-WIDE hatten die globalen Methoden etwas bessere Ergebnisse.

Die Verwendung eines Diffusions-Graphen hat sich nicht als eine zuverlässige Möglichkeit gezeigt, die Qualität der Ergebnisse zu verbessern. Durch Diffusion konzentrieren sich die ersten Ergebnisse mehr auf ein bestimmten Aspekt vom Anfragebild. Mit aussagekräftigen Features kann dies zu einer quantitativen Verbesserung führen, aber es kann auch die Ergebnisse verschlechtern.

Bei der Untersuchung anderer Faktoren, hat sich gezeigt dass die Tools VLAD und DELF zu große Nachteile haben, um praktisch brauchbar zu sein. VLAD Features generieren eine viel zu große Datenbank, wodurch Laufzeiten und Codebook-Größe, und somit auch Qualität negativ beeinflusst wird. Bei DELF liegt das Problem darin, dass die Laufzeiten der Implementation viel zu hoch ist.

Von den Deskriptoren haben alle Deep-Learning Methoden die besten Ergebnisse geliefert, wobei diese auch den höchsten Rechenaufwand haben. Durch die Verwendung einer Grafikkarte konnten die Laufzeiten ausgeglichen werden, wodurch globale Deep-Learning Methoden die besten Laufzeiten gehabt haben, gefolgt von ORB, SIFT und dann lokalen Deep-Learning Features. Die ORB Features haben keine gute Qualität zeigen können, weshalb diese größtenteils ignoriert wurden. Der Grund dafür, dass Deep-Learning die besten und die schlechtesten Laufzeiten hat, liegt daran, dass die lokalen Methoden Features aus mehreren Skalierungen desselben Bildes beziehen und die globalen Methoden mit einem Durchlauf auskommen. Der Vergleich zwischen den einzelnen ResNet Größen hat auch nachgewiesen, dass größere Netze genauer und langsamer sind.

Um die Funktionsweise der Tools zu ermöglichen, muss auch eine Vorbereitung gemacht werden. Dazu gehört das Training von Deep-Learning Modellen mit passenden annotierten Daten und die Erstellung eines Codebooks für lokale Methoden, die keine Annotation erfordert. Dadurch ist der Aufwand für ein CBIR-Tool mit Deep Learning höher als für ein Tool ohne, wobei lokale Deep-Learning Methoden mit Abstand die aufwändigste Vorbereitung haben (weil beide Schritte gemacht werden müssen und hohe Auflösungen notwendig sind).



(A) Datenbank erstellung (B) Anfrage

Abbildung 27: Screenshots der beiden Fenster von der GUI.

5 Zusammenfassendes Tool mit GUI

Im Rahmen dieser Arbeit wurde ein Programm zusammengestellt, das alle untersuchten Tools beinhaltet. Das Ziel dabei war es, ein GUI - Interface zu erstellen, um den Vergleich auf beliebigen Daten zu ermöglichen. Die aus diesem Programm resultierende GUI ist in Abb. 27 Dargestellt. Das Interface zur Erstellung einer Datenbank ist im Teil (A) zu sehen und beinhaltet eine Tool-Auswahl mit einigen spezifischen Parametern. Für die Erstellung einer Datenbank ist nur ein Ordner mit den Bildern, sowie ein trainiertes Modell bei Deep-Learning Tools notwendig. Das Codebook für lokale Features kann bei der Erstellung der Datenbank generiert werden (oder man nutzt ein vorhandenes).

Nach Erstellung der Datenbank werden alle nötigen Dateien in einem Ordner abgespeichert, der für eine Anfrage verwendet werden kann. Die GUI dafür ist in Teil (B) von Abb. 27 zu sehen und benötigt keine weiteren Parameter, außer den Datenbank - Ordner (und die gesuchte Anzahl der Ergebnisse), um eine Anfrage bearbeiten zu können.

Das Training von Deep-Learning Modellen ist in dem Tool nicht vorgesehen und muss innerhalb der einzelnen Tools durchgeführt werden. Das Training wäre auch schwer zu verallgemeinern aufgrund der Tatsache, dass Annotationen notwendig sind und eine Implementation nur ein bestimmtes Format nutzen könnte.

Die Implementationen der einzelnen Tools sind von den Autoren der jeweiligen Tools übernommen: [7, 20, 10, 11, 19, 2, 25, 15, 21]. Es wurden einige Anpassungen unternommen, wie zum Beispiel die Batch-Verarbeitung von ASMK.

6 Ausblick und Verbesserungen

Im Laufe dieser Arbeit mussten viele Kompromisse aufgrund begrenzter Zeit und Hardware getroffen werden. Beim evaluieren der auf NUS-WIDE trainierten Modelle ist aufgefallen, dass die Ergebnisse nicht so gut sind, wie es die quantitativen Maße vorgegeben haben. Um dies zu verbessern müsste ein ausführlicheres Training mit besseren Annotationen durchgeführt werden. Bei dem Training auf NUS-WIDE musste auch darauf verzichtet werden, ASMK-HOW mit ResNet50 zu verwenden, weil so die VRAM Größe von 8 GB überschritten wurde (unabhängig

von allen einstellbaren Parametern). Das verwendete Ergebnis konnte trotzdem die Stärken und Schwächen der Gruppe repräsentieren und zu einem Sinnvollen Vergleich beitragen. Die Verwendung von einem kleinen Netz zeit aber auch, dass ASMK noch Luft nach oben hat.

Beide der untersuchten lokalen Deep-Learning Methoden haben theoretisch die Möglichkeit bedacht, einen zusätzlichen globalen Deskriptor mit zu benutzen, um die Vorteile beider Gruppen zu kombinieren. Diese kombinierten Methoden könnten die besten Ergebnisse mit einer relativ hohen Laufzeit liefern, konnten hier aber nicht genauer betrachtet werden.

7 Zusammenfassung

In dieser Arbeit wurde ein Vergleich zwischen drei unterschiedlichen Deskriptor - Arten für Content-based Image Retrieval anhand von sieben unterschiedlicher Implementationen durchgeführt. Im unterschied zu vorherigen Arbeiten wurde der Vergleich nicht nur quantitativ, sondern auch qualitativ durchgeführt, was einen besseren Einblick in die Stärken und Schwächen der Methoden ergeben hat. Der Vergleich wurde auch auf den NUS-WIDE Datensatz ausgeweitet, für den eine neue Evaluation vorgestellt wurde. Die Implementationen der Methoden wurden auch in einem Python Programm zusammengeführt, das ein standardisiertes Interface für alle Methoden ermöglicht.

Die Ergebnisse von dem Vergleich haben ergeben, dass Deep Learning vor allem die Objekterkennung verbessert, was zu besseren Ergebnissen führt. Globale Deep Learning Features haben sich als die schnellste Methode (mit Grafikkarte) erwiesen. Sie haben bei allgemeinen Bildern die besten Ergebnisse erreicht, wobei lokale Methoden besser für spezielle Anwendungen optimiert werden konnten.

8 Literaturverzeichnis

Literatur

[1] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *European conference on computer vision*, pages 584–599. Springer, 2014.

[2] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1861–1870, 2019.

[3] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, Santorini, Greece., 2009.

[4] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In *European conference on computer vision*, pages 241–257. Springer, 2016.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition*, pages 84–92. Springer, 2015.

[7] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE transactions on pattern analysis and machine intelligence*, 34(9):1704–1716, 2011.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[9] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[10] Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies, 2017.

[11] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features, 2018.

[12] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[13] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5706–5715, 2018.

[14] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018.

[15] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5107–5116, 2019.

[16] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[17] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, IEEE International Conference on*, volume 3, pages 1470–1470. IEEE Computer Society, 2003.

[18] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1857–1865, 2016.

[19] Marvin Teichmann, Andre Araujo, Menglong Zhu, and Jack Sim. Detect-to-retrieve: Efficient regional aggregation for image search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5109–5118, 2019.

[20] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. To aggregate or not to aggregate: Selective match kernels for image search. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1401–1408, 2013.

[21] Giorgos Tolias, Tomas Jenicek, and Ondřej Chum. Learning and aggregating deep local descriptors for instance-level recognition. In *European Conference on Computer Vision*, pages 460–477. Springer, 2020.

[22] Koen Van De Sande, Theo Gevers, and Cees Snoek. Evaluating color descriptors for object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1582–1596, 2009.

[23] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1386–1393, 2014.

[24] Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning, 2018.

[25] Fan Yang, Ryota Hinami, Yusuke Matsui, Steven Ly, and Shin'ichi Satoh. Efficient image retrieval via decoupling diffusion into online and offline processing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9087–9094, 2019.

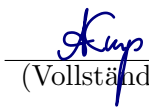
[26] Liang Zheng, Yi Yang, and Qi Tian. Sift meets cnn: A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1224–1244, 2017.

9 Eidesstattliche Versicherung

Ich versichere eidesstattlich durch eigenhändige Unterschrift, dass ich die Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht und ist in gleicher oder ähnlicher Weise noch nicht als Studienleistung zur Anerkennung oder Bewertung vorgelegt worden. Ich weiß , dass bei Abgabe einer falschen Versicherung die Prüfung als nicht bestanden zu gelten hat.

Rostock

02.06.2021 (Abgabedatum)

 (Vollständige Unterschrift)