Inhaltsverzeichnis

Schrifterkennung

mit SVMs und Neuronalen Netzen

1 Schrifterkennung mit SVMs

Max Braun, Farhadiba Mohammed, Annika Scheug

Hintergrund Wie in der Einleitung bereits erwähnt, werden in Deutschland rund 64 Millionen Briefe am Tag verschickt. Es ist nur verständlich, dass die Post anstelle vieler Angestellter, die die Briefe sortieren und dabei womöglich noch einige Fehler machen, lieber einem Programm das Sortieren überlässt. Da wir uns zunächst nicht wirklich vorstellen konnten, wie ein solches Programm funktionieren könnte, haben zwei Projektgruppen beschlossen, sich bei ihrer Arbeit mit Schrifterkennung zu beschäftigen und sich dem Problem auf zwei unterschiedliche Weisen anzunähern.
Eine Gruppe nutzt Support Vector Machines *(SVMs)*, die andere ein Convolutional Neural Network *(CNN)*.

Aufbau des Programms Da es beim maschinellen Lernen essenziell ist, über ein entsprechend großes Datenset zu verfügen, haben wir unseren Algorithmus zunächst einmal mit 60.000 Bildern ausgestattet.

```
feature, hog_image = feats.hog(sub_pic,
 orientations = 8,pixels_per_cell = (4,4),
 cells_per_block = (1,1),visualise = true)
```

Im ersten Schritt generieren wir die Features sämtlicher Bilder. Features sind z.B. Muster in einem Bild, wie Kanten und Farben oder allgemein gesagt, die Eigenschaften eines Objektes. Wie wir an dem Beispielbild mit dem Kameramann (Abb. 1.1) sehen können, wurden durch den Algorithmus die Kanten hervorgehoben und alles andere wurde ignoriert (Abb. 1.2). Dieses Featurebild kann als Vektor dargestellt werden und die SVM benutzt solche Vektoren, um die Unterschiede der Trainingsdaten zu ermitteln. Danach kann die SVM neue Objekte, hier sind es Bilder, richtig klassifizieren.
Bei dem Erstellen der Features gibt es mehrere Einstellungsmöglichkeiten, beispielsweise zur Variation der Pixelanzahl in einer Zelle (*pixels_per_cell*) und der Genauigkeit zur Erkennung von Konturen (*orientations*). Dabei gilt: Je kleiner die Zellen sind und je höher die Genauigkeit ist, desto länger braucht der Algorithmus zum berechnen, aber desto präziser ist das Endergebnis. Allerdings muss man auch

Abb. 1.1: Schwarz-Weiß Bild eines Kameramannes.

beachten, dass die Zellengröße nicht zu klein wird, da in sehr kleinen Zellen die Muster eventuell nicht für das Programm erkennbar sind.

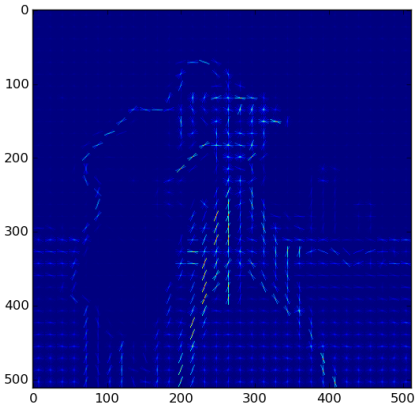

Abb. 1.2: Feature-Bild von Abb.1.1.

```
clf[:fit](features_trainset',
 label_train_data[:])
 ...
prob = clf[:predict_proba]
 (features_testset')
```

Anschließend wird die SVM mit den Bildern trainiert und ist somit einsatzbereit. Um nun die Genauigkeit der SVM zu testen, erzeugen wir ein Testset, dessen Features ebenfalls über den bestehenden Algorithmus generiert werden. Getestet wird die SVM, indem wir das Testset an die SVM übergeben und sie uns aufgrund der vorhandenen

Werte eine Schätzung abgibt, was am wahrscheinlichsten auf dem Bild abgebildet ist. Die Ergebnisse werden dann überprüft und die Genauigkeit wird errechnet.

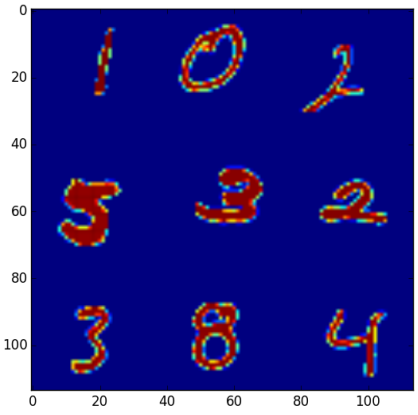

Abb. 1.3: Ausschnitt des generierten Bildes, das die SVM analysieren soll.

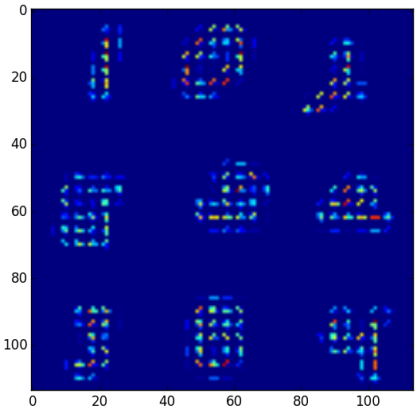

Abb. 1.4: Feature-Bild des generierten Bildes.

Daraufhin erzeugen wir ein Bild mit 10×10 Zahlen (Ausschnitt s. Abb. 1.3), die jeweils in einer 38×38 Pixelzelle zufällig verteilt werden. Dieses Bild wird in eine Matrix umgewandelt und ein Algorithmus iteriert über die Matrix, sodass die SVM bestimmen kann, welche 100 Zahlen abgebildet sind.

2 Zeichenerkennung mit neuronalen Netzen

Luca Bohn, Lukas Lück, Lukas Kamm

Nun betrachten wir neuronale Netze im Zusammenhang mit Schrifterkennung. Convolutional Neural Networks eignen sich besonders für die Verarbeitung von Bildern. Eine *Convolution* ist im konkreten Anwendungsfall mit Bildern die Zusammenfassung der Pixel in einem kleinen Bereich, die in ein neues Bild gespeichert werden. Dadurch können beispielsweise kleine Rotationen des Objekts auf einem Foto, das erkannt werden soll, kompensiert werden. Außerdem lassen sich im CNN *Feature Maps* vielseitig anwenden (näheres in Abschnitt **??**). Ein funktionsfähiges neuronales Netz zu implementieren, bedeutet einen großen Zeitaufwand und viel Feinoptimierung. Da uns diese Zeit nicht zur Verfügung steht und wir ein größeres Interesse an der Anwendung haben, nutzen wir Caffe. Bei Caffe handelt es sich um eine Software des Berkeley Vision and Learning Centers. Caffe bietet eine Art Bausatz für neuronale Netze. Man kann verschiedene Filter auswählen und sich so sein eigenes neuronales Netz erstellen. Caffe ist in der Lage, Bilder sehr schnell auszuwerten, nach Herstellerangaben benötigt es je Bild 4 ms in der Trainingsphase und 1 ms in der Testphase. Als Datensatz für die Zeichen nutzen wir MNIST. Dies ist ein freier Satz aus 28×28 px großen Bildern mit den arabischen Ziffern in Graustufe. Der Testsatz enthält 60.000, der Trainingssatz 10.000 Grafiken. Diese vorgefertigte Zeichendatenbank eignet sich vor allem für Testumgebungen im Bereich des maschinellen Lernens und das Testen von Algorithmen neuronaler Netze. Aufgrund dessen ist sie optimal für beide Schrifterkennungs-Teams geeignet.

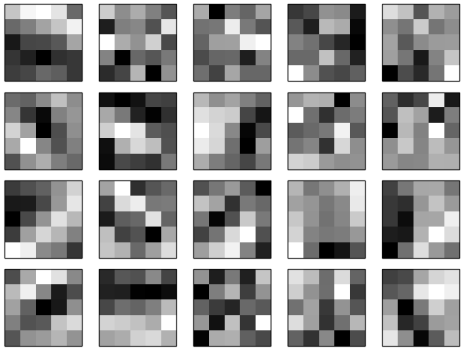

Abb. 1.5: Filter (sog. Feature Maps), spezialisiert für MNIST

2.1 Training

Voraussetzung für das maschinelle Lernen mit neuronalen Netzen ist das Anlernen mit Hilfe von Trainingsdaten. Hierzu trainieren wir Caffe mit allen 60.000 Grafiken aus dem MNIST-Datensatz.

2.2 Testen

Unser Ziel ist es, aus einem Bild mit Handschriftcharakter arabische Ziffern richtig zu erkennen. Hierzu erzeugen wir ein Bild mit 10×10 zufällig gewählten Ziffern, die jeweils um wenige Pixel von ihrer Zentrumsposition abweichen, um den Handschriftcharakter zu simulieren (vgl. Ausschnitt in Abb. 1.3). Anschließend iterieren wir pixelweise, ebenso wie die SVM-Projektgruppe, über das Bild und extrahieren wieder 28×28 px große Grafiken und übermitteln sie an das neuronale Netz.

3 Vergleich SVM und neuronales Netzwerk

3.1 Trainingsdauer

Allgemein sind SVMs schneller zu trainieren als neuronale Netze, da SVMs nur eine Schicht haben während neuronale Netze aus mehreren Schichten bestehen, in denen die Informationen verarbeitet werden, und deshalb natürlich länger zum Berechnen brauchen. In unserem Projekt waren allerdings anfangs die neuronalen Netze wesentlich schneller, was aber an der schlechteren Implementierung der SVM lag und sie dadurch nicht optimal genutzt wurde. Auch mussten wir bei der SVM zuerst die Features sämtlicher Bilder berechnen, bevor wir sie trainieren konnten. Dies war bei dem neuronalen Netz nicht notwendig, weshalb es insgesamt wesentlich schneller war, denn dieser Vorgang dauert am längsten.

3.2 Genauigkeit